Contents

Karlsruhe Institute of Technology

Probability & Computing

Overview & The Power of Randomness

Why is randomness useful in computation?

- Randomness facilitates the development of algorithms and data structures.

https://i.imgflip.com/3ajf5v.jpg?a470534

"For many applications, a randomized algorithm is the simplest algorithm available, or the fastest, or both."

"Randomized Algorithms", Motwani & Raghavan, 1995

- Sometimes a randomized approach is the *only* solution!

Idea

- Utilize randomness in algorithms and data structures to obtain much better performance than that of deterministic approaches
- But we have to pay for that ...
  - Maybe we only *expect* the approach to be fast
  - Maybe we only *expect* the approach to work correctly
- Goal: develop methods that fail only rarely

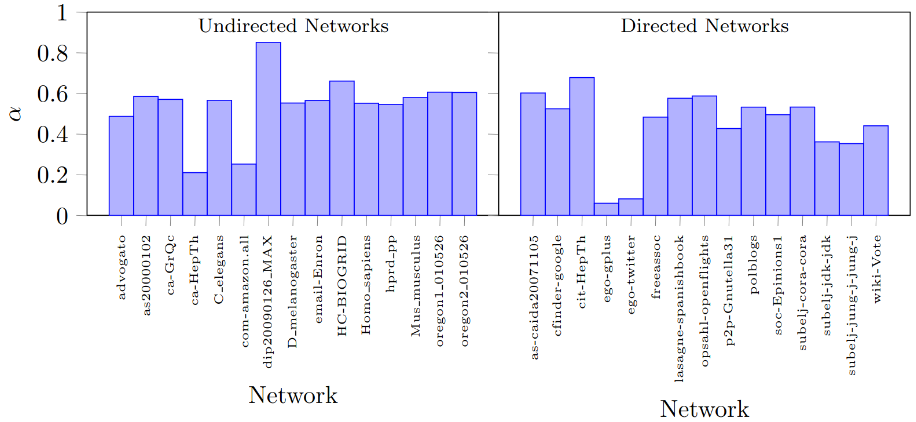
Why is randomness useful in computation?

- Useful when bridging the theory-practice gap regarding the performance of an appraoch

Theory-Practice Gap

- Algorithm performance often measured by worst-case running time (strong guarantee)
- Observe much better performance in practice than expected
- Example: bidirectional Breadth-First-Search
  - no asymptotic speed-up compared to standard BFS in the worst case
  - sublinear running time observed on many real-world networks

"KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation", Borassi & Natale, JEA, 2019



Average-Case Analysis

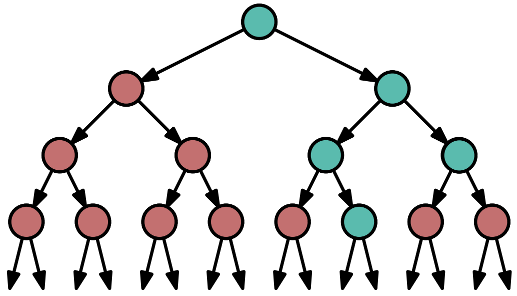
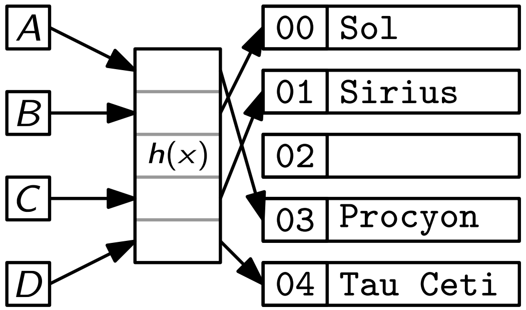
- Distinguish practical instances from the worst case
- Define probabilistic distributions (over possible inputs) that favor realistic instances
- Analyze performance assuming input is drawn from the distribution
- Expect good performance when hard instances are sufficiently unlikely

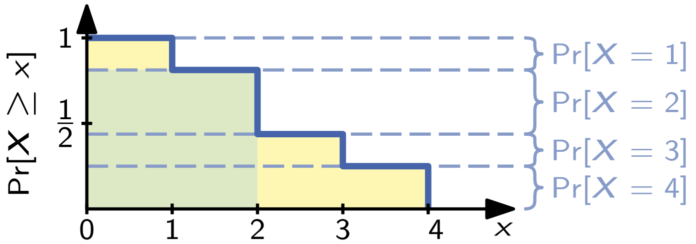
Overview

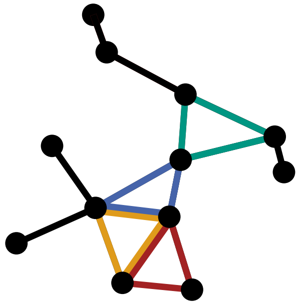
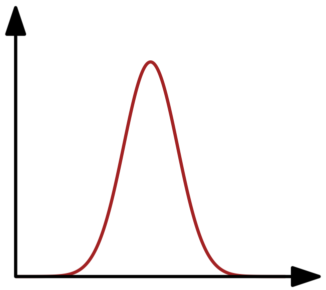
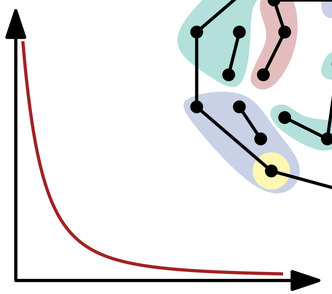
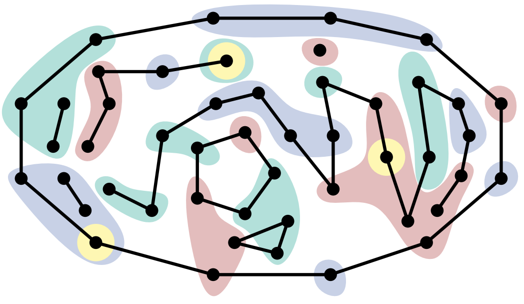
Randomized Algorithms & Data Structures
- Probability Amplification
- Streaming / Online-algorithms
- Hashing

Average-Case Analysis
- Random Graphs
- Algorithm Analysis

Toolbox
- Probabilistic Method
- Yao’s Principle
- Coupling
- Dealing with stochastic dependencies
- Concentration bounds



Organization

Team

Max
Lecture
(first part)

Stefan
Lecture
(second part)

Hans-Peter
Exercise

Thursday 11:30

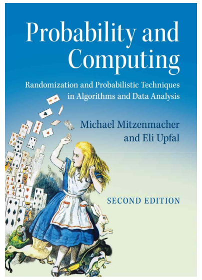
Tuesday 8:00 (every other week)

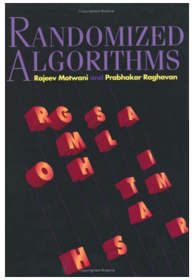
Assumed Background

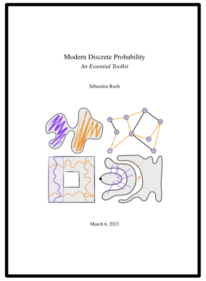
- Algorithms and data structures
- Probability theory

Material

- Slides
- Previous script
- *Probability and Computing*
- *Randomized Algorithms*
- *Modern Discrete Probability*







Website scale.iti.kit.edu/teaching/2023ws/randalg

Questions? Ilias, Discord, Matrix?

Sheets

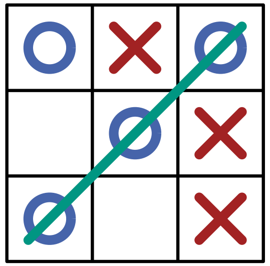
- Every week, hand in on the Thursday before the next exercise

Exam

- Oral
- Requirment: sheets handed in regularly

Power of Randomness: Let's Play a Game

Karlsruhe Institute of Technology



Tic-Tac-Toe

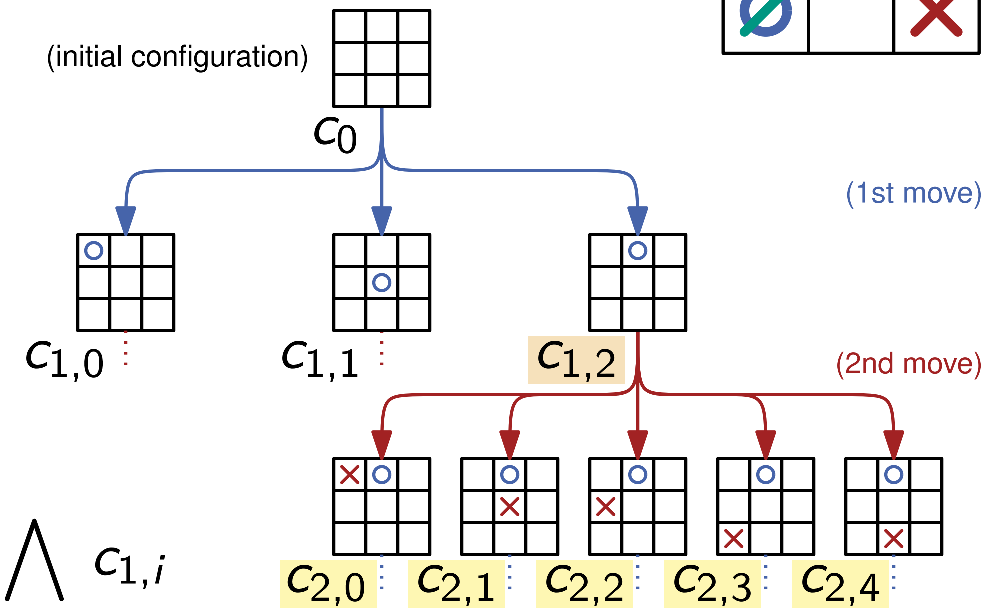
- Players take turns placing O and X in 3×3 grid
- First to get three in a line wins

Can Player 2 win the game?

Tree of Moves

- Each node is a board configuration
- A parent-child relation represents a valid move
- Label a config 1 if Player 2 can win, 0 o.w.

What label do we put on the root?



- $c_0 = 1$ if there exists *no* i such that $c_{1,i} = 0$
  or equivalently, if for *all* i we have $c_{1,i} = 1$

$$c_0 = \bigwedge_{i \in [2]} c_{1,i}$$

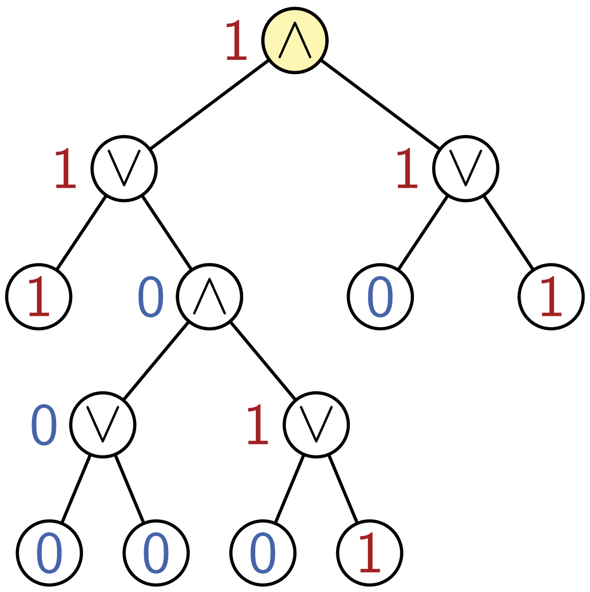
- $c_{1,2} = 1$ if there exists an i such that $c_{2,i} = 1$

$$c_0 = \bigvee_{i \in [4]} c_{2,i}$$

AND/OR-Trees

Structure

- Node types: $\wedge$-nodes, $\vee$-nodes, and leaves
- The root is a leaf or an $\wedge$-node
- $\wedge$-nodes have only $\vee$-nodes as children
- $\vee$-nodes have only AND/OR-trees as children



Evaluation

- Leaves contain boolean values
- Inner nodes evaluate to ...
  - the disjunction of their children, for $\vee$-nodes
  - the conjunction of their children, for $\wedge$-nodes

Example Complexities

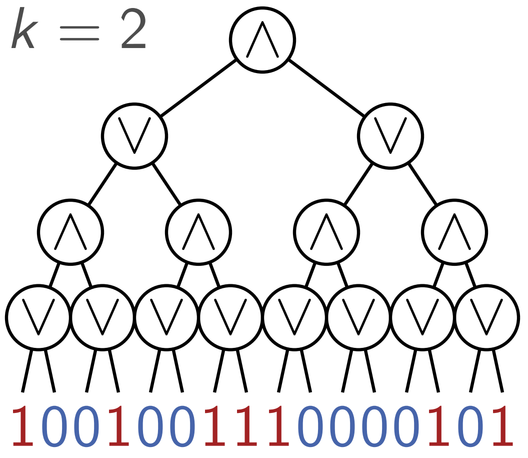
- Tic-Tac-Toe: 31896 (non-symmetric) games (leaves)
- Checkers: approx. 10^{40} leaves
- Chess: approx. 10^{123} leaves
- Go (19×19): approx. 10^{360} leaves

Deterministic Evaluation

Simplifying Assumption

- Each inner node has two children
- All leaves have the same depth $2k$

$\Rightarrow$ A bit-string of length $n = 4^k$ encodes the input completely



A Simple Deterministic Algorithm

- Compute all nodes bottom up
- Running time on layer ℓ: 2^ℓ

$$\sum_{\ell=0}^{2k} 2^\ell = 2^{2k+1} - 1 = \Theta(4^k) = \Theta(n)$$

Can we do better? **NO!**

Theorem: Let A be any deterministic AND/OR-tree-algorithm. For $k \geq 1$ there exists an input $x_1, \ldots, x_{4^k}$ s.t. A visits all 4^k leaves and the output is the value of the last one visited.

Deterministic Evaluation

Simplifying Assumption

- Each inner node has two children
- All leaves have the same depth $2k$

$\Rightarrow$ A bit-string of length $n = 4^k$ encodes the input completely

A Simple Deterministic Algorithm

- Compute all nodes bottom up
- Running time on layer ℓ: 2^ℓ

$$\sum_{\ell=0}^{2k} 2^\ell = 2^{2k+1} - 1 = \Theta(4^k) = \Theta(n)$$

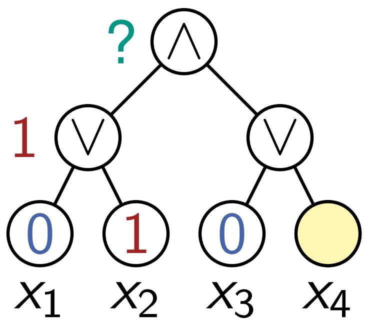
Can we do better? **NO!**

Proof via Induction

- Idea: We are an adversary who knows A and constructs an input (...on the fly, while the algorithm is running. Since A is deterministic this does not make a difference.)

Base: $k = 1$

- A visits ≥ 1 leaf: w.l.o.g. $A \to x_1$
- Set $x_1 := 0$ (value of parent and root *not* determined, yet)
- A needs to visit another leaf
- Case 1: $A \to x_2$
  - $x_1 := 1$ (value of parent determined, but not of root)
  - w.l.o.g. $A \to x_3$
  - $x_3 := 0$ (value of parent and root *not* determined, yet)



$\Rightarrow A \to x_4$

$\Rightarrow$ output is x_4 ✓

Theorem: Let A be any deterministic AND/OR-tree-algorithm. For $k \geq 1$ there exists an input $x_1, \ldots, x_{4^k}$ s.t. A visits all 4^k leaves and the output is the value of the last one visited.

Deterministic Evaluation

Simplifying Assumption

- Each inner node has two children
- All leaves have the same depth $2k$

  $\Rightarrow$ A bit-string of length $n = 4^k$ encodes the input completely

A Simple Deterministic Algorithm

- Compute all nodes bottom up
- Running time on layer ℓ: 2^ℓ

$$\sum_{\ell=0}^{2k} 2^\ell = 2^{2k+1} - 1 = \Theta(4^k) = \Theta(n)$$

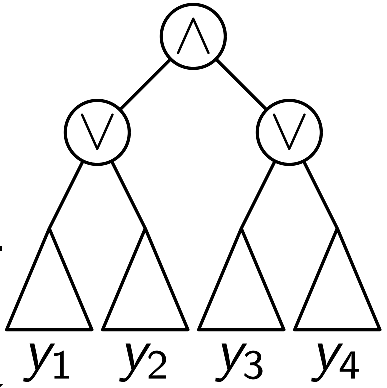
Can we do better? **NO!**

Proof via Induction

- Idea: We are an adversary who knows A and constructs an input (...on the fly, while the algorithm is running. Since A is deterministic this does not make a difference.)

Step: $k - 1 \to k$

- Consider tree of depth $2k$ as a tree of depth 2 with trees $y_1, \ldots, y_4$ (of depth $2(k-1)$) as "leaves"
- Analogous to the base, we can enforce that A needs to look at all y_i
- By induction, we can force A to look at all leaves in each y_i



$\Rightarrow$ A looks at all leaves ✓

Theorem: Let A be any deterministic AND/OR-tree-algorithm. For $k \geq 1$ there exists an input $x_1, \ldots, x_{4^k}$ s.t. A visits all 4^k leaves and the output is the value of the last one visited.

Randomized Evaluation

Karlsruhe Institute of Technology

Idea

- We can evaluate an $\wedge$-node to 0 if we find *one* 0-child
- We can evaluate an $\vee$-node to 1 if we find *one* 1-child

while ignoring the other child!

Algorithm

evalAndNode(v)

```
if v is leaf then
    return value(v)
c := uniformSample(v.children)
if evalOrNode(c) = 0 then
    return 0
c' := the other child
return evalOrNode(c')
```

Here each of the two children is selected with equal probability $1/2$.

evalOrNode(v)

$\vee$-nodes are not leaves in our setting

```
c := uniformSample(v.children)
if evalAndNode(c) = 1 then
    return 1
c' := the other child
return evalAndNode(c')
```

- Execute as **evalAndNode**(r) for root-node r

How long does that take?

Randomized Evaluation – Running Time

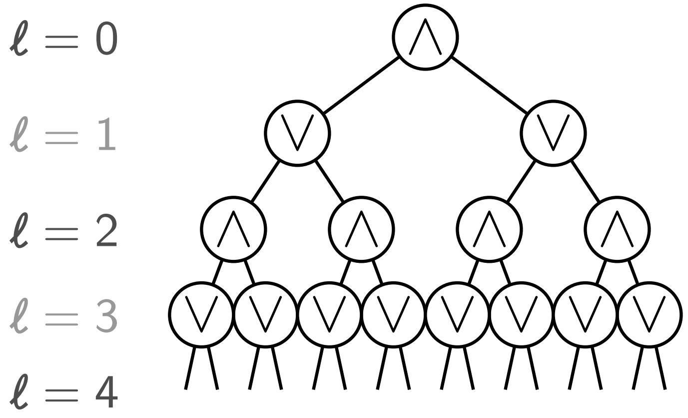
- Depends on how *lucky* we are, i.e., how often we can avoid checking the other child
- The running time is a *random variable*, we cannot deduce a specific value in advance

> **Theorem**: On *every* input $x_1, \dots x_{4^k}$ the **Randomized Evaluation** algorithm (RE) has an *expected running time* of $O(n^{\log_4(3)})$. $\approx O(n^{0.792\dots})$ is sublinear!

Proof via Induction (that the number X of visited leaves at depth $2k$ is $\leq 3^k = 3^{\log_4(n)} = n^{\log_4(3)}$ in expectation)

- Expected number of nodes evaluated on *even* layer $\ell = 2i$ is at most 3^i
- Expected number of nodes evaluated on *odd* layer ℓ is at most that of the layer beneath
- Expected number of total evaluated nodes is at most

$$\underbrace{\overbrace{3^0}^{\ell=0}}_{i=0} + \overbrace{3^1}^{\ell=1} + \underbrace{\overbrace{3^1}^{\ell=2}}_{i=1} + \overbrace{3^2}^{\ell=3} + \underbrace{\overbrace{3^2}^{\ell=4}}_{i=2} + \dots + \underbrace{\overbrace{3^k}^{\ell=2k}}_{i=k} \leq \sum_{i=0}^{k} 2 \cdot 3^i = \Theta(3^k)$$



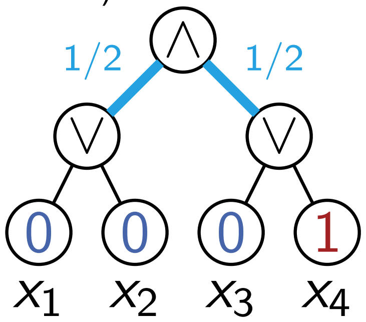
Randomized Evaluation – Running Time

- Depends on how *lucky* we are, i.e., how often we can avoid checking the other child
- The running time is a *random variable*, we cannot deduce a specific value in advance

Theorem: On *every* input $x_1, \dots x_{4^k}$ the **Randomized Evaluation** algorithm (RE) has an *expected running time* of $O(n^{\log_4(3)})$. $\approx O(n^{0.792\dots})$ is sublinear!

Proof via Induction (that the number X of visited leaves at depth $2k$ is $\leq 3^k = 3^{\log_4(n)} = n^{\log_4(3)}$ in expectation)

Base: $k = 1$



- Case analysis over all bit-strings x_1, x_2, x_3, x_4, example 0001
- Let X_L be number of leaves visited when going left first
  - Independent of leaf choice, need to look at other too: $X_L = 2$
  - When left $\vee$-node is checked, root value is determined
- Let X_R be number of leaves visited when going right first
  - $\Pr[\text{RE} \to x_3] = 1/2 \longrightarrow$ visit $x_4 \longrightarrow X_R = 2$
  - $\Pr[\text{RE} \to x_4] = 1/2 \longrightarrow$ do *not* visit $x_3 \longrightarrow X_R = 1$

$\mathbb{E}[X_L] = 2$

$\mathbb{E}[X_R] = 2 + \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1 = \frac{7}{2}$

- First left/right with prob $1/2$

$\mathbb{E}[X] = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot \frac{7}{2} = \frac{11}{4} \leq 3$ ✓

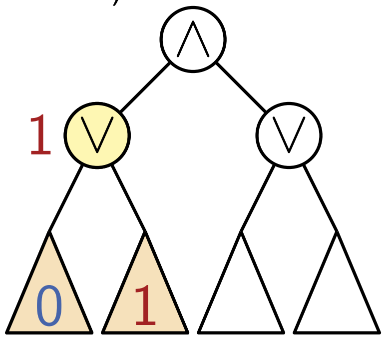
Randomized Evaluation – Running Time

- Depends on how *lucky* we are, i.e., how often we can avoid checking the other child
- The running time is a *random variable*, we cannot deduce a specific value in advance

Theorem: On *every* input $x_1, \ldots x_{4^k}$ the **Randomized Evaluation** algorithm (RE) has an *expected running time* of $O(n^{\log_4(3)})$. $\approx O(n^{0.792\ldots})$ is sublinear!

Proof via Induction (that the number X of visited leaves at depth $2k$ is $\leq 3^k = 3^{\log_4(n)} = n^{\log_4(3)}$ in expectation)

Step: $k - 1 \to k$



- Let Y be *trees* visited in $\vee$-node
- $\vee$-Case 0: node evaluates to 0 $\longrightarrow$ $\mathbb{E}[Y] = 2$
  - both sub-trees evaluate to 0 $\longrightarrow$ $Y = 2$
- $\vee$-Case 1: node evaluates to 1 $\longrightarrow$ $\mathbb{E}[Y] = p \cdot 1 + (1 - p) \cdot 2 = 2 - p \leq \frac{3}{2}$
  - at least one sub-tree evaluates to 1
  - with prob $p \geq 1/2$ (only!) this tree is visited first $\longrightarrow$ $Y = 1$
  - with prob $1 - p \leq 1/2$ both sub-trees are visited $\longrightarrow$ $Y = 2$

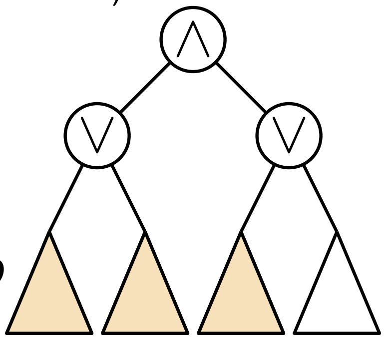
Randomized Evaluation – Running Time

- Depends on how *lucky* we are, i.e., how often we can avoid checking the other child
- The running time is a *random variable*, we cannot deduce a specific value in advance

Theorem: On *every* input $x_1, \ldots x_{4^k}$ the **Randomized Evaluation** algorithm (RE) has an *expected running time* of $O(n^{\log_4(3)})$. $\approx O(n^{0.792\ldots})$ is sublinear!

Proof via Induction (that the number X of visited leaves at depth $2k$ is $\leq 3^k = 3^{\log_4(n)} = n^{\log_4(3)}$ in expectation)

Step: $k - 1 \to k$



- Let Y be *trees* visited in $\vee$-node → Case 0: $\mathbb{E}[Y] = 2$ Case 1: $\mathbb{E}[Y] \leq \frac{3}{2}$
- Let Z be trees visited in $\wedge$-node
- $\wedge$-Case 0: node evaluates to 0 → $\mathbb{E}[Z] = p \cdot 2 + (1-p) \cdot (2 + \frac{3}{2}) = \frac{7}{2} - \frac{3}{2}p$ $\leq \frac{11}{4} \leq 3$
  - at least one $\vee$-node evaluates to 0
  - with prob $p \geq 1/2$ (only!) this node is visited first
  - with prob $1 - p \leq 1/2$ both $\vee$-nodes are visited
- $\wedge$-Case 1: node evaluates to 1 → $\mathbb{E}[Z] = 2 \cdot \frac{3}{2} = 3$
  - both $\vee$-nodes evaluate to 1

- Both cases: visit ≤ 3 trees in exp.
- Induction: exp. leaves per tree $\leq 3^{k-1}$

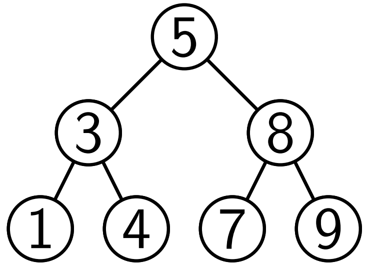
$\mathbb{E}[X] \leq 3 \cdot 3^{k-1} = 3^k$ ✓

Power of Randomness: Average-Case Analysis

Karlsruhe Institute of Technology

Binary Search Trees

- Goal: in a sequence of elements, quickly determine whether a given element is contained
- Example: $(1, 3, 4, 5, 7, 8, 9)$ Find: 4
- Idea: elements in left sub-tree are smaller, elements in right sub-tree are larger



Query

- Element equal to node? O.w. recurse in left/right child when element is smaller/larger
- Running time: linear in the depth of the tree

Maintenance

- Setting: elements appended over time, but never deleted
- How can we maintain the search-tree property as new elements arrive?

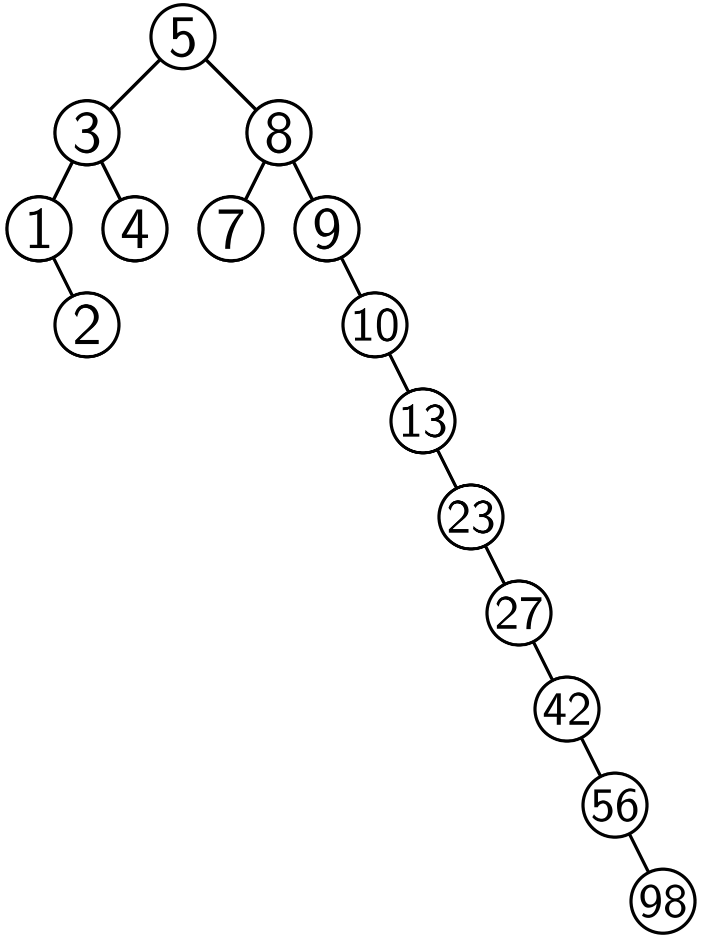
  Red-Black-Trees (a, b)*-Trees* *AVL-Trees*
- Complicated mechanisms that update the tree structure after an insertion
- Ensure that the depth is logarithmic in the number of nodes

Is all that necessary?

Keep it Simple

Simple Insert Strategy

- Place a new element where it belongs. ✓
- Example: Insert 2 , 10 , 13 , 23, 27, 42, 56, 98



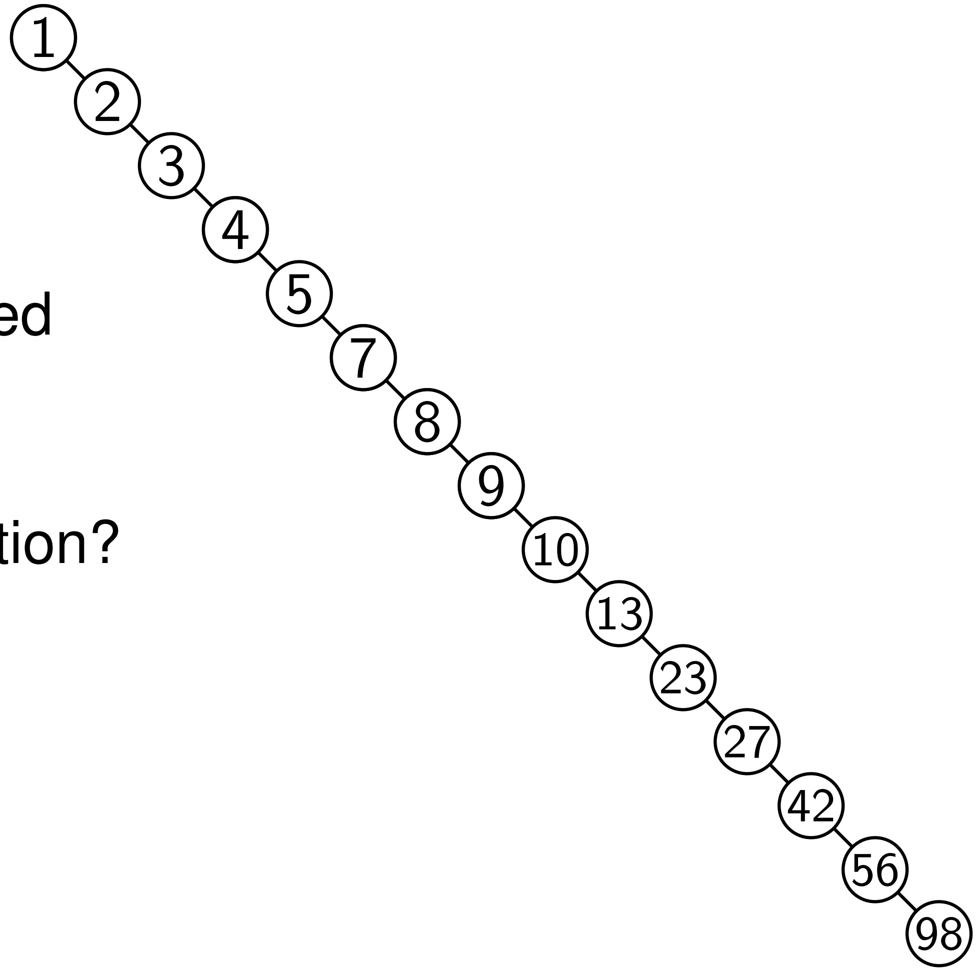
Keep it Simple

Simple Insert Strategy

- Place a new element where it belongs. ✓
- Example: Insert 2 , 10 , 13 , 23, 27, 42, 56, 98

Problem ?

- If elements come in sorted order, tree is unbalanced
- Worst case: linear running time for single query
- Is that *actually* a problem?
- Is it *likely* that this happens in a real-world application?
- Only 1 sequence yields this tree



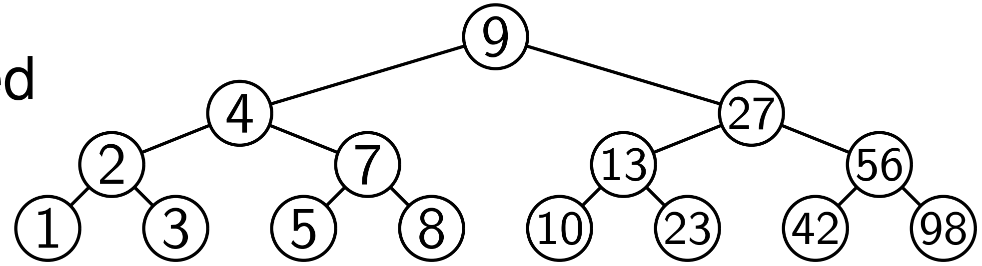
Keep it Simple

Simple Insert Strategy

- Place a new element where it belongs. ✓
- Example: Insert 2 , 10 , 13 , 23, 27, 42, 56, 98

Problem ?

- If elements come in sorted order, tree is unbalanced
- Worst case: linear running time for single query
- Is that *actually* a problem?
- Is it *likely* that this happens in a real-world application?
- Only 1 sequence yields this tree , 21964800 sequences yield a perfectly balanced tree



https://oeis.org/A056971

Average-Case Analysis

- Model real world via probability distribution over possible inputs, which is
  - simple (so that we can analyze it) ✓
  - realistic (so that we can make useful predictions about the real world) Not so clear...

In the following: uniform random permutation of the numbers

Simple Insert Strategy: Analysis

Theorem: Let S be a permutation of $M = \{1, 2, \ldots, n\}$ chosen uniformly at random. Then, the expected depth of a binary search tree with the Simple Insert Strategy is $O(\log(n))$.

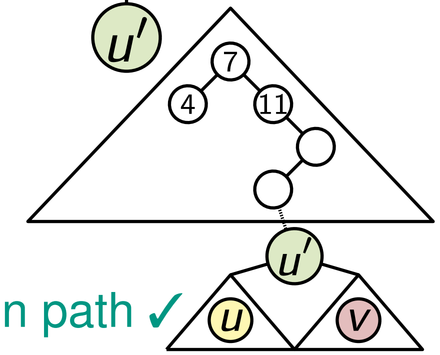
- w.l.o.g. we can assume the elements to be $1, \ldots, n$, as we are only interested in the order

Observation: Let T be a binary search tree with the Simple Insert Strategy and let $v \in T$ be an element. Then the path from v to the root contains a node $u < v$, if and only if u is the first among $M_{u,v} = \{u, \ldots, v\}$ in S.

$M = \{1, 2, 3, 4, \ldots, u, u+1, \ldots, v, \ldots, n\}$

$S = (7, 11, 4, \ldots, u, \ldots, v, \ldots, u+1, \ldots, 1)$

- Before an element in $M_{u,v}$ is added, all elements are smaller/larger
- All paths that would lead to $x \in M_{u,v}$ are identical
- Let $u' \in M_{u,v}$ be the *first* element from $M_{u,v}$ to appear in S
- From then on, u' is on the path that would lead to v
- Case 1: $u' = u$: u is on path ✓
- Case 2: $u' \neq u$: ($u < u'$) & u is in left sub-tree of u' but v is in right u not on path ✓



Simple Insert Strategy: Analysis

Theorem: Let S be a permutation of $M = \{1, 2, \dots, n\}$ chosen uniformly at random. Then, the expected depth of a binary search tree with the Simple Insert Strategy is $O(\log(n))$.

- w.l.o.g. we can assume the elements to be $1, \dots, n$, as we are only interested in the order

Observation: Let T be a binary search tree with the Simple Insert Strategy and let $v \in T$ be an element. Then the path from v to the root contains a node $u < v$, if and only if u is the first among $M_{u,v} = \{u, \dots, v\}$ in S. $u > v$

$M_{v,u} = \{v, \dots, u\}$ (for symmetry reasons)

- Let $S_{u,v}$ be the subsequence of S containing the elements in $M_{u,v}$
- Then $S_{u,v}$ is a uniform random permutation of $M_{u,v}$
- The probability that u is first in $S_{u,v}$ is
  $\Pr[\text{“}u \text{ first in } S_{u,v}\text{”}] = 1/|M_{u,v}| = 1/(v - u + 1)$
- Analogous for $S_{v,u}$
  $\Pr[\text{“}u \text{ first in } S_{v,u}\text{”}] = 1/(u - v + 1)$

$$M_{u,v} = \{u, u+1, u+2, v\}$$
$$S = (\dots, u, \dots, u+2, \dots, v, \dots, u+1, \dots)$$
$$S_{u,v} = (u, u+2, v, u+1)$$

Simple Insert Strategy: Analysis

Theorem: Let S be a permutation of $M = \{1, 2, \ldots, n\}$ chosen uniformly at random. Then, the expected depth of a binary search tree with the Simple Insert Strategy is $O(\log(n))$.

- w.l.o.g. we can assume the elements to be $1, \ldots, n$, as we are only interested in the order

Observation: Let T be a binary search tree with the Simple Insert Strategy and let $v \in T$ be an element. Then the path from v to the root contains a node $u < v$, if and only if u is the first among $M_{u,v} = \{u, \ldots, v\}$ in S. $u > v$

- Let X_u be the indicator random variable with

$$X_u = \begin{cases} 1, & \text{if } u \text{ is on the path to } v \\ 0, & \text{otherwise} \end{cases}$$

$$\boxed{\mathbb{E}[X_u] = \Pr[X_u = 1]}$$

$$\Pr[\text{“}u \text{ on path to } v\text{”}] = \begin{cases} 1/(v - u + 1), & \text{if } u < v \\ 1/(u - v + 1), & \text{if } v < u \end{cases}$$

- Then the length of the path to v is $\ell = \sum_{u \in \{1,\ldots,n\} \setminus \{v\}} X_u$

Harmonic number:
$H_n = \sum_{i=1}^{n} \frac{1}{i} \in O(\log(n))$

$$\mathbb{E}[\ell] = \mathbb{E}\left[\sum_{u=1}^{v-1} X_u + \sum_{u=v+1}^{n} X_u\right] = \sum_{u=1}^{v-1} \frac{1}{v-u+1} + \sum_{u=v+1}^{n} \frac{1}{u-v+1} = \underbrace{\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{v}}_{H_v - 1} + \underbrace{\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n-v+1}}_{H_{n-v+1} - 1} \in O(\log(n)) ✓$$

Conclusion

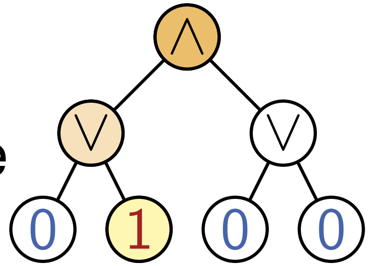
Organizational

- Homepage: scale.iti.kit.edu/teaching/2023ws/randalg
- A place for questions will be linked on the website

Randomized Algorithms

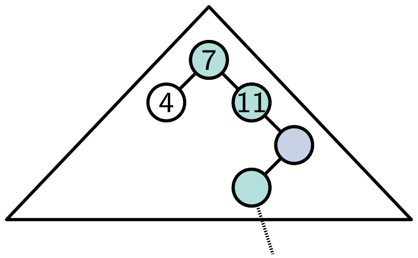
- Often simpler/faster than deterministic ones (sometimes the only possible way)
- At the cost of certainty (may be slow, may be wrong)

  Quicksort (expected $O(n \log(n))$ but $O(n^2)$ worst case) Next week!
- Example: AND/OR-Trees, expected running time sublinear in the input size



Average-Case Analysis

- Model real world using probability distributions over inputs
- If worst case is unlikely, expect good running times
- Example: Binary search-trees with simple insert strategy have same expected depth as complicated deterministic data structures



Karlsruhe Institute of Technology

Probability & Computing

Probability Amplification

The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma \colon P \times P \mapsto \mathbb{R}_+$

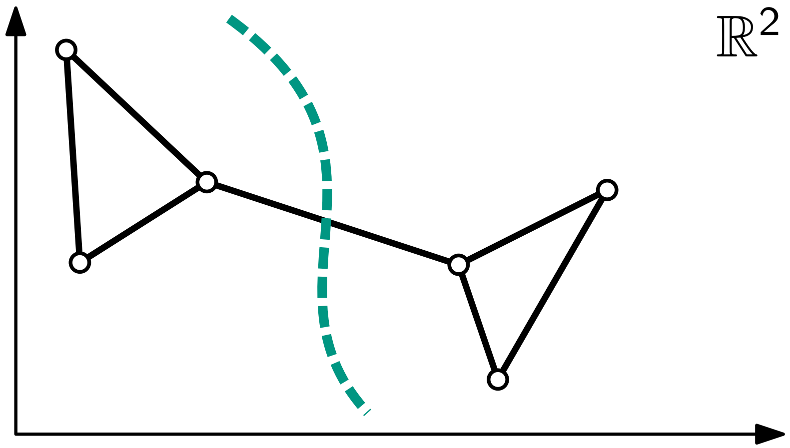
Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity

Applications: Compression, medical diagnosis, etc.

Approach: Model as graph

- Each point is a node
- Edges between all node pairs, with the weight given by the similarity of the two nodes
- Find *cut-set* (edges to remove) of minimal weight such that the graph decomposes into k components.



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

Today

$k = 2$ and $\sigma \colon P \times P \mapsto \{0, 1\}$

The Edge-Connectivity Problem

Cuts

- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$. (in general one can consider more than two parts)
- *Cut-set*: set of edges with one endpoint in V_1 and the other in V_2
- *Weight*: size of the cut-set (or sum of weights in a weighted graph)

Excursion: Cuts with Terminals

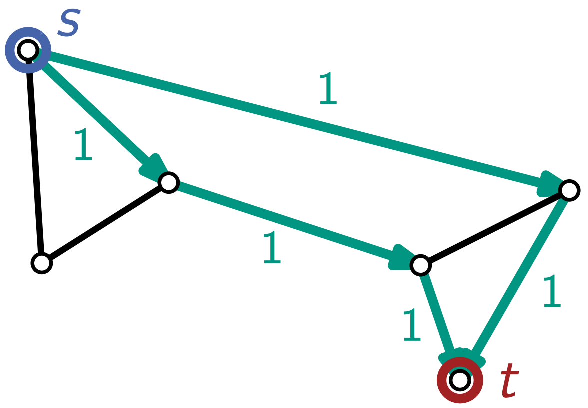
- each part contains exactly one of a specified vertex set

k-Edge-Connectivity

- k*-edge-connected*: a minimum cut has weight at least k (we cannot disconnect the graph by removing less than k edges)

Edge-Connectivity

- max. k such that G is k-edge-connected (exactly the weight of a min-cut)



Excursion: Flows

- given source s and target t
- assign *flow* to edges s.t.
  - in-flow = out-flow for all vertices (not s and t)
  - flow of an edge bounded by edge-capacity (here: ≤ 1)
  - flow in t is maximized

Thm. Max-Flow = Min-Cut.

Deterministic Algorithms for Edge-Connectivity

Flow-based

"Max flows in O(nm) time, or better", Orlin, STOC'13

- Compute max-flow between all vertex pairs $\rightarrow O(n^2 \cdot \underbrace{T_{\text{max-flow}}}^{O(nm)}) \subseteq O(n^3 m)$
- Compute max-flow between v and all others $\rightarrow O(n \cdot T_{\text{max-flow}}) \subseteq O(n^2 m) \rightarrow \Omega(n^3)$

(if a cut of size k exists, it has to cut v from some vertex)

Matroid-based

"A Matroid Approach to Finding Edge Connectivity and Packing Arborescences", Gabow, JCSS, 1995

- Involved technique based on the fact that min-cut $=$ max. number of dijsoint, directed spanning trees $\rightarrow O(m + k^2 n \log(n/k))$
- Good if k is small but still $\Omega(n^3)$ in the worst case

Contraction-based

"A simple min-cut algorithm", Stoer & Wagner, JACM, 1997

- Iteratively pick two vertices (in a smart way) and compare the min-cuts where they are / are not in the same part $\rightarrow O(mn + n^2 \log(n)) \rightarrow \Omega(n^3)$

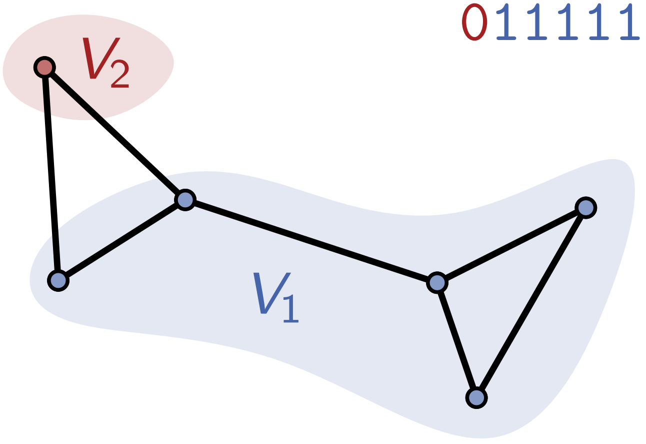
Enter: The Power of Randomness!

A Simple(?) Randomized Algorithm

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2)/2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Simple(?) Randomized Cut

- Simple idea: choose a cut at random among all possible cuts and return it.

What do we mean?
What distribution?

- Uniform distribution: We do not want to potentially favor non-minimum cuts
- Problem: How do we choose a cut *uniformly* at random?
  - Represent cut using bit-string
  - How can we choose a unfiorm random bit-string *while avoiding* `11...1` and `00...0`?

n random bits? $\rightarrow$ does not avoid `11...1` and `00...0`

random number from $\{1, \ldots, 2^n - 2\}$? $\rightarrow$ exponential in input size

rejection sampling? running time not deterministic (though probably what you'd do in practice)

Excursion: Uniform Non-Identical Bit Strings

[For educational purposes only!]

- **Goal**: Choose uniformly at random from the length n bit-strings that are not 0^n or 1^n
- Number of valid bit-strings:

$$2^n - 2 = \left(\sum_{k=0}^{n}\binom{n}{k}\right) - 2 = \sum_{k=1}^{n-1}\binom{n}{k}$$

$$2^n = \sum_{k=0}^{n}\binom{n}{k}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

Assumptions: We can sample ...
- uniformly from $\{0, ..., O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

- 2-step process: choose k & choose k 1s in n bits

unibs(n)

$b := 00\ldots0$ // n zeros
$k :=$ **rand**$(\{1, \ldots, n-1\})$ // number of 1s
$P :=$ **randSet**$(\{1, \ldots, n\}, k)$ // positions of 1s
$b[P] = 1$ // set 1s in b
return b

How to sample k?
- uniform?

$$\left.\begin{aligned}\Pr[1000] &= 1/3 \cdot 1/4 = 1/12\\ \Pr[1100] &= 1/3 \cdot 1/6 = 1/18\end{aligned}\right\} \neq 1/\underbrace{14}_{2^n-2}$$

- choose k with prob $\binom{n}{k}/(2^n - 2)$
- Reduce to uniform using *Inverse Transform Sampling*

How to sample P?

$n = 4$

$$\left.\begin{matrix}1000\\0100\\0010\\0001\end{matrix}\right\}k=1 \quad \left.\begin{matrix}1100\\1010\\1001\\0110\\0101\\0011\end{matrix}\right\}k=2 \quad \left.\begin{matrix}1110\\1101\\1011\\0111\end{matrix}\right\}k=3$$

Excursion-Excursion: Reservoir Sampling

[For educational purposes only!]

- **Goal**: Choose a set of size k uniformly at random from the n elements.
- **Idea**:
  - initialize **reservoir** with first k elements
  - replace reservoir elements at random

Assumptions: We can sample ...
- uniformly from $\{0, ..., O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

randSet$(\{1, \ldots, n\}, k)$

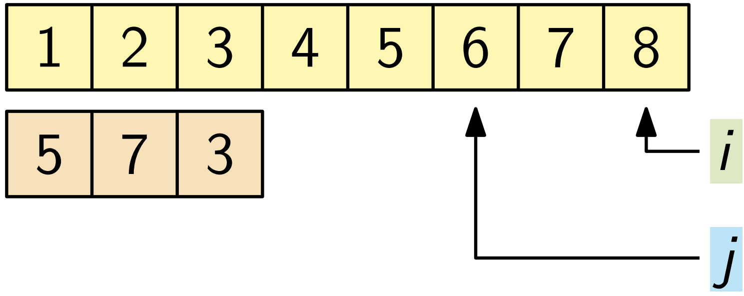
$r := [1, \ldots, k]$ // reservoir // $O(k)$

for i from $k+1$ to n **do** // $O(n-k)$

  $j :=$ **unif**$(\{1, \ldots, i\})$ // $O(1)$

  if $j \leq k$ **then** $r[j] = i$ // $O(1)$

return r



- **Running time**: $O(n)$

Excursion: Uniform Non-Homogeneous Bit Strings

[For educational purposes only!]

- **Goal**: Choose uniformly at random from the length n bit strings that are not 0^n or 1^n
- 2-step process:
  - choose k
  - choose k 1s in n bits

Assumptions: We can sample . . .
- uniformly from $\{0, ..., O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

unibs(n)

$b := 00...0$ // n zeros // $O(n)$

$k :=$ **rand**$(\{1, ..., n-1\})$ // number of 1s // $O(\log(n))$ via Inverse Transform Sampling

$P :=$ **randSet**$(\{1, ..., n\}, k)$ // positions of 1s // $O(n)$ via Reservoir Sampling

$b[P] = 1$ // set 1s in b // $O(k) \subseteq O(n)$

return b

Under our assumptions, we can sample a length n bit string that is not 0^n or 1^n uniformly at random in time $O(n)$.

Simple Randomized Cut

- Simple idea: choose a cut uniformly at random among all possible cuts and return it.
- Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability

- $2^{n-1} - 1$ cuts in a graph with n nodes
- How many min-cuts? $\rightarrow$ pessimistic assumption: 1

Observation: On a graph with n nodes, **Simple Randomized Cut** runs in $O(n)$ time and returns a minimum cut with probability at least $1/(2^{n-1} - 1)$.$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

  $\Pr[\text{“minimum found”}] \geq 1 - \left(1 - 1/(2^{n-1} - 1)\right)^t \geq 1 - e^{-t/(2^{n-1}-1)}$

  $1 + x \leq e^x$ for $x \in \mathbb{R}$

- For $t = 2^{n-1} - 1$ minimum found with constant probability $1 - 1/e \approx 0.63$
- For $t = (2^{n-1} - 1) \cdot \log(n)$ minimum found with high probability $1 - 1/n$

Probability Amplification

Karlsruhe Institute of Technology

Definition: A **Monte Carlo Algorithm** is a randomized algorithm that terminates deterministically and whose output is correct only with a certain probability $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

bias

majority

best

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

- After t (independent) runs return the ...

  $\Pr[\text{“success”}] \geq 1 - (1 - p)^t \geq 1 - e^{-pt}$ (for two-sided errors it's a bit more complicated)

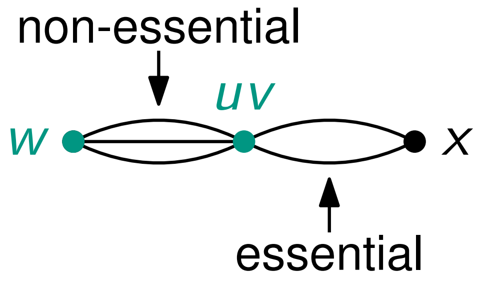
- Error probability decreases exponentially in t

For Simple Randomized Cut we had to pay with exponentially large running time ...

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut



Contraction Algorithm

- Motivation: distinguish $\overbrace{\textit{non-essential}}^{\text{not part of a min-cut}}$ as well as $\left.\textit{essential}\right\}$ edges part of a min-cut & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

  for $i = 1$ to $n-2$ **do** // $O(n)$

    $e :=$ **unif**(E_{i-1}) // $O(1)$

    $G_i = G_{i-1}$.**contract**(e) // $O(n)$

  return unique cut in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut in G_i is a cut in G_0.

- Consider min-cut with cut set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$$m = \frac{1}{2}\sum_{v \in V} \deg(v) \geq \frac{1}{2}\sum_{v \in V} k \geq \frac{1}{2} nk$$

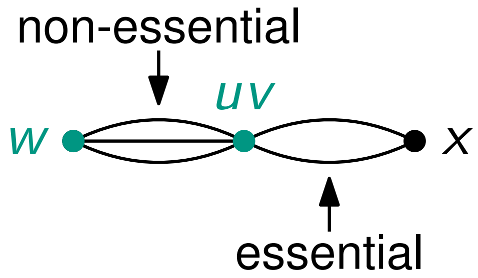
$$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m} \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$$

Karger's Algorithm

Karlsruhe Institute of Technology

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut



Contraction Algorithm

- Motivation: distinguish $\overbrace{\textit{non-essential}}^{\text{not part of a min-cut}}$ as well as $\underbrace{\textit{essential}}_{\text{part of a min-cut}}$ edges & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

```
for i = 1 to n − 2 do            // O(n)
    e := unif(E_{i−1})           // O(1)
    G_i = G_{i−1}.contract(e)    // O(n)
return unique cut in G_{n−2}
```

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut in G_i is a cut in G_0.

- Consider min-cut with cut set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ (holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right)\cdots\left(\frac{2}{\cancel{4}}\right)\left(\frac{1}{\cancel{3}}\right)$$

$$\geq \frac{2}{n\,(n-1)}$$

Karger's Algorithm Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $2/(n(n-1))$.

$$\Pr[\text{"min-cut found"}] \geq 1-\exp\left(-\frac{2t}{n(n-1)}\right) = 1-\frac{1}{n}$$

for $t = \frac{n(n-1)}{2}\log(n)$

Success probability $\geq p$

Number of repetitions t

Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2\log(n))$ Karger repetitions run in $O(n^4\log(n))$ total time and return a min-cut with high probability. Much better than exp. time Simple Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \dots, C_\ell$ be all the min-cuts in G and $\underbrace{\mathcal{E}^i_{n-2}}_{\text{disjoint, since the algorithm returns only one cut}}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n(n-1)}$

$$1 \geq \Pr\left[\bigcup_{i\in[\ell]} \mathcal{E}^i_{n-2}\right] = \sum_{i\in[\ell]} \Pr[\mathcal{E}^i_{n-2}] \geq \frac{2\cdot\ell}{n(n-1)}$$

Observation: A graph on n nodes contains at most $\frac{n(n-1)}{2}$ minimum cuts.

More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right) \cdots \left(\frac{n-i}{\cancel{n-i+2}}\right)\left(\frac{n-i-1}{\cancel{n-i+1}}\right)$$

$$= \frac{(n-i)(n-i-1)}{n(n-1)}$$

- With increasing number of steps the probability for a min-cut to survive decreases

KargerStein$(G_0 = (V_0, E_0))$
- **if** $|V_0| = 2$ **then** return unique cut
- **for** $i = 1$ to $t = |V_0| - \frac{|V_0|}{\sqrt{2}} - 1$ **do**
  - $e :=$ **unif**(E_{i-1})
  - $G_i = G_{i-1}$.**contract**(e)
- $C_1 :=$ **KargerStein**(G_t) // inde-
- $C_2 :=$ **KargerStein**(G_t) // pendent // runs
- **return** smaller of C_1, C_2

- Idea: stop when a min-cut is still likely to exist and recurse
- After $t = n - n/\sqrt{2} - 1$ steps we have

$$\Pr[\mathcal{E}_t] = \frac{(\cancel{n - n} + n/\sqrt{2} + 1\ )(\cancel{n - n} + n/\sqrt{2} + \cancel{1 - 1})}{n(n-1)} = \frac{n^2/2 + n/\sqrt{2}}{n(n-1)} = \frac{\cancel{n}(n/2 + 1/\sqrt{2})}{\cancel{n}(n-1)} = \frac{1}{2} \cdot \underbrace{\frac{n+\sqrt{2}}{n-1}}_{\geq 1} \geq \frac{1}{2}$$

- Probability that no mistake made after t steps still large

Karger-Stein: Running Time

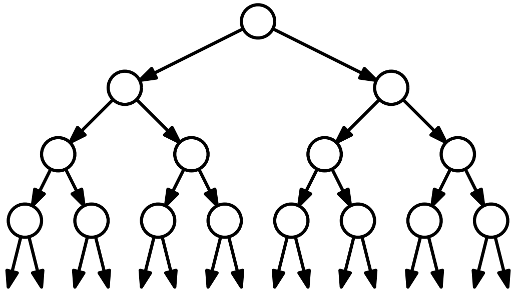
Recursion

- After $t = n - n/\sqrt{2} - 1$ steps the number of nodes is $n/\sqrt{2} + 1$

$$T(n) = 2T\left(\frac{n}{\sqrt{2}} + 1\right) + O(n^2)$$

Recursion tree

- Layers: $\log_{\sqrt{2}}(n)$
- Nodes on layer j: 2^j
- Time on layer j: $O\left(\left(\frac{n}{\sqrt{2}^j}\right)^2\right)$



$\textbf{KargerStein}(G_0 = (V_0, E_0))$

// $O(1)$	**if** $\lvert V_0 \rvert = 2$ **then** return unique cut
// $O(n)$	**for** $i = 1$ to $t = \lvert V_0 \rvert - \frac{\lvert V_0 \rvert}{\sqrt{2}} - 1$ **do**
// $O(1)$	$\quad e :=$ **unif**(E_{i-1})
// $O(n)$	$\quad G_i = G_{i-1}$.**contract**(e)

$C_1 :=$ **KargerStein**(G_t) // inde-
// pendent
$C_2 :=$ **KargerStein**(G_t) // runs
return smaller of C_1, C_2

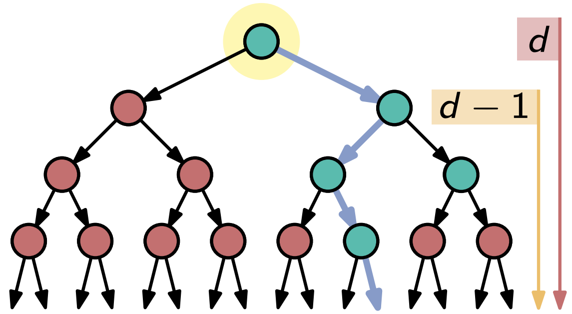
$$T(n) = \sum_{j=1}^{\log_{\sqrt{2}}(n)} 2^j \cdot O\left(\left(\frac{n}{\sqrt{2}^j}\right)^2\right) = O\left(n^2 \cdot \sum_{j=1}^{\log_{\sqrt{2}}(n)} \frac{2^j}{2^j}\right) = O(n^2 \log_{\sqrt{2}}(n)) = O(n^2 \log(n))$$

Karger-Stein: Success Probability

- After $t = n - n/\sqrt{2} - 1$ steps we have $\Pr[\mathcal{E}_t] \geq 1/2$ (t was chosen to achieve exactly that)

Recursion tree

- A node is a successful node if it still contains a min-cut of the original graph
- A path is a successful path if it contains only successful nodes
- $\mathcal{P}_d$: there exists a successful path of length d starting at the root



$\Pr[\mathcal{P}_0] \geq 1/2$

$\Pr[\mathcal{P}_d] = \Pr[\mathcal{P}_0] \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) \geq \frac{1}{2} \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) = \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$

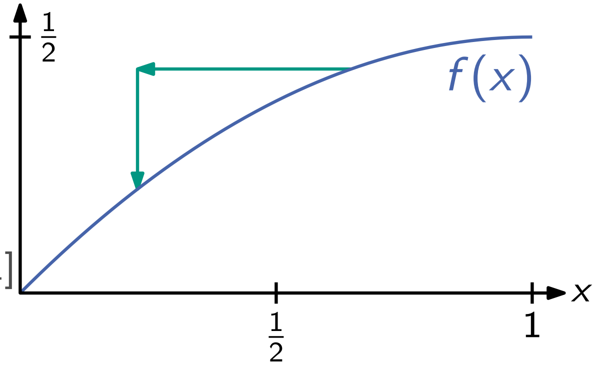
Claim $\Pr[\mathcal{P}_d] \geq \frac{1}{d+2}$ (proof via induction)

- Base case $d = 0$: $\Pr[\mathcal{P}_0] \geq 1/2$, Assumption: $\Pr[\mathcal{P}_{d-1}] \geq \frac{1}{d+1}$
- Step: $\Pr[\mathcal{P}_d] \geq \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$
  $\geq \frac{1}{d+1} - \frac{1}{2}\left(\frac{1}{d+1}\right)^2$

$f(x) = x - \frac{1}{2}x^2$

$f(x)\frac{\mathrm{d}}{\mathrm{d}x} = \underbrace{1 - x}_{\geq 0}$

for $x \in [0, 1]$



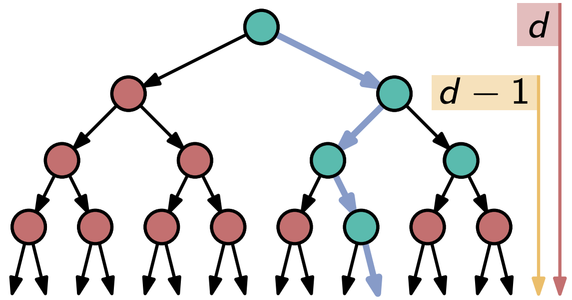
smaller x yields smaller $f(x)$

Karger-Stein: Success Probability

- After $t = n - n/\sqrt{2} - 1$ steps we have $\Pr[\mathcal{E}_t] \geq 1/2$ (t was chosen to achieve exactly that)

Recursion tree

- A node is a successful node if it still contains a min-cut of the original graph
- A path is a successful path if it contains only successful nodes
- $\mathcal{P}_d$: there exists a successful path of length d starting at the root



$\Pr[\mathcal{P}_0] \geq 1/2$

$\Pr[\mathcal{P}_d] = \Pr[\mathcal{P}_0] \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) \geq \frac{1}{2} \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) = \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$

Claim $\Pr[\mathcal{P}_d] \geq \frac{1}{d+2}$ (proof via induction)

- Base case $d = 0$: $\Pr[\mathcal{P}_0] \geq 1/2$, Assumption: $\Pr[\mathcal{P}_{d-1}] \geq \frac{1}{d+1}$
- Step: $\Pr[\mathcal{P}_d] \geq \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$

$$\geq \frac{1}{d+1} - \frac{1}{(2d+2)(d+1)}$$

$2d \geq d$ for $d \geq 0$

$$\geq \frac{1}{d+1} - \frac{1}{(d+2)(d+1)}$$

$$= \frac{\cancel{d+1}}{(\cancel{d+1})(d+2)} = \frac{1}{d+2}$$

- $\Pr[\text{“min-cut on layer } d\text{”}] \geq \frac{1}{d+2}$
- How many layers in the tree?
  $\rightarrow \log_{\sqrt{2}}(n)$
- $\Pr[\text{“min-cut returned”}] \geq \frac{1}{O(\log(n))}$

Karger-Stein Amplified

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Reminder: Karger $\rightarrow 1/O(n^2)$ in $O(n^2)$ time

Amplification

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right)$$

for $t = \log^2(n)$

Success probability $\geq p$

Number of repetitions t

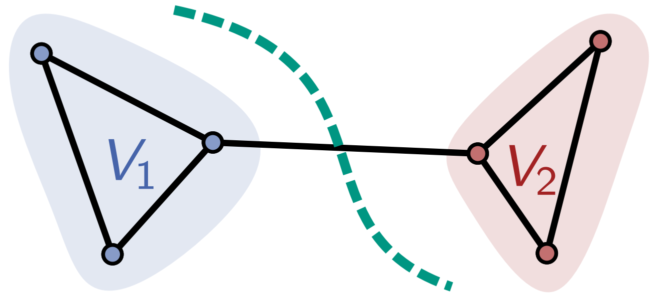
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(\log^2(n))$ repetitions of Karger-Stein run in $O(n^2 \log^3(n))$ total time and return a minimum cut with high probability.

- Compared to $O(n^4 \log(n))$ for Karger
- Compared to $\Omega(n^3)$ for deterministic approaches

Conclusion

Cuts
- Fundamental graph problem
- Many deterministic flow-based algorithms
- . . . with worst-case running times in $\Omega(n^3)$



Randomized Algorithms
- Simple randomized cut via reservoir sampling
- Karger's edge-contraction algorithm

Assumptions: We can sample . . .
- uniformly from $\{0, ..., O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

Probability Amplification
- Monte Carlo algorithms with and without biases
- Repetitions amplify success probability
- Karger-Stein: Amplify before failure probability gets large

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Outlook

"Minimum cuts in near-linear time", Karger, J.Acm. '00

Success w.h.p. in time $O(m \log^3(n))$

"Faster algorithms for edge connectivity via random 2-out contractions", Ghaffari & Nowicki & Thorup, SODA'20

Success w.h.p. in time $O(m \log(n))$ and $O(m + n \log^3(n))$

Probability & Computing

Coupling & Erdős-Rényi Random Graphs

Wheels of Fortune

The Problem

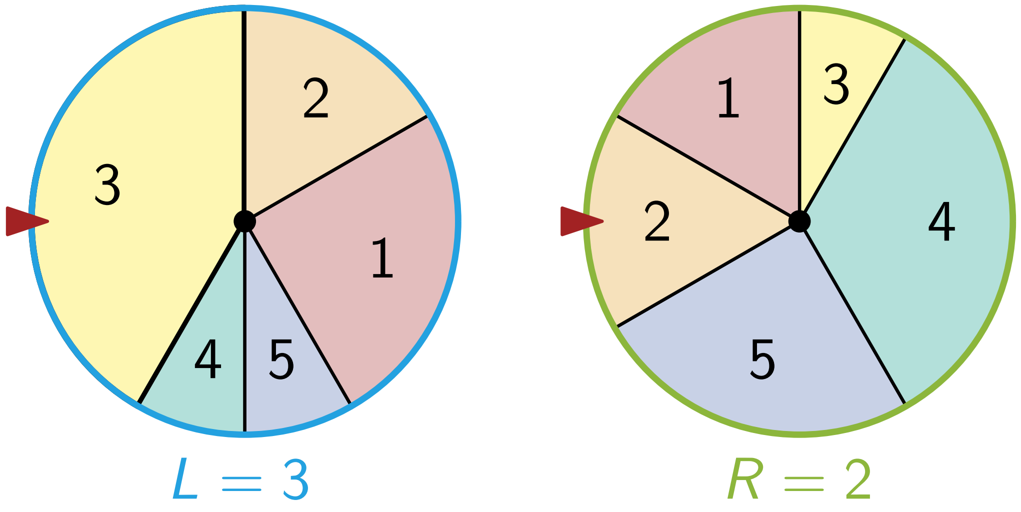
- Consider the two wheels of fortune
- The higher the value the larger the price
- Which do you spin? Why? Can we prove that?

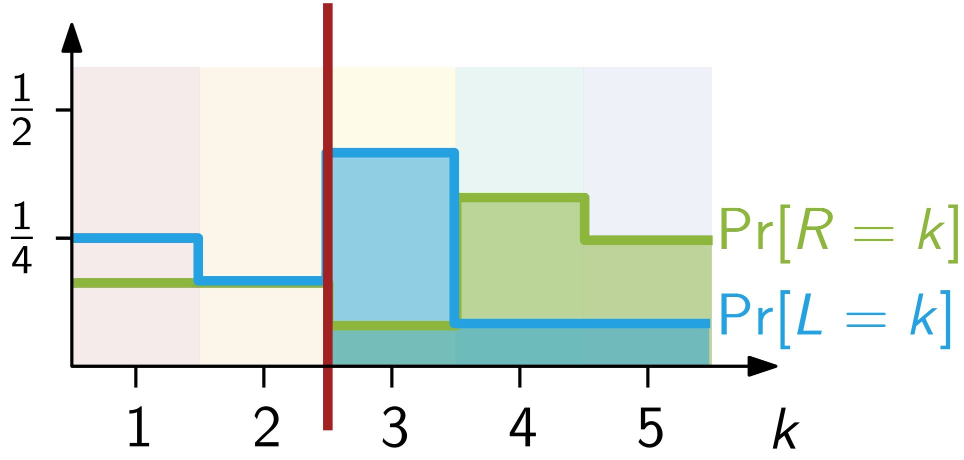
The Maths

- Let L be the value of the left wheel
- Let R be the value of the right wheel
- To show: For all values k: $\Pr[R \geq k] \geq \Pr[L \geq k]$

Proof

- For each k
  - Compute the sums of the probabilities
  - Compare
  - Tedious...





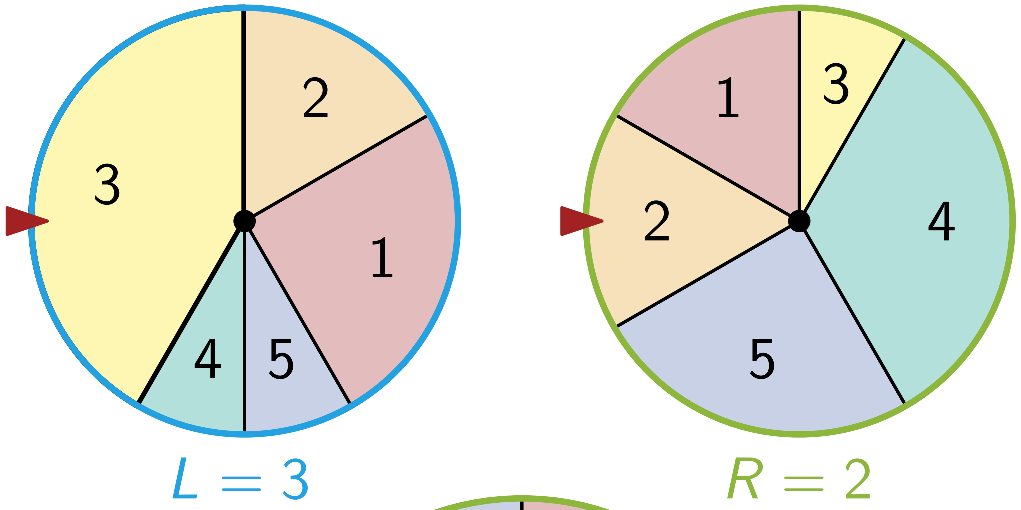
Wheels of Fortune

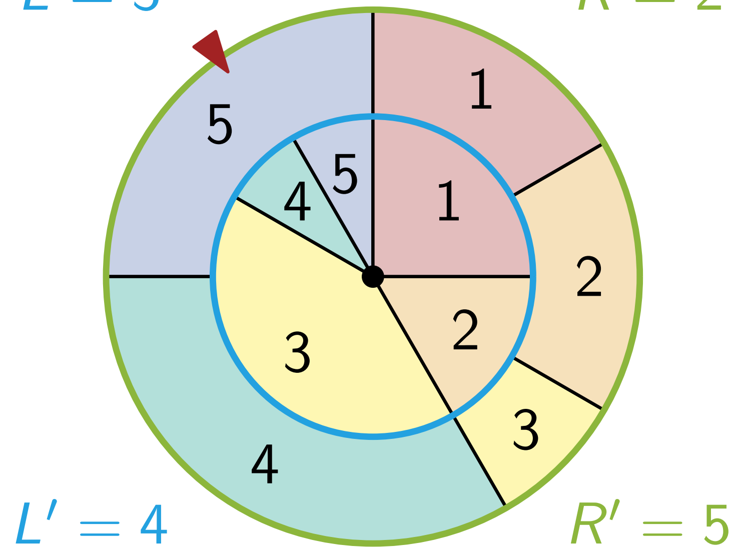
The Problem
- Consider the two wheels of fortune
- The higher the value the larger the price
- Which do you spin? Why? Can we prove that?

The Maths
- Let L be the value of the left wheel
- Let R be the value of the right wheel
- To show: For all values k: $\Pr[R \geq k] \geq \Pr[L \geq k]$ ✓

Proof: Frankenstein's Wheel of Fortune!
- Sort the wheels (does not change their distributions)
- Adjust sizes and glue together
- Spin as one wheel: L' inner number, R' outer number
- Note that $L \overset{d}{=} L'$ and $R \overset{d}{=} R'$ equal distributions
- But L' and R' are dependent and *always* $R' \geq L'$

  $\Rightarrow \Pr[R' \geq k] \geq \Pr[L' \geq k]$





What just happened?

Setup & Method

independent

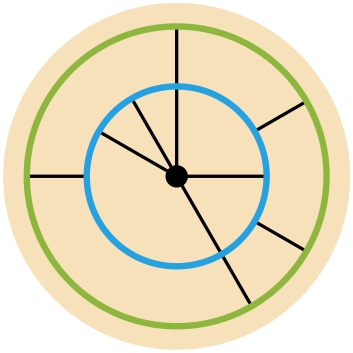
- Random variable L on the left wheel and R on right wheel

$\overset{d}{=}$ $\overset{d}{=}$

- Random variable L' on inner wheel and R' on outer wheel

dependent

The coupling defines how L' and R' are related



- Define a relation between random variables to make statements about one using the other Here: $\Pr[R' \geq k] \geq \Pr[L' \geq k] \Rightarrow \Pr[R \geq k] \geq \Pr[L \geq k]$

Definition: Let X_1, X_2 be random variables defined on probability spaces $(\Omega_1, \Sigma_1, \Pr_1)$ and $(\Omega_2, \Sigma_2, \Pr_2)$, respectively. A **coupling** of X_1 and X_2 is a pair of random variables (X_1', X_2') defined on a new probability space $(\Omega, \Sigma, \Pr)$ such that $X_1 \overset{d}{=} X_1'$ and $X_2 \overset{d}{=} X_2'$.

- X_1' and X_2' live in the *same* space
- Typically we define X_1' and X_2' to be dependent
- Typically we do not talk about the probability spaces explicitly

- Abstracting away technicalities, people just "couple" X_1 and X_2 "directly", without introducing X_1' and X_2'

Application: Biased Coins

The Problem

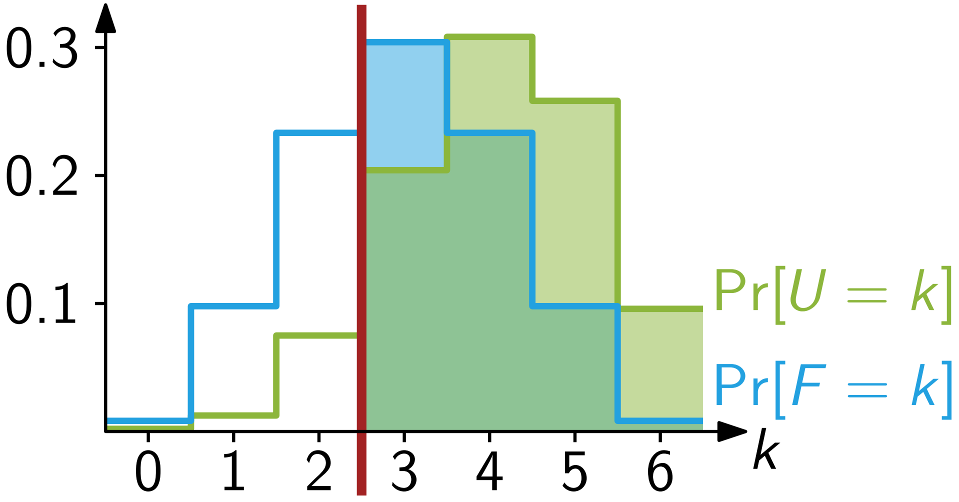
- We have a fair $\{0,1\}$-coin that yields 1 with probability $\frac{1}{2}$
- And an unfair $\{0,1\}$-coin that yields 1 with probability $\frac{2}{3}$
- Throw each coin n times, count the 1s, yielding F and U
- You pick a coin. You win if your coin gets more 1s than the other. Which do you pick?

$$F = \sum \; 0 \; 1 \; 0 \; 0 \; 0 \; 1 = 2$$

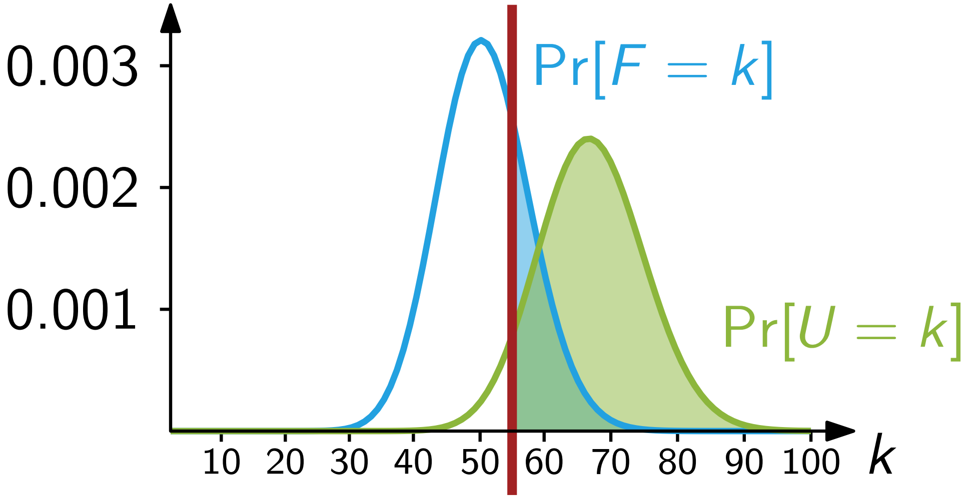
$$U = \sum \; 0 \; 0 \; 1 \; 1 \; 1 \; 1 = 4$$

Claim $\Pr[U \geq k] \geq \Pr[F \geq k]$

Proof *Compare sums for all* $k \leq 6$



And if $n = 100$? *so many sums...*



Application: Biased Coins

The Problem

- We have a fair $\{0,1\}$-coin that yields 1 with probability $\frac{1}{2}$
- And an unfair $\{0,1\}$-coin that yields 1 with probability $\frac{2}{3}$
- Throw each coin n times, count the 1s, yielding F and U
- You pick a coin. You win if your coin gets more 1s than the other. Which do you pick?

$$F = \sum 0\ 1\ 0\ 0\ 0\ 1 = 2$$

$$U = \sum 0\ 0\ 1\ 1\ 1\ 1 = 4$$

Claim $\Pr[U \geq k] \geq \Pr[F \geq k]$

Proof

- Let F_i be indicator for ith fair coin
- Let U_i be indicator for ith unfair coin
- Let W_i be the result of a fair die-roll
  - Define $F_i' = 1$ iff $W_i \leq 3 \Rightarrow F_i \overset{d}{=} F_i'$
  - Define $U_i' = 1$ iff $W_i \leq 4 \Rightarrow U_i \overset{d}{=} U_i'$
- F_i' and U_i' are dependent and *always* $U_i' \geq F_i'$

Coupling: Random variables X_1, X_2. Define random variables X_1', X_2' in a shared probability space such that $X_1 \overset{d}{=} X_1'$ and $X_2 \overset{d}{=} X_2'$.

W_i	⚀	⚁	⚂	⚃	⚄	⚅
Pr	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
F_i'	1	1	1	0	0	0
	$\wedge\vert$	$\wedge\vert$	$\wedge\vert$	$\wedge\vert$	$\wedge\vert$	$\wedge\vert$
U_i'	1	1	1	1	0	0

$$\begin{tikzcd} F_i \arrow[r, leftrightarrow, "\text{independent}"] \arrow[d, phantom, "\overset{d}{=}"] & U_i \arrow[d, phantom, "\overset{d}{=}"] \\ F_i' \arrow[r, leftrightarrow, "\text{dependent}"] & U_i' \end{tikzcd}$$

Application: Biased Coins

The Problem

- We have a fair $\{0,1\}$-coin that yields 1 with probability $\frac{1}{2}$
- And an unfair $\{0,1\}$-coin that yields 1 with probability $\frac{2}{3}$
- Throw each coin n times, count the 1s, yielding F and U
- You pick a coin. You win if your coin gets more 1s than the other. Which do you pick?

$$F = \sum 0\ 1\ 0\ 0\ 0\ 1 = 2$$

$$U = \sum 0\ 0\ 1\ 1\ 1\ 1 = 4$$

Claim $\Pr[U \geq k] \geq \Pr[F \geq k]$ ✓

Proof

- Let F_i be indicator for ith fair coin
- Let U_i be indicator for ith unfair coin
- Let W_i be the result of a fair die-roll
  - Define $F_i' = 1$ iff $W_i \leq 3 \Rightarrow F_i \stackrel{d}{=} F_i'$
  - Define $U_i' = 1$ iff $W_i \leq 4 \Rightarrow U_i \stackrel{d}{=} U_i'$
- F_i' and U_i' are dependent and *always* $U_i' \geq F_i'$
  $\Rightarrow U' \geq F' \Rightarrow \Pr[U' \geq k] \geq \Pr[F' \geq k]$

$F = \sum_{i=1}^{n} F_i$

$U = \sum_{i=1}^{n} U_i$

$F' = \sum_{i=1}^{n} F_i'$

$U' = \sum_{i=1}^{n} U_i'$

Coupling: Random variables X_1, X_2. Define random variables X_1', X_2' in a shared probability space such that $X_1 \stackrel{d}{=} X_1'$ and $X_2 \stackrel{d}{=} X_2'$.

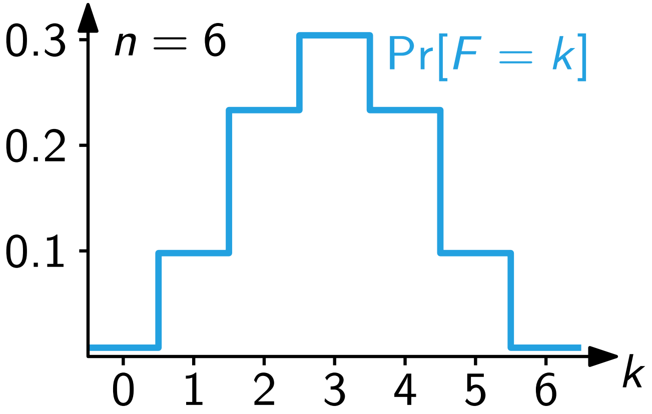
$$\begin{array}{ccc} F & \xleftrightarrow{\text{independent}} & U \\ \stackrel{d}{=} & & \stackrel{d}{=} \\ F' & \xleftrightarrow{\text{dependent}} & U' \end{array} \qquad \begin{array}{ccc} F_i & \xleftrightarrow{\text{independent}} & U_i \\ \stackrel{d}{=} & & \stackrel{d}{=} \\ F_i' & \xleftrightarrow{\text{dependent}} & U_i' \end{array}$$

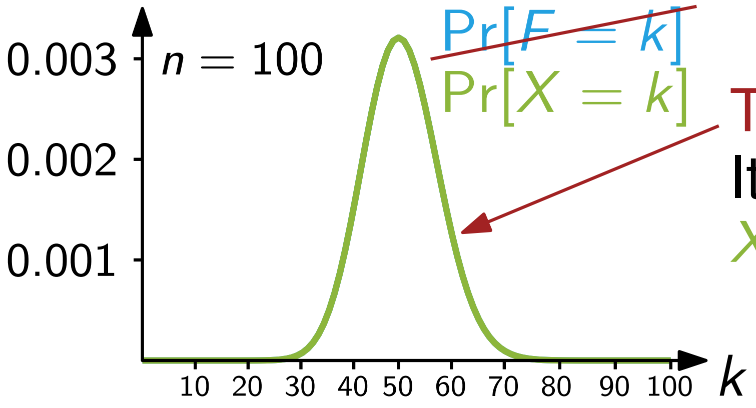
Observation: Independent rand. var. X_i, Y_i for $i \in [n]$ with couplings (X_i', Y_i') for $i \in [n]$.
Then, for any function f: $(f(X_1', \dots, X_n'), f(Y_1', \dots, Y_n'))$ is a coupling of $f(X_1, \dots, X_n)$ and $f(Y_1, \dots, Y_n)$.

The Binomial-Poisson-Approximation or “How I Lied To You”

Setup

- Fair $\{0,1\}$-coin X with $\Pr[X=1] = p = \frac{1}{2}$ — This is a Bernoulli rand. var. $X \sim \mathrm{Ber}(p)$
- Sum of n ind. coins $F = \sum_{i=1}^{n} X_i$, $X_i \sim \mathrm{Ber}(p)$ — This is a Binomial rand. var. $F \sim \mathrm{Bin}(n,p)$
  $\Pr[F=k] = \binom{n}{k} p^k (1-p)^{n-k}$
- ... which we have seen today already





This is not a binomial distribution!
It’s a Poisson distribution with $\lambda = 50$
$X \sim \mathrm{Pois}(\lambda)$: $\Pr[X=k] = \lambda^k e^{-\lambda}/k!$

- Why lie? It was easier to plot that way and I thought you wouldn’t notice...
- How dare I? As n increases, the two distributions are very close...

What does that mean?

Total Variation Distance

- A measure of distance between the distributions of random variables

(Disclaimer: In the following we use a very simplified notation that abstracts away a lot of details!)

Definition: Let X, Y be random variables taking values in a set S. The **total variation distance** of X and Y is $d_{TV}(X,Y) = \frac{1}{2}\sum_{x \in S} |\Pr[X=x] - \Pr[Y=x]|$.

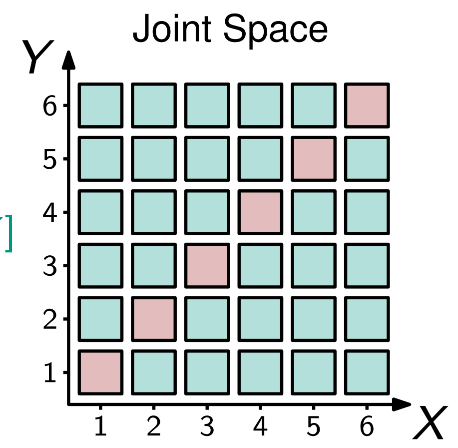
- Intuition: Sum over the differences in the probabilities
- Maybe a bit tedious to work with, simple bound:

Fréchet: $\Pr[A] - \Pr[B] \leq \Pr[A \wedge \bar{B}]$

$S_X = \{x \in S \mid \Pr[X=x] \geq Pr[Y=x]\}$ $\quad S_Y = S \setminus S_X$

$$
\begin{aligned}
\cancel{2}d_{TV}(X,Y) &= \sum_{x \in S} |\Pr[X=x] - \Pr[Y=x]| \\
&= \sum_{x \in S_X} \cancel{|}\Pr[X=x] - \Pr[Y=x]\cancel{|} + \sum_{x \in S_Y} \cancel{|}\Pr[X=x] - \Pr[Y=x]\cancel{|} \\
&= \sum_{x \in S_X} \Pr[X=x] - \Pr[Y=x] + \sum_{x \in S_Y} \Pr[Y=x] - \Pr[X=x] \\
&\leq \sum_{x \in S_X} \Pr[X=x \wedge Y \neq x] + \sum_{x \in S_Y} \Pr[Y=x \wedge X \neq x] \\
&\leq \sum_{x \in S} \Pr[X=x \wedge Y \neq x] + \sum_{x \in S} \Pr[Y=x \wedge X \neq x] \\
&= \Pr[X \neq Y] + \Pr[Y \neq X] = \cancel{2}\Pr[X \neq Y]
\end{aligned}
$$

$\Pr[X \neq Y]$



Lemma: $d_{TV}(X,Y) \leq \Pr[X \neq Y]$.

Total Variation Distance

- A measure of distance between the distributions of random variables

(Disclaimer: In the following we use a very simplified notation that abstracts away a lot of details!)

Definition: Let X, Y be random variables taking values in a set S. The **total variation distance** of X and Y is $d_{TV}(X,Y) = \frac{1}{2}\sum_{x\in S}\left|\Pr[X=x] - \Pr[Y=x]\right|$.

- Intuition: Sum over the differences in the probabilities
- Maybe a bit tedious to work with, simple bound:

Fréchet: $\Pr[A] - \Pr[B] \leq \Pr[A \wedge \bar{B}]$

Lemma: $d_{TV}(X,Y) \leq \Pr[X \neq Y]$.

- Note that d_{TV} is defined via the distributions of X and Y
- For any coupling (X', Y') of X, Y we have $X' \stackrel{d}{=} X$ and $Y' \stackrel{d}{=} Y$. Thus, $d_{TV}(X,Y) = d_{TV}(X',Y')$

Lemma (coupling inequality): Let X, Y be random variables. Then for any coupling (X', Y') of X and Y it holds that $d_{TV}(X,Y) \leq \Pr[X' \neq Y']$.

Lemma (triangle inequality): For rand. var. X, Y, Z: $d_{TV}(X,Z) \leq d_{TV}(X,Y) + d_{TV}(Y,Z)$.

The Binomial-Poisson-Approximation

- Ind. $X_i \sim \mathrm{Ber}(p)$ for $i \in [n]$ $\longrightarrow$ $X = \sum_{i=1}^n X_i$ $\longrightarrow$ $X \sim \mathrm{Bin}(n,p)$
- Ind. $Y_i \sim \mathrm{Pois}(\lambda)$ for $i \in [n]$, $\lambda = -\log(1-p)$ $\longrightarrow$ $Y = \sum_{i=1}^n Y_i$ $\longrightarrow$ $Y \sim \mathrm{Pois}(n\lambda)$

Lemma: $X \sim \mathrm{Bin}(n,p)$, $Y \sim \mathrm{Pois}(-n\log(1-p))$: $d_{TV}(X,Y) \le \frac{n}{2}\log(1-p)^2$.

$\Pr[Y_i = k] = e^{-\lambda}\lambda^k/k!$

$$\begin{tikzcd} X_i \arrow[r, leftrightarrow, "\text{independent}"] \arrow[d, phantom, "\overset{d}{=}\ \checkmark"] & Y_i \arrow[d, phantom, "\overset{d}{=}\ \checkmark"] \\ X_i' \arrow[r, leftrightarrow, "\text{dependent}"] & Y_i' \end{tikzcd}$$

Proof

- For each i we couple Y_i and X_i: $Y_i' = Y_i$, $X_i' = \min\{Y_i', 1\}$
- To show that this is a coupling, we need $X_i \overset{d}{=} X_i'$

$$\Pr[X_i' = 0] = \Pr[Y_i = 0] = e^{-\lambda} = e^{\log(1-p)} = 1 - p = \Pr[X_i = 0] \ \checkmark$$

$$\Pr[X_i' = 1] = \Pr[Y_i > 0] = 1 - \Pr[Y_i = 0] = 1 - \Pr[X_i = 0] = \Pr[X_i = 1] \ \checkmark$$

- $X' = \sum_{i=1}^n X_i'$, $Y' = \sum_{i=1}^n Y_i'$ $\Rightarrow$ (X', Y') coupling of (X, Y)

$$d_{TV}(X,Y) \le \Pr[X' \ne Y'] \underset{\text{union bound}}{\le} \sum_{i=1}^n \Pr[X_i' \ne Y_i'] = \sum_{i=1}^n \Pr[Y_i' \ge 2]$$

$$= \sum_{i=1}^n e^{-\lambda} \sum_{j \ge 2} \frac{\lambda^j}{j!} \le \sum_{i=1}^n \frac{\lambda^2}{2} \cancel{e^{-\lambda}} \underbrace{\cancel{\sum_{j\ge 0} \frac{\lambda^j}{j!}}}_{= e^{\lambda}} = \sum_{i=1}^n \frac{\lambda^2}{2} \ \checkmark$$

Function of Couplings
Couplings (X_i', Y_i') of (X_i, Y_i):
$(f(X_i'), f(Y_i'))$ coupling of $(f(X_i), f(Y_i))$.

Coupling Inequality
For any coupling (X', Y') of X, Y:
$d_{TV}(X,Y) \le \Pr[X' \ne Y']$.

Recap: Theory-Practice Gap

Karlsruhe Institute of Technology

Theory

- Many computational problems are assumed to be hard
- Looks like there are no algorithms that can solve these problems fast

$\underbrace{\text{SAT} \quad \text{Vertex Cover} \quad \text{Independent Set}}_{\text{NP-hard}}$

Practice

- Many computational problems can be solved extremely fast
  - “Modern SAT solvers can often handle problems with millions of clauses and hundreds of thousands of variables”

    “Propagation = Lazy Clause Generation”, Ohrimenko, Stuckey & Codish, CP, 2017
  - For many real-world graphs optimal vertex covers (containing up to millions of nodes) can be found in seconds

    “Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover”, Akiba & Iwata, TCS, 2016

Average-Case Analysis

- Acknowledge difference between theoretical worst-case instances and practical ones
- Represent real world using mathematical models and analyze those theoretically

Random Graph Models

A **graph model** describes a mechanism that can be used to generate a graph.

- Given a set of vertices, how are edges in the graph formed?
- The model consists of *rules* defining which vertices are adjacent
- In *random* graph models these rules involve randomness

Desirable Properties

- *Simplicity*: We cannot analyze a model that is too complicated
- *Realism*: We do not want to analyze a model that cannot be used to make predictions about the real world
- *Fast Generation*: We want to be able to generate many, large benchmark instances to ...
  - analyze structural and algorithmic properties empirically
  - generate hypotheses about asymptotic behavior

Let's start with a simple model!

Erdős–Rényi Random Graphs

Karlsruhe Institute of Technology

History

- Initially introduced by Edgar Gilbert in 1959

“Random Graphs”, Gilbert, Ann. Math. Statist., 1959

- A related version introduced by Paul Erdős and Alfréd Rényi in 1959

“On Random Graphs I”, Erdős & Rényi, Publ. Math. Debr., 1959

Definitions

Gilbert’s model, though often meant when talking about Erdős–Rényi graphs

$G(n, p)$

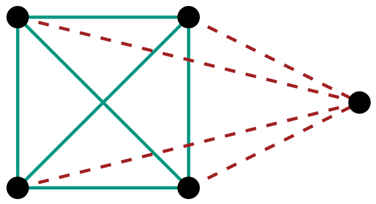
- Start with n nodes
- Independently connect any two with fixed probability p

$G(n, m)$

- Start with n nodes
- From the $\binom{n}{2}$ possible edges select m uniformly at random

---

- For $\tilde{p} = m/\binom{n}{2}$ the *expected* number of edges in $G(n, \tilde{p})$ matches m
- In $G(n, p)$ edges are independent, in $G(n, m)$ they are not
  - If a $G(5, 6)$ contains a 4-clique, there can be no edge incident to the 5th node



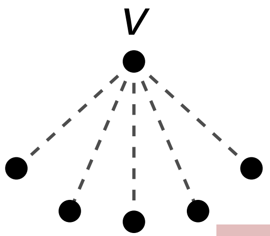
Existence of red edges depends on existence of green ones.

- Since many real-world networks are $\overbrace{\textit{sparse}}^{\text{number of edges linear in number of nodes}}$, we focus on $p = \frac{c}{n}$ for $c \in \Theta(1)$

ER – Degree of a Vertex

Vertex Degree

- Number of neighbors, number of incident edges
- each of $n-1$ potential edges exists with prob. p
- $\deg(v) \sim \mathrm{Bin}(n-1, p)$



$\to \Pr[\deg(v) = k] = \binom{n-1}{k} p^k (1-p)^{n-1-k}$

$\to \mathbb{E}[\deg(v)] = (n-1)p$ & $\mathrm{Var}[\deg(v)] = (n-1)p(1-p)$ *Inconvenient...*

$G(n,p)$
Independently connect any two nodes with fixed probability p.

$p = \frac{c}{n}$, $c \in \Theta(1)$

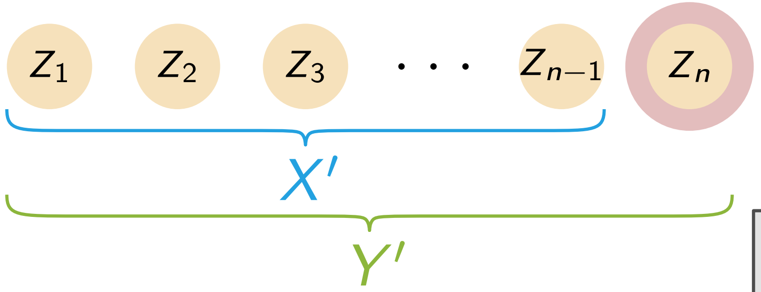
Approximation

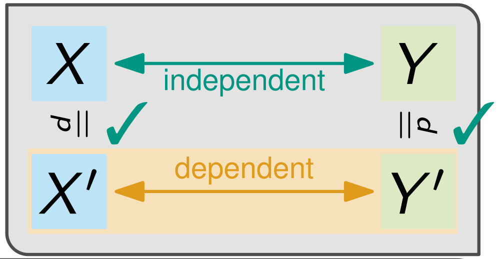
Lemma: Let $p = \frac{c}{n}$ for $c \in \Theta(1)$, let $X \sim \mathrm{Bin}(n-1, p)$ and let $Y \sim \mathrm{Bin}(n, p)$. Then, $d_{TV}(X, Y) = o(1)$.

Proof

- Independent $Z_i \sim \mathrm{Ber}(p)$ for $i \in [n]$
- $X' = \sum_{i=1}^{n-1} Z_i$, $Y' = X' + Z_n$

$$d_{TV}(X,Y) \le \Pr[X' \neq Y'] = \Pr[Z_n = 1] = \frac{c}{n} = o(1) ✓$$



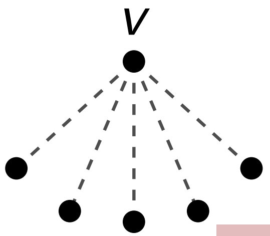


Coupling Inequality
For any coupling (X', Y') of X, Y: $d_{TV}(X, Y) \le \Pr[X' \neq Y']$.

ER – Degree of a Vertex

Vertex Degree

- Number of neighbors, number of incident edges
- each of $n-1$ potential edges exists with prob. p
- $\deg(v) \sim \mathrm{Bin}(n-1, p)$ → $\Pr[\deg(v) = k] = \binom{n-1}{k} p^k (1-p)^{n-1-k}$
  → $\mathbb{E}[\deg(v)] = (n-1)p$ & $\mathrm{Var}[\deg(v)] = (n-1)p(1-p)$ *Inconvenient...*



$G(n,p)$
Independently connect any two nodes with fixed probability p.

$p = \frac{c}{n}$, $c \in \Theta(1)$

Approximation

Lemma: Let $p = \frac{c}{n}$ for $c \in \Theta(1)$, let $X \sim \mathrm{Bin}(n-1, p)$ and let $Y \sim \mathrm{Bin}(n, p)$. Then, $d_{TV}(X, Y) = o(1)$. And for $Z \sim \mathrm{Pois}(c + O(\frac{1}{n}))$: $d_{TV}(X, Z) = o(1)$.

$$d_{TV}(X,Z) \leq d_{TV}(X,Y) + d_{TV}(Y,Z)$$
$$= o(1) + \underbrace{\tfrac{n}{2}\log(1-p)^2}_{\frac{n}{2}(-p-O(p^2)))^2} = o(1)$$
$$= \tfrac{n}{2}(p^2 + O(p^3))$$
$$= \tfrac{n}{2}((\tfrac{c}{n})^2 + O((\tfrac{c}{n})^3))$$
$$= \tfrac{c^2}{2n} + O(\tfrac{c^3}{n^2}) = o(1)$$

- $\mathbb{E}[Z] = \mathrm{Var}[Z] \approx c$, much simpler than the above!

Binomial-Poisson-Approximation
$Y \sim \mathrm{Bin}(n,p)$, $Z \sim \mathrm{Pois}(-n\log(1-p))$:
$d_{TV}(Y,Z) \leq \frac{n}{2}\log(1-p)^2$.

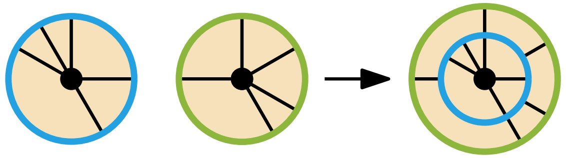
Triangle Inequality
$d_{TV}(X,Z) \leq d_{TV}(X,Y) + d_{TV}(Y,Z)$.

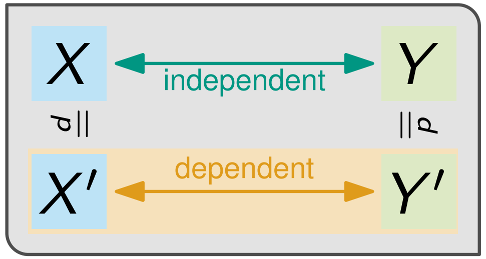
Taylor $p \to 0$: $\log(1-p) = -p - O(p^2)$

Conclusion

Coupling

- Define relation between rand. var. to make statements about one using the other
- A coupling of (X, Y) is a pair (X', Y') of random variables in a shared probability space such that $X \overset{d}{=} X'$ and $Y \overset{d}{=} Y'$
- Often X' and Y' dependent
- Examples: Wheel of fortune & Unfair dice
- Coupling inequality to bound *total variation distance*





Random Graph Models

- Mathematical models represent real-world networks and allow for theoretical analysis
- Desirable properties: simple, realistic, fast to generate

Erdős-Rényi Random Graphs

- $G(n, p)$: Start with n nodes, connect any two with fixed probability p, independently
- In sparse $G(n, p)$ the degree of a vertex is approximately Poisson-distributed

Outlook: Degree Distribution vs. Degree Distribution

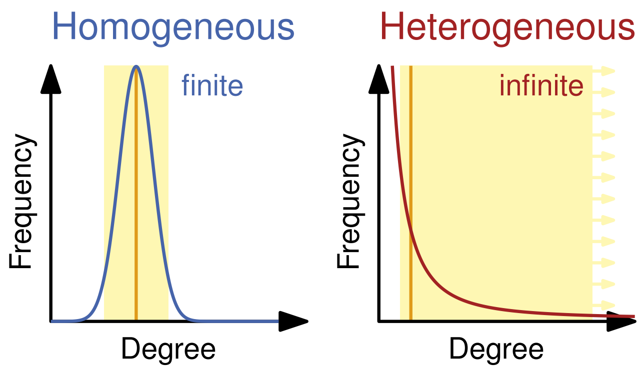
Distributions

- Probability distribution of the degree of a *given* vertex in a $G(n, \frac{c}{n})$ approaches $\mathrm{Pois}(c)$
- Empirical distribution of the degrees of *all* vertices in a graph $G = (V, E)$

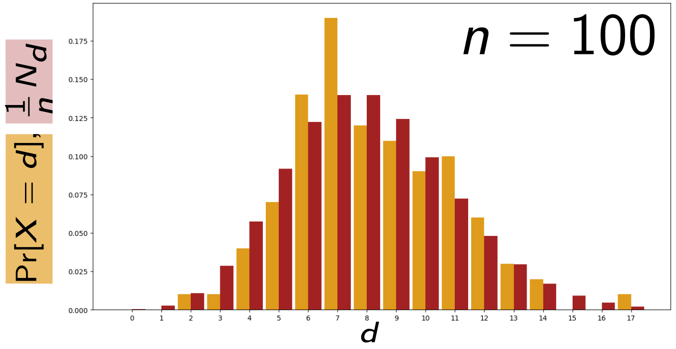
  $N_d = \sum_{v \in V} \mathbb{1}_{\{\deg(v)=d\}}$ (normalized: $\frac{1}{n} N_d$, for $n = |V|$)

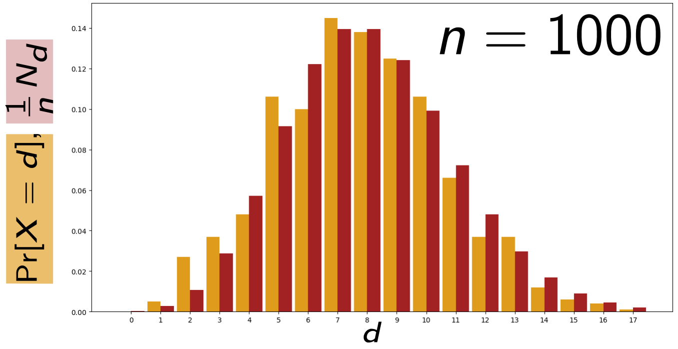
Characterizing a Distribution

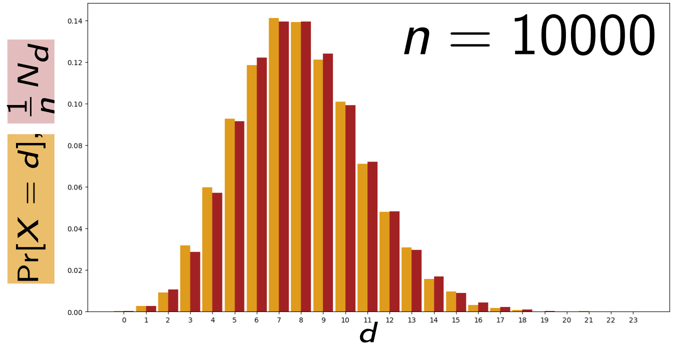
- *Mean*: What degree would we expect for a vertex?
- *Variance*: (very rough intuition) How far would we expect the degree of a vertex to deviate from the mean?



Empirical Distribution of $G(n, \frac{c}{n})$ → homogeneous







Karlsruhe Institute of Technology

Probability & Computing

Concentration

Expectation Management

What does it mean?

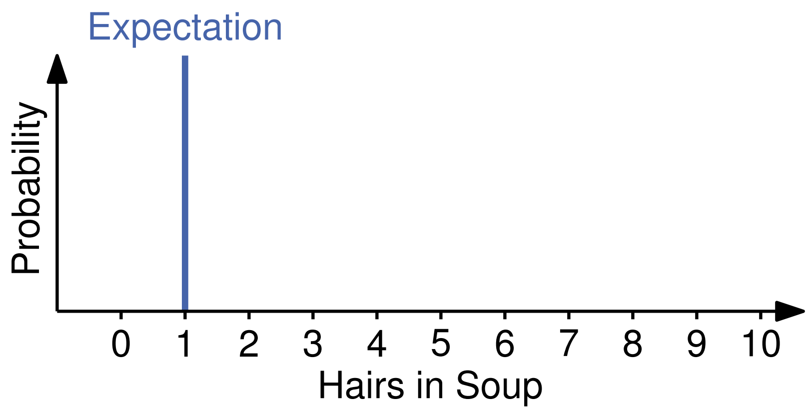
- "QuickSort has an *expected* running time of $O(n\log(n))$."
- "The vertex has an *expected* degree of c."
- "*In expectation* there is one hair in my soup."

Expectation

- The average of infinitly many trials
- How useful is that information in practice?
- Does not tell us much about the shape of the distribution
- Does not come with a level of certainty

Concentration

- In practice, expectation is often a good start
- But for meaningful statements, we need to know how likely we are close to the exepcation



Knowing that the expected value is 1 hair:

How likely is it that I get at least 10?

Not at all Somewhat

How likely is it that I get less than 2?

Extremely Somewhat

Definition: A **concentration inequality** bounds the probability of a random variable to deviate from a given value (typically its expectation) by a certain amount.

Markov's Inequality

Karlsruhe Institute of Technology

About Markov

- Andrei "The Furious" Andreyevich Markov (Russian mathematician)
- Unhappy with the state of living at the time (1921)
- Informed St. Petersburg Academy of Sciences that he could not attend meetings due to not having shoes
- After getting shoes from the Communist Party he replied:

  Finally, I received footwear. However, it is stupidly stitched together and does not accord with my measurements. Thus I cannot attend the meetings. I propose placing the footwear in a museum, as an example of the material culture of the current time.

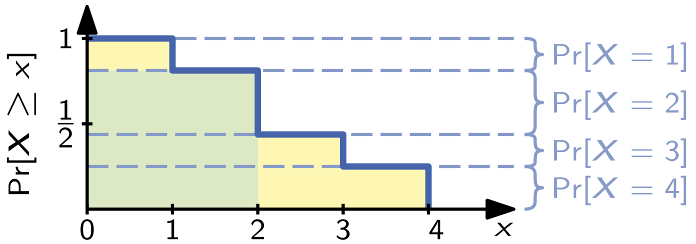
"Shape, The hidden geometry of absolutely everything", Jordan Ellenberg

Markov's Inequality

Karlsruhe Institute of Technology

Theorem (Markov's inequality): Let X be a non-negative random variable and let $a > 0$. Then, $\Pr[X \geq a] \leq \mathbb{E}[X]/a$.

Visual Proof



$$\mathbb{E}[X] = \sum_x x \cdot \Pr[X = x] \;\geq\; a \cdot \Pr[X \geq a]$$

fits into

Proof $\mathbb{E}[X] = \underbrace{\mathbb{E}[X \mid X < a]}_{\geq 0} \cdot \underbrace{\Pr[X < a]}_{\geq 0} + \underbrace{\mathbb{E}[X \mid X \geq a]}_{\geq a} \cdot \Pr[X \geq a] \geq a \cdot \Pr[X \geq a]$ ✓

Corollary: Let X be a non-negative rand. var. and $a > 0$. Then, $\Pr[X \geq a \cdot \mathbb{E}[X]] \leq 1/a$.

- "*In expectation* there is one hair in my soup."
  - How likely is it that I get at least 10? $\Pr[X \geq 10] \leq 1/10$
  - How likely is that I get less than 2? $\Pr[X < 2] = 1 - \Pr[X \geq 2] \geq 1 - 1/2 = 1/2$

Oh no...

Application: Unfair Coins

Karlsruhe Institute of Technology

- The sum of 20 unfair $\{0,1\}$-coin tosses: $X \sim \mathrm{Bin}(20, \frac{1}{5})$
- What is the probability of getting at least 16 ones?

1 0 1 0 0 1 0 0 1 1

0 0 0 0 1 0 0 0 1 1

$X = 8$

Markov: X non-negative, $a > 0$: $\Pr[X \geq a] \leq \mathbb{E}[X]/a$.

$$\Pr[X \geq 16] \leq \underbrace{\mathbb{E}[X]}_{20 \cdot \frac{1}{5} = 4}/16 = 0.25$$

- How tight is that bound? Not very?

$$\Pr[X \geq 16] = \sum_{k=16}^{20} \binom{20}{k} \left(\frac{1}{5}\right)^k \cdot \left(1 - \frac{1}{5}\right)^{20-k} \approx 0.0000000138$$

Maybe it is just a weak bound?

Fair Coin

- A single $\{0,1\}$-coin toss: $Y \sim \mathrm{Ber}(\frac{1}{2})$
- What is the probability of getting at least 1?
  - Clearly: $\Pr[Y \geq 1] = \Pr[Y = 1] = \frac{1}{2}$
  - Markov: $\Pr[Y \geq 1] \leq \mathbb{E}[Y]/1 = \mathbb{E}[Y] = \frac{1}{2}$

There exists a random variable and an $a > 0$ such that Markov's inequality is *exact*.

$\Rightarrow$ There is no better bound (that relies only on the expected value)

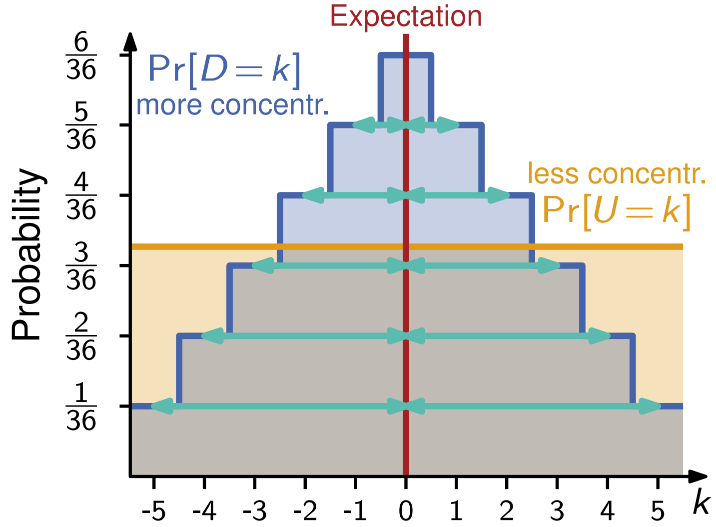
We need more information about the shape of the distribution!

Characterizing the Shape of a Distribution

- How much information do we need to characterize the shape of a distribution?

Example

- X, Y independent fair die-rolls, $D = X - Y$
- U uniform distribution over $\{-5, -4, \ldots, 5\}$
- Consider all probabilities individually Tedious... We need to aggregate!



Expectation? $f(k) = k$

$\mathbb{E}[D] = \sum_k \Pr[D = k] \cdot k = 0$

$\mathbb{E}[U] = \sum_k \Pr[U = k] \cdot k = 0$

Same value, different shapes (also just seen with Markov: $\mathbb{E}$ not enough)

- Problem: $+$ & $-$ terms cancel

$\Rightarrow$ Fix: absolute value $f(k) = |k|$

$\mathbb{E}[|D|] = \sum_k \Pr[D = k] \cdot |k| \approx 1.945$

$\mathbb{E}[|U|] = \sum_k \Pr[U = k] \cdot |k| \approx 2.727$

Distance to $\mathbb{E}$

More concentrated! Smaller expected distance to $\mathbb{E}$

- Problem: Nobody likes absolute value

$\Rightarrow$ Fix: square instead $f(k) = k^2$

$\mathbb{E}[D^2] = \sum_k \Pr[D = k] \cdot k^2 \approx 5.833$

$\mathbb{E}[U^2] = \sum_k \Pr[U = k] \cdot k^2 = 10.0$

Squared distance to $\mathbb{E}$

These are just expectations of functions of random variables!

Do you have a Moment?

Expectation and Functions

- Random variable X taking values in a set S
- A function f, e.g. $f(X) = X$; $f(X) = |X|$, $f(X) = X^2$, $f(X) = \sqrt{X}$, $f(X) = X^3$, $f(X) = e^X$
- $\mathbb{E}[f(X)] = \sum_{x \in S} \Pr[X = x] \cdot f(x)$

These turn out to be particularly useful!

Moments

Definition: For random variable X and $n \in \mathbb{N}$ the ***n*-th raw moment** is $\mathbb{E}[X^n]$.

- Just seen: For $\mathbb{E}[X] = 0$, this captures distances to $\mathbb{E}[X]$ What if $\mathbb{E}[X] \neq 0$?

Definition: For random variable X and $n \in \mathbb{N}$ the ***n*-th central moment** is $\mathbb{E}[(X - \mathbb{E}[X])^n]$.

- Just seen: the $\underbrace{\text{2nd central moment}}_{\mathbb{E}[(X - \mathbb{E}[X])^2] = \mathrm{Var}[X]}$ captures squared distances to the expected value

$$\mathbb{E}[(X - \mathbb{E}[X])^2] = \mathrm{Var}[X]$$

- The smaller the variance, the more concentrated the random variable

... and with Markov's help, we can turn that insight into a concentration inequality!

Chebychev's Inequality

Markov's teacher! (Markov's inequality actually appeared earlier in Chebychev's works)

Theorem (Chebychev's inequality): Let X be a random variable with finite variance and let $b > 0$. Then, $\Pr[|X - \mathbb{E}[X]| \geq b] \leq \mathrm{Var}[X]/b^2$.

Proof

Markov: $Y \geq 0$, $a > 0$: $\Pr[Y \geq a] \leq \mathbb{E}[Y]/a$

$$\Pr\left[|X - \mathbb{E}[X]| \geq b\right] = \Pr\left[\overbrace{(X - \mathbb{E}[X])^2}^{\geq 0} \geq b^2\right] \leq \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]/b^2 = \mathrm{Var}[X]/b^2 \ \checkmark$$

Application: Unfair Coins

1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1

- $X \sim \mathrm{Bin}(20, \frac{1}{5})$, $\Pr[X \geq 16]$? | $\mathbb{E}[X] = 20 \cdot \frac{1}{5} = 4$ | $\mathrm{Var}[X] = 20 \cdot \frac{1}{5} \cdot (1 - \frac{1}{5}) = \frac{16}{5}$

  $\Pr[X \geq 16] = \sum_{k=16}^{20} \binom{20}{k} (\frac{1}{5})^k \cdot (1 - \frac{1}{5})^{20-k} \approx 0.0000000138$

  $X \sim \mathrm{Bin}(n, p) : \mathrm{Var}[X] = np(1-p)$
- Markov: $\Rightarrow \Pr[X \geq 16] \leq \mathbb{E}[X]/16 = 0.25$
- Chebychev:

$$\begin{aligned}\Pr[X \geq 16] &\leq \Pr[X \geq 16 \vee X \leq -8] \\ &= \Pr[|X - \mathbb{E}[X]| \geq 12] \\ &\leq \frac{\mathrm{Var}[X]}{12^2} = \frac{16}{5 \cdot 144} \approx 0.022\end{aligned}$$

Order of magnitude better than Markov!

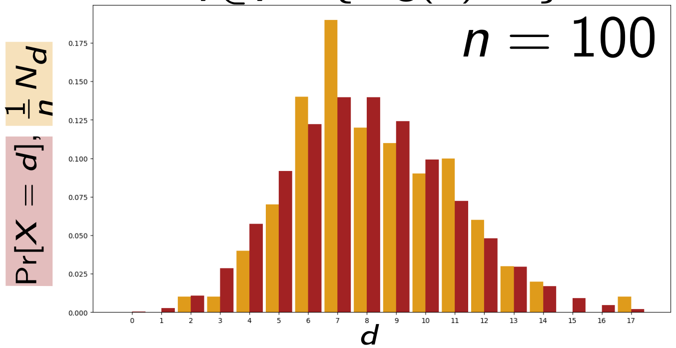
$$\begin{aligned} & X \geq 16 \\ \Leftrightarrow\ & X - \mathbb{E}[X] \geq 16 - \mathbb{E}[X] \\ \Leftrightarrow\ & X - \mathbb{E}[X] \geq 12 \\ & |X - \mathbb{E}[X]| \geq 12 \Rightarrow X \geq 16 \text{ or } X \leq -8\end{aligned}$$

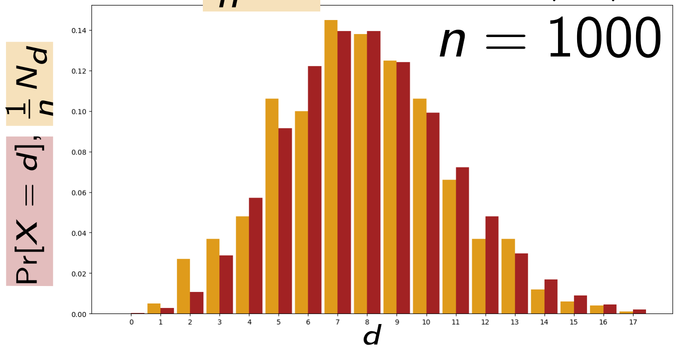
Application: ER – Degree Distribution

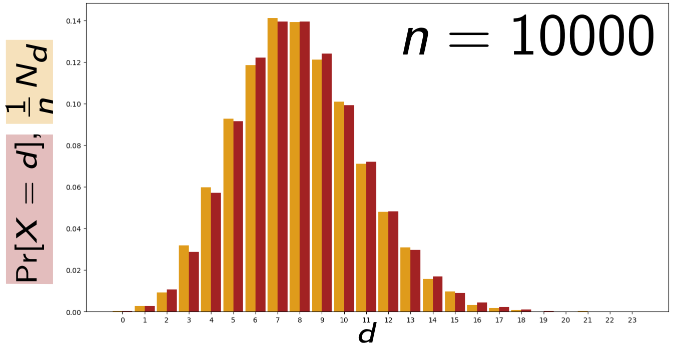
Recap

- $G(n, p)$: Start with n nodes, connect any two with fixed probability p, independently
- Probability distribution of the degree of a *single* node v: $\deg(v) \sim \text{Bin}(n-1, p)$
- For $p = c/n$ with $c \in \Theta(1)$ the degree of a vertex is approximately Poisson-distributed
  - Total variation distance of X, Y taking values in a set S: $d_{TV}(X, Y) = \frac{1}{2} \sum_{x \in S} |\Pr[X = x] - \Pr[Y = x]|$
  - For $\lambda = -n \log(1-p) = c + O(1/n)$ and $X \sim \text{Pois}(\lambda)$ we have $d_{TV}(\deg(v), X) = o(1)$
- Empirical distribution of the degrees of *all* vertices in a graph $G = (V, E)$

$N_d = \sum_{v \in V} \mathbb{1}_{\{\deg(v) = d\}}$ (normalized: $\frac{1}{n} N_d$, for $n = |V|$)







Application: ER – Degree Distribution

Theorem: Consider a $G(n,p)$ with $p = c/n$ for constant $c > 0$. For $\lambda = -n\log(1-p)$, let $X \sim \text{Pois}(\lambda)$. Then for all $d > 0$ and every $\varepsilon > 0$ we have

$\lambda = c + O(1/n) \to c$ for $n \to \infty$

$$\lim_{n\to\infty} \Pr\left[\left|\Pr[X=d] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] = 0.$$

Proof

- Step 1: $\Pr[X{=}d]$ is close to the expectation of $\frac{1}{n}N_d$ $\quad \lim_{n\to\infty} \left|\Pr[X=d] - \mathbb{E}\left[\frac{1}{n}N_d\right]\right| = 0$ ✓

$$\left|\Pr[X=d] - \underbrace{\mathbb{E}\left[\tfrac{1}{n}N_d\right]}\right| = |\Pr[X=d] - \Pr[\deg(v)=d]| \leq \textstyle\sum_{d\geq 0} |\Pr[X=d] - \Pr[\deg(v)=d]|$$

$$\begin{aligned}
&= \tfrac{1}{n}\mathbb{E}[N_d] \\
&= \tfrac{1}{n}\mathbb{E}\left[\textstyle\sum_{v\in V} \mathbb{1}_{\{\deg(v)=d\}}\right] \\
&= \tfrac{1}{n}\textstyle\sum_{v\in V} \mathbb{E}[\mathbb{1}_{\{\deg(v)=d\}}] \\
&= \tfrac{1}{n}\textstyle\sum_{v\in V} \Pr[\deg(v)=d] \\
&= \Pr[\deg(v)=d]
\end{aligned}$$

$$= 2 \cdot d_{TV}(X, \deg(v))$$

$$= o(1) \xrightarrow{n\to\infty} 0 \text{ ✓}$$

Already shown last time!

$d_{TV}(X,Y) = \frac{1}{2}\sum_{x\in S} |\Pr[X=x] - \Pr[Y=x]|$

- Step 2: $\frac{1}{n}N_d$ is concentrated $\quad \lim_{n\to\infty} \Pr\left[\left|\mathbb{E}\left[\frac{1}{n}N_d\right] - \frac{1}{n}N_d\right| \geq \varepsilon\right] = 0$

Step 2: Concentration of $\frac{1}{n}N_d$

$$\Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] \leq \underbrace{\mathrm{Var}\left[\tfrac{1}{n}N_d\right]}\,/\varepsilon^2$$

$$\begin{aligned}
\mathrm{Var}[\tfrac{1}{n}N_d] &= \mathbb{E}[(\tfrac{1}{n}N_d)^2] - \mathbb{E}[\tfrac{1}{n}N_d]^2 \\
&= \tfrac{1}{n^2}\left(\underbrace{\mathbb{E}[(N_d)^2]} - \underbrace{\mathbb{E}[N_d]^2}_{= (n\Pr[\deg(v)=d])^2 \text{ (see Step 1)}}\right)
\end{aligned}$$

$N_d = \sum_{v\in V} \mathbb{1}_{\{\deg(v)=d\}}$
$$\begin{aligned}
\mathbb{E}[(N_d)^2] &= \mathbb{E}\Big[\Big(\sum_{v\in V} \mathbb{1}_{\{\deg(v)=d\}}\Big)^2\Big] \\
&= \mathbb{E}\Big[\sum_{v\in V}(\mathbb{1}_{\{\deg(v)=d\}})^2 + \sum_{v\in V}\sum_{u\neq v} \mathbb{1}_{\{\deg(v)=d\}} \cdot \mathbb{1}_{\{\deg(u)=d\}}\Big]
\end{aligned}$$

Indicator RV X: $X^2 = X$, Lin. of Exp.
$$= \mathbb{E}\Big[\sum_{v\in V} \mathbb{1}_{\{\deg(v)=d\}}\Big] + \mathbb{E}\Big[\sum_{v\in V}\sum_{u\neq v} \mathbb{1}_{\{\deg(v)=d\}} \cdot \mathbb{1}_{\{\deg(u)=d\}}\Big]$$

Lin. of Exp.
$$= \sum_{v\in V} \underbrace{\mathbb{E}[\mathbb{1}_{\{\deg(v)=d\}}]}_{=\Pr[\deg(v)=d]} + \underbrace{\sum_{v\in V}\sum_{u\neq v} \mathbb{E}[\underbrace{\mathbb{1}_{\{\deg(v)=d\}} \cdot \mathbb{1}_{\{\deg(u)=d\}}}_{=1 \text{ iff } \deg(v)=d \,\wedge\, \deg(u)=d}]}_{=\Pr[\deg(v)=d \,\wedge\, \deg(u)=d]}$$

$$= n\cdot\Pr[\deg(v)=d] + n(n-1)\cdot\Pr[\deg(v)=d \wedge \deg(u)=d]$$

$$\lim_{n\to\infty} \Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] = 0$$

Chebychev: X finite variance, $b > 0$
$\Pr[|X - \mathbb{E}[X]| \geq b] \leq \mathrm{Var}[X]/b^2$

$$\Big(\sum_i a_i\Big)^2 = \sum_i a_i^2 + \sum_i \sum_{j\neq i} a_i a_j$$

Step 2: Concentration of $\frac{1}{n} N_d$

$$\Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n} N_d\right] - \tfrac{1}{n} N_d\right| \geq \varepsilon\right] \leq \underbrace{\operatorname{Var}\left[\tfrac{1}{n} N_d\right]} / \varepsilon^2$$

$$\begin{aligned}
\operatorname{Var}[\tfrac{1}{n} N_d] &= \mathbb{E}[(\tfrac{1}{n} N_d)^2] - \mathbb{E}[\tfrac{1}{n} N_d]^2 \\
&= \frac{1}{n^2}\left(\mathbb{E}[(N_d)^2] - \mathbb{E}[N_d]^2\right) \\
&= \frac{1}{n^2}\big(n \Pr[\deg(v) = d] \\
&\qquad + n(n-1) \Pr[\deg(v) = d \wedge \deg(u) = d] \\
&\qquad - (n \Pr[\deg(v) = d])^2\big) \\
&= \frac{1}{n} \Pr[\deg(v) = d] && \leq 1 \\
&\qquad + \frac{n-1}{n} \Pr[\deg(v) = d \wedge \deg(u) = d] && \leq 1 \\
&\qquad - \Pr[\deg(v) = d]^2 \\
&\leq \frac{1}{n} + \Pr[\deg(v) = d \wedge \deg(u) = d] \\
&\qquad - \Pr[\deg(v) = d]^2
\end{aligned}$$

$$\lim_{n \to \infty} \Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n} N_d\right] - \tfrac{1}{n} N_d\right| \geq \varepsilon\right] = 0$$

Chebychev: X finite variance, $b > 0$
$\Pr[|X - \mathbb{E}[X]| \geq b] \leq \operatorname{Var}[X]/b^2$

$$\left(\sum_i a_i\right)^2 = \sum_i a_i^2 + \sum_i \sum_{j \neq i} a_i a_j$$

Step 2: Concentration of $\frac{1}{n}N_d$

$$\Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] \leq \underbrace{\mathrm{Var}\left[\tfrac{1}{n}N_d\right]}\,/\varepsilon^2$$

$$\begin{aligned}
\mathrm{Var}[\tfrac{1}{n}N_d] &= \mathbb{E}[(\tfrac{1}{n}N_d)^2] - \mathbb{E}[\tfrac{1}{n}N_d]^2 \\
&= \frac{1}{n^2}\left(\mathbb{E}[(N_d)^2] - \mathbb{E}[N_d]^2\right) \\
&\leq \frac{1}{n} + \Pr[\deg(v) = d \wedge \deg(u) = d] \\
&\quad - \Pr[\deg(v) = d]\Pr[\deg(u) = d] \\
&= \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d] \\
&\quad - \Pr[X_1 + Y_1 = d]\Pr[X_2 + Y_2 = d] \\
&= \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d] \\
&\quad - \Pr[X_1 + Y_1 = d \wedge X_2 + Y_2 = d] \\
&\leq \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d \\
&\quad \wedge (X_1 + Y_1 \neq d \vee X_2 + Y_2 \neq d)]
\end{aligned}$$

For the whole event to occur, this needs to happen

Which excludes this from happening

$$\deg(v) \stackrel{d}{=} \deg(u)$$

$$\lim_{n\to\infty} \Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] = 0$$

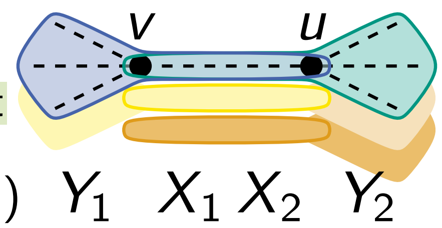
Chebychev: X finite variance, $b > 0$
$\Pr[|X - \mathbb{E}[X]| \geq b] \leq \mathrm{Var}[X]/b^2$

$$\left(\sum_i a_i\right)^2 = \sum_i a_i^2 + \sum_i \sum_{j\neq i} a_i a_j$$

Fréchet: $\Pr[A] - \Pr[B] \leq \Pr[A \wedge \bar{B}]$

Couplings

- Consider $\deg(u)$ and $\deg(v)$
- $Y_1, Y_2 \sim \mathrm{Bin}(n-2, p)$ } independent
- $X_1, X_2 \sim \mathrm{Ber}(p)$ } independent
- $(\deg(v), \deg(u)) \stackrel{d}{=} (X_1+Y_1, X_1+Y_2)$



$$\begin{array}{ccc}
\deg(v) & \xleftrightarrow{\text{dependent}} & \deg(u) \\
\| d & & \| d \\
X_1+Y_1 & \xleftrightarrow{\text{dependent}} & X_1+Y_2
\end{array}
\qquad
\begin{array}{ccc}
\deg(v) & \xleftrightarrow{\text{dependent}} & \deg(u) \\
\| d & & \| d \\
X_1+Y_1 & \xleftrightarrow{\text{independent}} & X_2+Y_2
\end{array}$$

Step 2: Concentration of $\frac{1}{n}N_d$

$$\Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] \leq \underbrace{\mathrm{Var}\left[\tfrac{1}{n}N_d\right]}\,/\varepsilon^2$$

$$\begin{aligned}
\mathrm{Var}[\tfrac{1}{n}N_d] &= \mathbb{E}[(\tfrac{1}{n}N_d)^2] - \mathbb{E}[\tfrac{1}{n}N_d]^2 \\
&= \frac{1}{n^2}\left(\mathbb{E}[(N_d)^2] - \mathbb{E}[N_d]^2\right) \\
&\leq \frac{1}{n} + \Pr[\deg(v) = d \wedge \deg(u) = d] \\
&\quad - \Pr[\deg(v) = d]\Pr[\deg(u) = d] \qquad \deg(v) \overset{d}{=} \deg(u) \\
&= \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d] \\
&\quad - \Pr[X_1 + Y_1 = d]\Pr[X_2 + Y_2 = d] \\
&= \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d] \\
&\quad - \Pr[X_1 + Y_1 = d \wedge X_2 + Y_2 = d] \\
&\leq \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d \\
&\quad \wedge (X_1 + Y_1 \neq d \vee X_2 + Y_2 \neq d)] = \frac{1}{n} + \Pr[X_1 + Y_1 = d \wedge X_1 + Y_2 = d \wedge X_2 + Y_2 \neq d]
\end{aligned}$$

For the whole event to occur, this needs to happen

Which excludes this from happening

$$\lim_{n\to\infty} \Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right] - \tfrac{1}{n}N_d\right| \geq \varepsilon\right] = 0$$

Chebychev: X finite variance, $b > 0$
$\Pr[|X - \mathbb{E}[X]| \geq b] \leq \mathrm{Var}[X]/b^2$

$$\left(\sum_i a_i\right)^2 = \sum_i a_i^2 + \sum_i \sum_{j \neq i} a_i a_j$$

Fréchet: $\Pr[A] - \Pr[B] \leq \Pr[A \wedge \bar{B}]$

Step 2: Concentration of $\frac{1}{n}N_d$

$$\Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right]-\tfrac{1}{n}N_d\right|\geq\varepsilon\right]\leq\underbrace{\mathrm{Var}\left[\tfrac{1}{n}N_d\right]}/\varepsilon^2\xrightarrow{n\to\infty}0$$

$$\lim_{n\to\infty}\Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n}N_d\right]-\tfrac{1}{n}N_d\right|\geq\varepsilon\right]=0\ \checkmark$$

$$\mathrm{Var}[\tfrac{1}{n}N_d]=\mathbb{E}[(\tfrac{1}{n}N_d)^2]-\mathbb{E}[\tfrac{1}{n}N_d]^2\leq\tfrac{1}{n}+2p=\tfrac{1}{n}+2\tfrac{c}{n}\xrightarrow{n\to\infty}0$$

$$=\tfrac{1}{n^2}\left(\mathbb{E}[(N_d)^2]-\mathbb{E}[N_d]^2\right)$$

$$\leq\tfrac{1}{n}+\Pr[\deg(v)=d\wedge\deg(u)=d]$$

$$-\Pr[\deg(v)=d]\Pr[\deg(u)=d]\quad \deg(v)\stackrel{d}{=}\deg(u)$$

$$\leq\tfrac{1}{n}+\Pr[X_1+Y_1=d\wedge X_1+Y_2=d\wedge X_2+Y_2\neq d]$$

$$=\tfrac{1}{n}+\Pr[X_1+Y_1=d\wedge X_1+Y_2=d\wedge X_2+Y_2\neq d\mid X_1=0]\underbrace{\Pr[X_1=0]}_{\leq 1}$$

$$+\underbrace{\Pr[X_1+Y_1=d\wedge X_1+Y_2=d\wedge X_2+Y_2\neq d\mid X_1=1]}_{\leq 1}\Pr[X_1=1]$$

Law of total probability

$$\leq\tfrac{1}{n}+\Pr[Y_1=d\wedge Y_2=d\wedge X_2+Y_2\neq d\mid X_1=0]$$

$$+\Pr[X_1=1]$$

$\Rightarrow X_2=1$

independent

$$=\tfrac{1}{n}+\Pr[Y_1=d\wedge Y_2=d\wedge X_2=1\cancel{\mid X_1=0}]+\Pr[X_1=1]\leq\tfrac{1}{n}+\Pr[X_2=1]+\Pr[X_1=1]$$

Chebychev: X finite variance, $b>0$
$\Pr[|X-\mathbb{E}[X]|\geq b]\leq\mathrm{Var}[X]/b^2$

$$\left(\textstyle\sum_i a_i\right)^2=\textstyle\sum_i a_i^2+\sum_i\sum_{j\neq i}a_ia_j$$

Fréchet: $\Pr[A]-\Pr[B]\leq\Pr[A\wedge\bar{B}]$

- $Y_1,Y_2\sim\mathrm{Bin}(n-2,p)$
- $X_1,X_2\sim\mathrm{Ber}(p)$

independent

Application: ER – Degree Distribution

Karlsruhe Institute of Technology

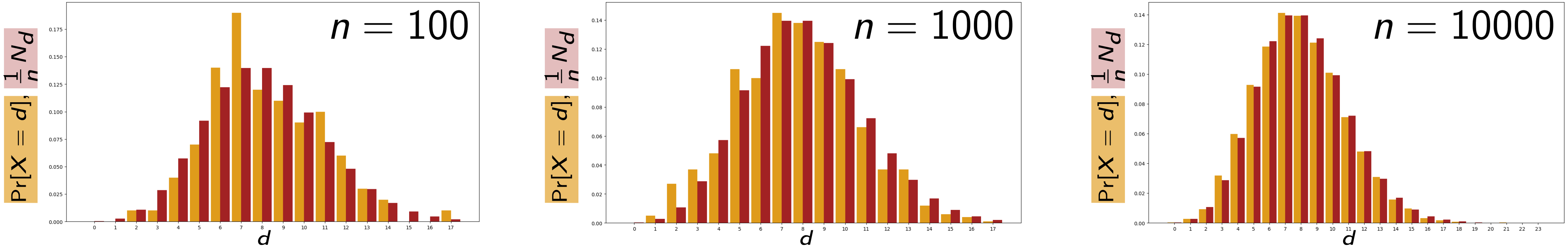
Theorem: Consider a $G(n, p)$ with $p = c/n$ for constant $c > 0$. For $\lambda = -n \log(1-p)$, let $X \sim \text{Pois}(\lambda)$. Then for all $d > 0$ and every $\varepsilon > 0$ we have

$\lambda = c + O(1/n) \to c$ for $n \to \infty$

$$\lim_{n\to\infty} \Pr\left[\left|\Pr[X = d] - \tfrac{1}{n} N_d\right| \geq \varepsilon\right] = 0.$$

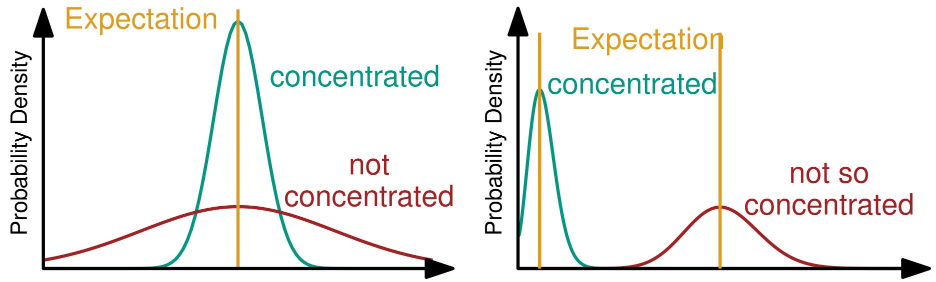
Proof

- Step 1: $\Pr[X = d]$ is close to the expectation of $\frac{1}{n} N_d$ $\qquad \lim_{n\to\infty} \left|\Pr[X = d] - \mathbb{E}\left[\tfrac{1}{n} N_d\right]\right| = 0$ ✓
- Step 2: $\frac{1}{n} N_d$ is concentrated (via Chebychev) $\qquad \lim_{n\to\infty} \Pr\left[\left|\mathbb{E}\left[\tfrac{1}{n} N_d\right] - \tfrac{1}{n} N_d\right| \geq \varepsilon\right] = 0$ ✓



Concentration Bounds So Far

Definition: A **concentration inequality** bounds the probability of a random variable to deviate from a given value (typically its expectation) by a certain amount.



Markov

- based on expectation (first moment)
- X non-negative random variable and $a > 0$

  $\Pr[X \geq a] \leq \mathbb{E}[X]/a$
- tight

Chebychev

- based on variance (second moment)
- X random variable with finite variance and $b > 0$

  $\Pr[|X - \mathbb{E}[X]| \geq b] \leq \mathrm{Var}[X]/b^2$
- tight (stated without proof)

Can we utilize higher-order moments for even stronger bounds?

Another Moment Please

- The n-th raw moment of a random variable X is $\mathbb{E}[X^n]$
- We can capture *all* moments of X using a single function

Looks scary, but is again just $\mathbb{E}[f(X)]$ for $f(X) = e^{tX}$

Definition: For a random variable X the **moment generating function** is $M_X(t) = \mathbb{E}[e^{tX}]$

- Where the name comes from: For the n-th derivative $M_X^{(n)}(t)$ we have $M_X^{(n)}(0) = \mathbb{E}[X^n]$

(assuming the function exists in a neighborhood around 0)

Theorem: For independent random variables X, Y: $M_{X+Y}(t) = M_X(t) \cdot M_Y(t)$.

Proof $M_{X+Y}(t) = \mathbb{E}[e^{t(X+Y)}] = \mathbb{E}[e^{tX} \cdot e^{tY}] = \mathbb{E}[e^{tX}] \cdot \mathbb{E}[e^{tY}] = M_X(t) \cdot M_Y(t)$ ✓

Concentration Inequality

Had his 100th birthday in 2023! Thought the bound (now named after him) to be so trivial that he didn't mention that it actually came from Herman Rubin.

"A conversation with Herman Chernoff", John Bather, Statist. Sci. 1996

Theorem (Chernoff Bounds): Let X be a random variable and $a > 0$.
Then, $\Pr[X \geq a] \leq \min_{t>0} \mathbb{E}[e^{tX}]/e^{ta}$ and $\Pr[X \leq a] \leq \min_{t<0} \mathbb{E}[e^{tX}]/e^{ta}$.

Proof for all $t > 0$: $\Pr[X \geq a] = \Pr[e^{tX} \geq e^{ta}] \leq \mathbb{E}[e^{tX}]/e^{ta}$

$\leq \min_{t>0} \mathbb{E}[e^{tX}]/e^{ta}$ ✓

for all $t < 0$: analogous. ✓

Markov: X non-negative, $b > 0$:
$\Pr[X \geq b] \leq \mathbb{E}[X]/b$.

Get bounds for specific random variables by finding a good t!

Application: Binomial Distribution

Theorem: Let $X \sim \mathrm{Bin}(n, p)$. Then for any $\varepsilon > 0$

$$\Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] \leq \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mathbb{E}[X]}.$$

Chernoff: Random variable X and $a > 0$: $\Pr[X \geq a] \leq \min_{t>0} \mathbb{E}[e^{tX}]/e^{ta}$.

Mom. Gen. Function: $M_X(t) = \mathbb{E}[e^{tX}]$

Moment Addition: Independent X, Y: $M_{X+Y}(t) = M_X(t) \cdot M_Y(t)$.

Proof Consider X as the sum of independent $X_i \sim \mathrm{Ber}(p)$

$$M_{X_i}(t) = \mathbb{E}[e^{tX_i}] = \Pr[X_i = 0] \cdot e^{t \cdot 0} + \Pr[X_i = 1] \cdot e^{t \cdot 1}$$
$$= (1-p) + pe^t = 1 + (e^t - 1)p \underset{1+x \leq e^x}{\leq} e^{(e^t-1)p}$$

$$M_X(t) = M_{\sum X_i}(t) = \prod_{i=1}^n M_{X_i}(t) \leq \prod_{i=1}^n e^{(e^t-1)p} = e^{(e^t-1)\cdot np}$$
$$= e^{(e^t-1)\mathbb{E}[X]}$$

$$\Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] \leq \min_{t>0} \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\varepsilon)\mathbb{E}[X]}} \leq \min_{t>0} \frac{e^{(e^t-1)\mathbb{E}[X]}}{e^{t(1+\varepsilon)\mathbb{E}[X]}} = \min_{t>0} \left(\frac{e^{(e^t-1)}}{e^{t(1+\varepsilon)}}\right)^{\mathbb{E}[X]} \underset{\text{for } t = \log(1+\varepsilon)}{\leq} \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mathbb{E}[X]} \checkmark$$

Example 1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1

- Sum of 20 unfair $\{0,1\}$-coin tosses: $X \sim \mathrm{Bin}(20, \frac{1}{5})$, $\mathbb{E}[X] = 4$
- $\Pr[X \geq 16] = \Pr[X \geq (1+3)\mathbb{E}[X]] \leq \left(\frac{e^3}{(1+3)^{1+3}}\right)^4 = \frac{e^{12}}{4^{4^4}} \approx 0.00003789$

Markov:	≤ 0.25
Chebychev:	$\lessapprox 0.022$
Actual:	≈ 0.0000000138

Institute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithms

Chernoff – Simpler Versions

Theorem: Let $X \sim \mathrm{Bin}(n, p)$. Then for any $\varepsilon > 0$

$$\Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] \leq \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mathbb{E}[X]}.$$

Chernoff: Random variable X and $a > 0$: $\Pr[X \geq a] \leq \min_{t>0} \mathbb{E}[e^{tX}]/e^{ta}$.

Corollary: Let $X \sim \mathrm{Bin}(n, p)$. Then for any $t \geq 6\mathbb{E}[X]$, $\Pr[X \geq t] \leq 2^{-t}$.

Corollary: Let $X \sim \mathrm{Bin}(n, p)$. Then for any $\varepsilon \in (0, 1]$, $\Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] \leq e^{-\varepsilon^2/3 \cdot \mathbb{E}[X]}$.

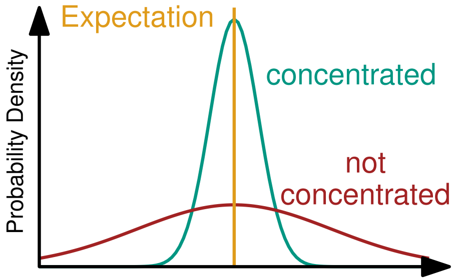
Corollary: Let $X \sim \mathrm{Bin}(n, p)$. Then for any $\varepsilon \in (0, 1)$, $\Pr[X \leq (1-\varepsilon)\mathbb{E}[X]] \leq e^{-\varepsilon^2/2 \cdot \mathbb{E}[X]}$.

- In fact, these also work when the X_i are Bernoulli random variables with different success probabilities

Conclusion

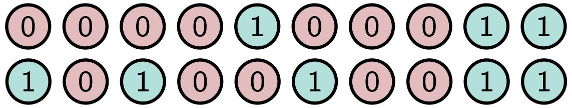
Concentration

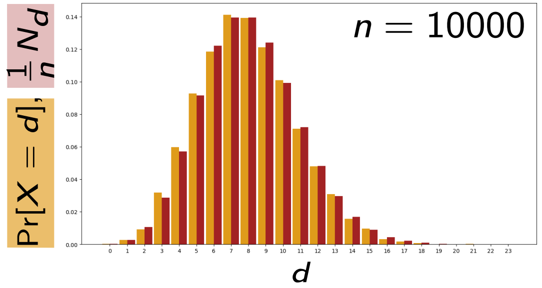
- Is a random variable likely to yield values close to the expectation?
- Concentration inequalities bound the probability for a random variable to deviate from its expectation



Moments

- Used to characterize the shape of a distribution
- First moment: expected value
- Second moment: variance
- Moment generating functions to determine higher-order moments





Concentration Inequalities

- Markov: Based on first moment
- Chebychev: Squaring within Markov (utilizing second moment)
- Chernoff: Exponentiating within Markov (utilizing moment generating functions)
- Examples: Sum of coin flips, empirical degree distribution of ER graphs

Karlsruhe Institute of Technology

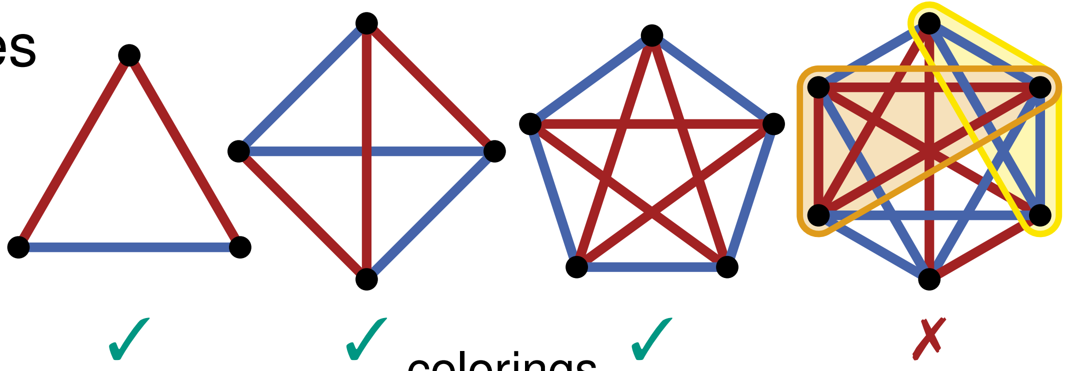
Probability & Computing

Probabilistic Method

Complete Coloring

The Problem

- Let G be the complete graph on n vertices (every vertex is adjacent to every other vertex)
- A k-clique is a complete subgraph with k vertices
- A coloring of the graph assigns each edge one of two colors: red or blue
- In a graph with n vertices, does there *exist* a coloring with *no* monochromatic k-clique?



colorings
$k = 3$

The Solution?

- Brute-force algorithm?
  - $n = 6 \Rightarrow 2^{n(n-1)} = 2^{30} = 1{,}073{,}741{,}824$ possible colorings
  - $k = 3 \Rightarrow \binom{6}{3} = 20$ triangles to check $\Rightarrow$ 60 edges per coloring
  - What about $n = 1000$ and $k = 20$?

  naive implementation: 20min
  no coloring exists

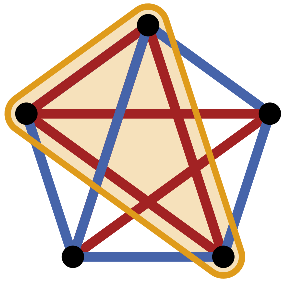
- Randomized algorithm?
  - How often shall we try before assuming that no coloring exists?

Randomized Coloring

Karlsruhe Institute of Technology

Algorithm

- For each edge independently, choose one of the colors with probability $1/2$
- Let X be the indicator variable with $X = 1$ if and only if the resulting coloring contains a monochromatic k-clique
- Let $H_1, \dots, H_{\binom{n}{k}}$ be all the different k-cliques
- Let X_i be the indicator variable with $X_i = 1$ if and only if H_i is monochromatic



- What is $\Pr[X_i = 1]$?
  - Consider the first edge that gets colored (we do not care which color it is, but...)
  - The $\binom{k}{2} - 1$ remaining edges need to get the same color

$$\Pr[X_i = 1] = \left(\tfrac{1}{2}\right)^{\binom{k}{2}-1} = 2^{-\binom{k}{2}+1}$$

- What is $\Pr[X = 1]$?

$$\Pr[X = 1] = \Pr\left[\exists_{i \in [\binom{n}{k}]} : X_i = 1\right] \overset{\text{union bound}}{\leq} \sum_{i=1}^{\binom{n}{k}} \Pr[X_i = 1]$$

$$= \binom{n}{k} 2^{-\binom{k}{2}+1} \leq \frac{n^k}{k!} 2^{-\frac{k(k-1)}{2}+1} = \frac{n^k}{k!} 2 \cdot 2^{-\frac{k^2-k}{2}} = \frac{n^k}{k!} 2 \cdot (2^{-\frac{k}{2}})^k \cdot 2^{\frac{k}{2}}$$

simplify by assuming $k \geq 2\log(n)$

$\Rightarrow 2^{-\frac{k}{2}} \leq \frac{1}{n}$

$$\leq \frac{1}{k!} 2\sqrt{2}^k \leq \frac{2\sqrt{2}^k}{e(\frac{k}{e})^k} = \underbrace{\frac{2}{e}}_{<1} \underbrace{\left(\frac{\sqrt{2}e}{k}\right)^k}_{<1} < 1$$

$$\Rightarrow \Pr[X = 0] = 1 - \Pr[X = 1] > 0$$

It may happen that the algorithm returns a coloring with the desired property!

not very confident...

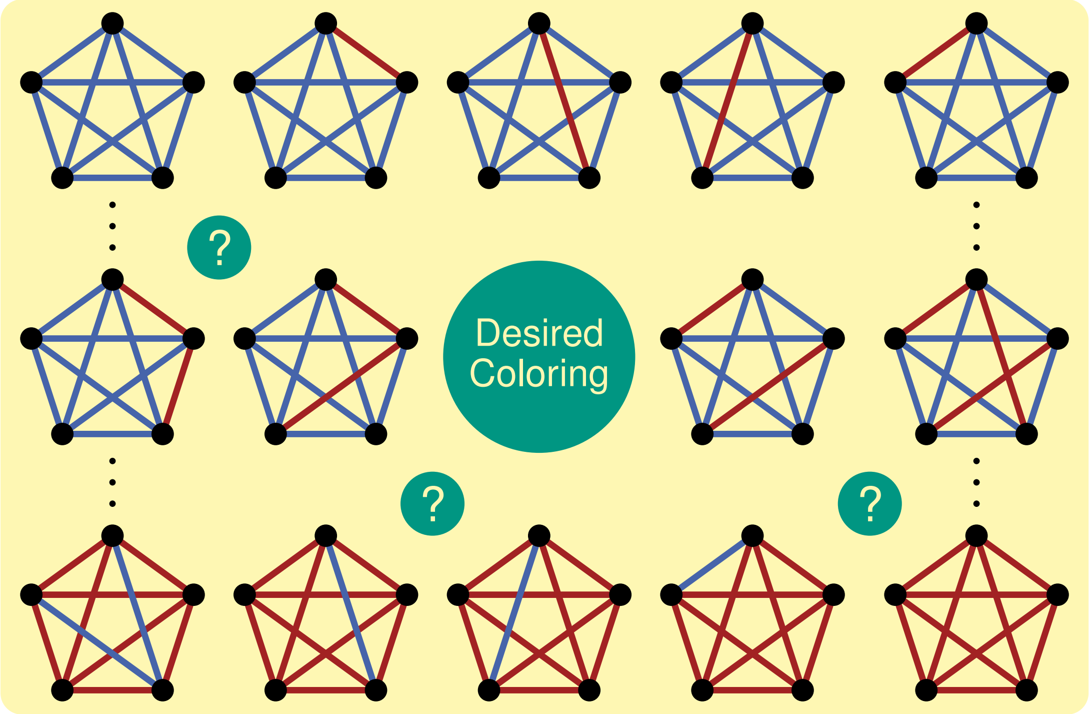
What did we just show?!

The Probability Space

- What is the sample space of the algorithm?
  - Each edge is red or blue with prob. $1/2$
  - $\binom{n}{2}$ edges $\Rightarrow 2^{\binom{n}{2}}$ possible colorings
- Each occurs with equal probability $1/2^{\binom{n}{2}}$

Just Shown

- $X = 0 \Rightarrow$ coloring returned by algorithm contains *no* monochromatic k-clique
- $\Pr[X = 0] > 0$
- Consequence: At least one such coloring in the sample space! Unclear where. But we know *deterministically* that it exists!

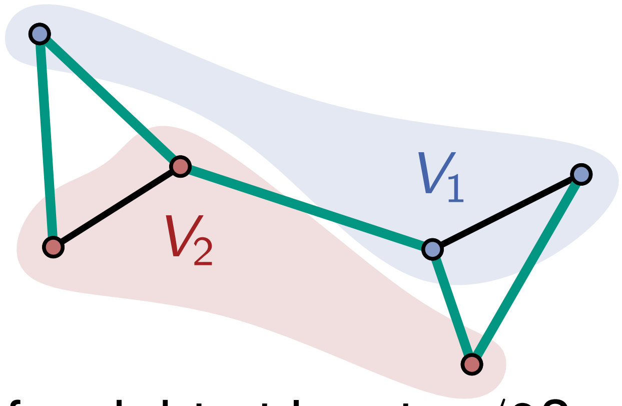


No need to actually run the algorithm to find it!

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process. (pioneered by Paul Erdős)

Application: Cuts

Recap



- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into V_1, V_2 s.t. $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$
- *Cut-set*: set of edges with one endpoint in V_1 and the other in V_2
- *Weight*: size of the cut-set
- Question now: In a graph with m edges, does there *exist* a cut of weight at least $m/2$?

Random Process

- Add each vertex to one of the two sets with equal prob. $\frac{1}{2}$

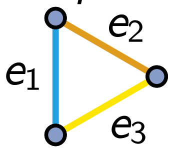
> **Probabilistic Method**: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

Positive Probability

- Consider edges $e_1, \ldots, e_m$ and let X_i be the indicator that is 1 iff e_i is in the cut-set
- $X = \sum_{i=1}^{m} X_i$ is the weight of the cut
- To show: $\Pr[X \geq \frac{m}{2}] > 0$

$\Pr[X \geq \frac{m}{2}] = \Pr\left[\sum_{i=1}^{m} X_i \geq \frac{m}{2}\right] = ???$

- Depends on the graph?
- The X_i are not even independent...



$X_2 = X_3 = 1 \Rightarrow X_1 = 1$

Probabilistic Method: The Expectation Argument

Theorem: Let X be a random variable taking values in a set S. Then, $\Pr[X \geq \mathbb{E}[X]] > 0$ and $\Pr[X \leq \mathbb{E}[X]] > 0$.

- There always exists at least one sample that yields $X \geq \mathbb{E}[X]$ ($X \leq \mathbb{E}[X]$)

Proof ($\Pr[X \geq \mathbb{E}[X]] > 0$, the other works analogous)

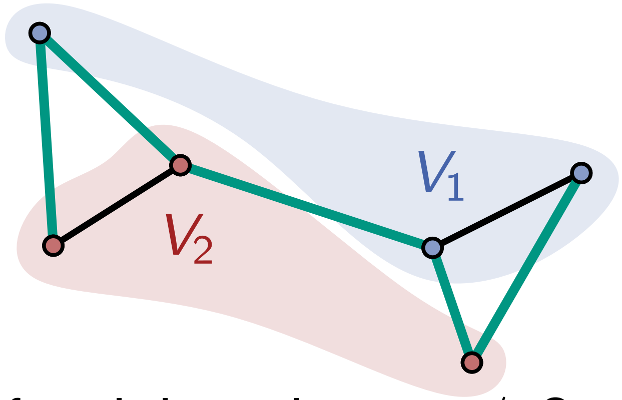
- Towards a contradiction assume $\Pr[X \geq \mathbb{E}[X]] = 0$

$$\begin{aligned}
\mathbb{E}[X] = \sum_{x \in S} x \cdot \Pr[X = x] &= \sum_{x \in S, x < \mathbb{E}[X]} x \cdot \Pr[X = x] \\
&\lightning < \sum_{x \in S, x < \mathbb{E}[X]} \mathbb{E}[X] \cdot \Pr[X = x] \\
&= \mathbb{E}[X] \cdot \sum_{x \in S, x < \mathbb{E}[X]} \Pr[X = x] \\
&\leq \mathbb{E}[X]
\end{aligned}$$

Application: Cuts – Second Try

Recap

- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into V_1, V_2 s.t. $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$
- *Cut-set*: set of edges with one endpoint in V_1 and the other in V_2
- *Weight*: size of the cut-set
- Question now: In a graph with m edges, does there *exist* a cut of weight at least $m/2$?



Random Process

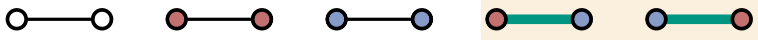
- Add each vertex to one of the two sets with equal prob. $\frac{1}{2}$

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

Positive Probability

- Consider edges $e_1, \ldots, e_m$ and let X_i be the indicator that is 1 iff e_i is in the cut-set
- $X = \sum_{i=1}^{m} X_i$ is the weight of the cut
- To show: $\Pr[X \geq \frac{m}{2}] > 0$ ✓ $\Pr[X \geq \mathbb{E}[X]] > 0$

$$\mathbb{E}[X] = \mathbb{E}[\textstyle\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} \mathbb{E}[X_i]$$
$$= m \cdot \Pr[X_i = 1] = \frac{m}{2}$$

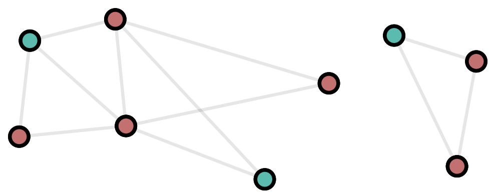


e_i					
Pr	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$ +	$\frac{1}{4}$	$= \frac{1}{2}$

Application: Independent Sets

The Problem

- Two vertices in a graph are *independent*, if they are not adjacent
- An *independent set* of a graph is a subgraph whose vertices are pairwise independent
- Let $\alpha(G)$ denote the size of a largest independent set in G (in general, determining $\alpha(G)$ is NP-complete)



$d = \frac{24}{9} \Rightarrow$ Survival rate: $\frac{3}{8}$

Theorem: Let G be a graph with n vertices and $m \geq n/2$ edges. Then $\alpha(G) \geq n^2/(4m)$.

Proof

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

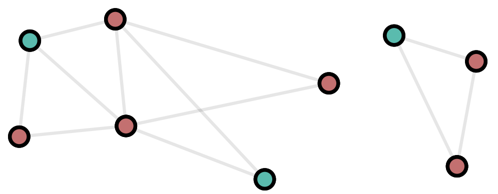
Random Process

- Let $d = 2m/n$ be the average degree of G
- Independently, delete each vertex with probability $1 - \frac{1}{d}$
- Afterwards, for each remaining edge, delete one endpoint chosen uniformly at random
- Note that the remaining vertices form an independent set

Application: Independent Sets

The Problem

- Two vertices in a graph are *independent*, if they are not adjacent
- An *independent set* of a graph is a subgraph whose vertices are pairwise independent
- Let $\alpha(G)$ denote the size of a largest independent set in G (in general, determining $\alpha(G)$ is NP-complete)



$d = \frac{24}{9} \Rightarrow$ Survival rate: $\frac{3}{8}$

Theorem: Let G be a graph with n vertices and $m \geq n/2$ edges. Then $\alpha(G) \geq n^2/(4m)$.

Proof

Positive Probability

$\mathbb{E}$**-Argument**: $\Pr[X \geq \mathbb{E}[X]] > 0$

Random Process: $d = 2m/n$
Step 1: Delete v with prob. $1 - \frac{1}{d}$
Step 2: Delete one endpoint of each e

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

- X_V: number of *vertices* that survive the first step
- X_E: number of *edges* that survive the first step
- Step 2: each of the X_E edges removes ≤ 1 vertex
- Size of resulting independent set S is $\geq X_V - X_E$
- $\Pr[|S| \geq n^2/(4m)] \geq \Pr[X_V - X_E \geq n^2/(4m)] > 0$ ✓

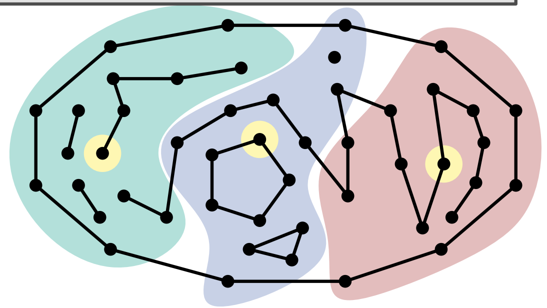
- $\mathbb{E}[X_V] = n \cdot \frac{1}{d}$ (since each vertex survives with prob. $\frac{1}{d}$)
- Edge $\{u, v\}$ survives if both u, v do
- $\mathbb{E}[X_E] = m \cdot \frac{1}{d^2} = \frac{nd}{2} \cdot \frac{1}{d^2} = \frac{n}{2d}$
- $\mathbb{E}[X_V - X_E] = \mathbb{E}[X_V] - \mathbb{E}[X_E] = \frac{n}{d} - \frac{n}{2d}$
  $= \frac{n}{2d} = \frac{n}{2(2m/n)} = n^2/(4m)$

Application: Dependent Independent Set

Theorem: Let $G = (V, E)$ be a graph with max-degree Δ. For any partition $V_1 \cup \ldots \cup V_t = V$ such that $|V_i| \geq 8\Delta$, there exists an independent set containing one vertex from each V_i.

Proof

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.



$\Delta = 2 \;\Rightarrow 8\Delta = 16$

Random Process

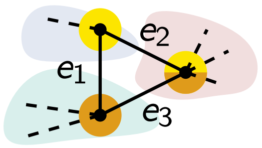
- Assume $|V_i| = k = 8\Delta$ for all i (otherwise remove vertices from too large V_i)
- Let S be the set obtained by independently choosing one vertex uniformly at random from each V_i

Positive Probability

- To show: Pr["S independent"] > 0
- S is independent iff no edge $e = \{u, v\}$ has $\overbrace{e \subseteq S}^{\text{(both endpoints in } S)}$, Let A_e be the event that $e \subseteq S$

$$\Pr[\text{“}S\text{ independent”}] = \Pr\left[\bigcap_{e\in E} \neg A_e\right] \cancel{=} \prod_{e\in E} \Pr[\neg A_e] = \prod_{e\in E}(1 - \underbrace{\Pr[A_e]}_{\leq \frac{1}{k}\cdot\frac{1}{k}}) \geq \prod_{e\in E}\left(1 - \frac{1}{k^2}\right) \underset{k>1}{>} 0 \checkmark$$

The events are not independent!



$\Pr[A_{e_1}] = \frac{1}{k^2}$

$\Pr[A_{e_1} \mid A_{e_2} \cap A_{e_3}] = 1$ The probability of an event is affected by the outcomes of other events. Dependence...

To be or not to be... independent

Karlsruhe Institute of Technology

Independence

Definition: Event A is **independent of an event** B if $\Pr[A \mid B] = \Pr[A]$. $(\Pr[A \cap B] = \Pr[A]\Pr[B])$

Definition: Event A is **independent of a set of events** $\mathcal{E}$ if for all subsets $\mathcal{E}' = \{B_1, B_2, \dots, B_k\} \subseteq \mathcal{E}$ we have $\Pr[A \mid \bigcap_{i \in [k]} B_i] = \Pr[A]$.

Example

- Triangle, independently color each vertex red/blue with prob. $\frac{1}{2}$
- Let A_{ij} for $i < j$ be the event that i and j have the same color
- $A = A_{12}$, $B = A_{23}$:

  $\Pr[A_{12}] = \boxed{\frac{1}{2}}$

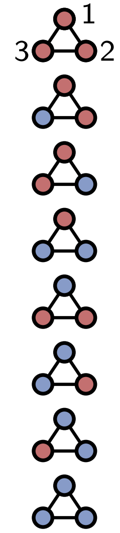
  $\Pr[A_{12} \mid A_{23}] = \frac{\Pr[A_{12} \cap A_{23}]}{\Pr[A_{23}]} = \frac{1/4}{1/2} = \frac{1}{2}$

  (same holds for all choices of A and B)
- All A_{ij} are *pairwise* independent

- $A = A_{12}$, $\mathcal{E} = \{A_{13}, A_{23}\}$:

  $\Pr[A_{12} \mid A_{13} \cap A_{23}]$

  $= \frac{\Pr[A_{12} \cap A_{13} \cap A_{23}]}{\Pr[A_{13} \cap A_{23}]} = \frac{1/4}{1/4} = \boxed{1}$
- A_{ij} not independent of the other events



Pr	Graph	1	2	3	A_{12}	A_{13}	A_{23}
$\frac{1}{8}$		red	red	red	✓	✓	✓
$\frac{1}{8}$		red	red	blue	✓	✗	✗
$\frac{1}{8}$		red	blue	red	✗	✓	✗
$\frac{1}{8}$		red	blue	blue	✗	✗	✓
$\frac{1}{8}$		blue	red	red	✗	✗	✓
$\frac{1}{8}$		blue	red	blue	✗	✓	✗
$\frac{1}{8}$		blue	blue	red	✓	✗	✗
$\frac{1}{8}$		blue	blue	blue	✓	✓	✓

Lovász Local Lemma (LLL)

Theorem: Let $E_1, \dots, E_n$ be events such that each E_i for $i \in [n]$ is independent of all but at most $d > 0$ of the other events. Let $p = \max_{i\in[n]} \Pr[E_i]$. If $4dp \le 1$, then $\Pr[\bigcap_{i\in[n]} \neg E_i] > 0$.

- If $d = 0$, everything is independent and we can just compute the probability as the product
- For each $i \in [n]$ let $D_i \subseteq [n]$ be the such that E_i is independent of $\{E_1, \dots, E_n\} \setminus (\bigcup_{j\in\{i\}\cup D_i} E_j)$, then $|D(i)| \le d$.

(Remove events defined by D_i to make E_i independent of the rest.)

Proof

Notation: For $S \subseteq [n]$ write $\mathcal{I}(S) = \bigcap_{i\in S} \neg E_i$

Conditional Probability: $\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B]$

$$\underbrace{\Pr[\overbrace{\textstyle\bigcap_{i\in[n]} \neg E_i}^{\mathcal{I}([n])}]} \overset{\text{“Chain Rule”}}{=} \prod_{i\in[n]} \Pr[\neg E_i \mid \mathcal{I}([i-1])]$$

$$= \Pr[\neg E_n \cap (\neg E_{n-1} \cap \neg E_{n-2} \cap \dots \cap \neg E_1)]$$

$$= \Pr[\neg E_n \mid (\neg E_{n-1} \cap \neg E_{n-2} \cap \dots \cap \neg E_1)] \cdot \Pr[(\neg E_{n-1} \cap \neg E_{n-2} \cap \dots \cap \neg E_1)]$$

$$= \Pr[\neg E_n \mid \underbrace{(\neg E_{n-1} \cap \neg E_{n-2} \cap \dots \cap \neg E_1)}_{\mathcal{I}([n-1])}] \cdot \Pr[\neg E_{n-1} \mid \underbrace{(\neg E_{n-2} \cap \dots \cap \neg E_1)}_{\mathcal{I}([n-2])}] \cdot \Pr[(\neg E_{n-2} \cap \dots \cap \neg E_1)]$$

Lovász Local Lemma (LLL)

Karlsruhe Institute of Technology

Theorem: Let $E_1, \dots, E_n$ be events such that each E_i for $i \in [n]$ is independent of all but at most $d > 0$ of the other events. Let $p = \max_{i \in [n]} \Pr[E_i]$. If $4dp \le 1$, then $\Pr[\bigcap_{i\in[n]} \neg E_i] > 0$.

- If $d = 0$, everything is independent and we can just compute the probability as the product
- For each $i \in [n]$ let $D_i \subseteq [n]$ be the such that E_i is independent of $\{E_1, \dots, E_n\} \setminus (\bigcup_{j \in \{i\} \cup D_i} E_j)$, then $|D(i)| \le d$.

(Remove events defined by D_i to make E_i independent of the rest.)

Proof

$$\Pr[\overbrace{\textstyle\bigcap_{i\in[n]} \neg E_i}^{\mathcal{I}([n])}] \overset{\text{“Chain Rule”}}{=} \prod_{i\in[n]} \Pr[\neg E_i \mid \mathcal{I}([i-1])]$$
$$= \prod_{i\in[n]} (1 - \underbrace{\Pr[E_i \mid \mathcal{I}([i-1])]}_{\text{Claim: } \le 2p})$$
$$\ge \prod_{i\in[n]} (1-2p)$$
$$\ge \prod_{i\in[n]} 1/2$$

Notation: For $S \subseteq [n]$ write $\mathcal{I}(S) = \bigcap_{i\in S} \neg E_i$

Conditional Probability: $\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B]$

- Since $d > 0$ and $4dp \le 1$, we have $4p \le 1$ and thus $2p \le 1/2$

LLL – Proof of Claim

Claim: For all $S_i \subseteq \{1, ..., n\} \setminus \{i\}$, $\Pr[E_i \mid \mathcal{I}(S_i)] \leq 2p$.

Proof (via induction over the size $s = |S_i|$)

Start: $s = 0 \rightarrow S_i = \emptyset \rightarrow \Pr[E_i \mid \mathcal{I}(S_i)] = \Pr[E_i] \leq p \leq 2p$ ✓

Step: $s > 0$

- Case 1: $D_i' = S_i \cap D_i = \emptyset$
  - E_i is independent of $\{E_j \mid j \in S_i\} \rightarrow \Pr[E_i \mid \mathcal{I}(S_i)] = \Pr[E_i] \leq p \leq 2p$ ✓
- Case 2: $D_i' = S_i \cap D_i \neq \emptyset$

$$\Pr[E_i \mid \mathcal{I}(S_i)] = \frac{\Pr[E_i \cap \mathcal{I}(S_i)]}{\Pr[\mathcal{I}(S_i)]} = \frac{\Pr[E_i \cap \mathcal{I}(D_i') \cap \mathcal{I}(S_i \setminus D_i')]}{\Pr[\mathcal{I}(S_i)]} = \frac{\Pr[E_i \cap \mathcal{I}(D_i') \cap \mathcal{I}(S_i \setminus D_i')]}{\Pr[\mathcal{I}(D_i') \cap \mathcal{I}(S_i \setminus D_i')]}$$

$$\mathcal{I}(S_i) = \bigcap_{j \in S_i} \neg E_j$$

$$S_i = (S_i \setminus D_i') \cup D_i' \quad \rightarrow \quad = \bigcap_{j \in S_i \setminus D_i'} \neg E_j \cap \bigcap_{j \in D_i'} \neg E_j$$

$$= \mathcal{I}(S_i \setminus D_i') \cap \mathcal{I}(D_i')$$

LLL: Events $E_1, ..., E_n$
- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, ..., E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Notation: For $S \subseteq [n]$ write $\mathcal{I}(S) = \bigcap_{i \in S} \neg E_i$

Conditional Probability: $\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B]$

LLL – Proof of Claim

Karlsruhe Institute of Technology

Claim: For all $S_i \subseteq \{1, ..., n\} \setminus \{i\}$, $\Pr[E_i \mid \mathcal{I}(S_i)] \leq 2p$.

Proof (via induction over the size $s = |S_i|$)

Start: $s = 0 \rightarrow S_i = \emptyset \rightarrow \Pr[E_i \mid \mathcal{I}(S_i)] = \Pr[E_i] \leq p \leq 2p$ ✓

Step: $s > 0$

- Case 1: $D_i' = S_i \cap D_i = \emptyset$
  - E_i is independent of $\{E_j \mid j \in S_i\} \rightarrow \Pr[E_i \mid \mathcal{I}(S_i)] = \Pr[E_i] \leq p \leq 2p$ ✓
- Case 2: $D_i' = S_i \cap D_i \neq \emptyset$

$$\Pr[E_i \mid \mathcal{I}(S_i)] = \frac{\Pr[E_i \cap \mathcal{I}(S_i)]}{\Pr[\mathcal{I}(S_i)]} = \frac{\Pr[E_i \cap \mathcal{I}(D_i') \cap \mathcal{I}(S_i \setminus D_i')]}{\Pr[\mathcal{I}(S_i)]} = \frac{\Pr[E_i \cap \mathcal{I}(D_i') \cap \mathcal{I}(S_i \setminus D_i')]}{\Pr[\mathcal{I}(D_i') \cap \mathcal{I}(S_i \setminus D_i')]}$$

$$= \frac{\Pr[E_i \cap \mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')] \cdot \cancel{\Pr[\mathcal{I}(S_i \setminus D_i')]}}{\Pr[\mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')] \cdot \cancel{\Pr[\mathcal{I}(S_i \setminus D_i')]}} \leq \frac{\Pr[E_i \mid \mathcal{I}(S_i \setminus D_i')]}{\Pr[\mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')]}$$

$$= \frac{\Pr[E_i]}{\Pr[\mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')]} \leq \frac{p}{\Pr[\mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')]} \Big\} \text{remains to show} \geq \tfrac{1}{2}$$

Removing the D_i' makes E_i independent of the remaining events.

LLL: Events $E_1, ..., E_n$

- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, ..., E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Notation: For $S \subseteq [n]$ write $\mathcal{I}(S) = \bigcap_{i \in S} \neg E_i$

Conditional Probability: $\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B]$

$\Pr[A \cap B] \leq \Pr[A]$

LLL – Proof of Claim

Karlsruhe Institute of Technology

Claim: For all $S_i \subseteq \{1, ..., n\} \setminus \{i\}$, $\Pr[E_i \mid \mathcal{I}(S_i)] \leq 2p$.

Proof (via induction over the size $s = |S_i|$)

Start: $s = 0 \rightarrow S_i = \emptyset \rightarrow \Pr[E_i \mid \mathcal{I}(S_i)] = \Pr[E_i] \leq p \leq 2p$ ✓

Step: $s > 0$

- Case 1: $D_i' = S_i \cap D_i = \emptyset$
  - E_i is independent of $\{E_j \mid j \in S_i\} \rightarrow \Pr[E_i \mid \mathcal{I}(S_i)] = \Pr[E_i] \leq p \leq 2p$ ✓
- Case 2: $D_i' = S_i \cap D_i \neq \emptyset$

$$\Pr[E_i \mid \mathcal{I}(S_i)] \leq \frac{p}{\Pr[\mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')]} \Big\}\text{remains to show} \geq \tfrac{1}{2}$$

$$\begin{aligned}
\Pr[\mathcal{I}(D_i') \mid \mathcal{I}(S_i \setminus D_i')] &= \Pr[\textstyle\bigcap_{j \in D_i'} \neg E_j \mid \mathcal{I}(S_i \setminus D_i')] \\
&= 1 - \Pr[\textstyle\bigcup_{j \in D_i'} E_j \mid \mathcal{I}(S_i \setminus D_i')] \\
\text{(via union bound)} \quad &\geq 1 - \textstyle\sum_{j \in D_i'} \Pr[E_j \mid \mathcal{I}(\underbrace{S_i \setminus D_i'})] \\
|D_i'| \leq |D_i| \quad &\geq 1 - \textstyle\sum_{j \in D_i'} 2p \geq 1 - d \cdot 2p \geq \tfrac{1}{2} \checkmark
\end{aligned}$$

$\subsetneq S_i$, since $S_i \cap D_i' \neq \emptyset$

$\Rightarrow |S_i \setminus D_i'| < s$ and we can apply induction hypothesis

LLL: Events $E_1, ..., E_n$
- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, ..., E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Notation: For $S \subseteq [n]$ write $\mathcal{I}(S) = \bigcap_{i \in S} \neg E_i$

Conditional Probability: $\Pr[A \cap B] = \Pr[A \mid B] \cdot \Pr[B]$

$\Pr[A \cap B] \leq \Pr[A]$

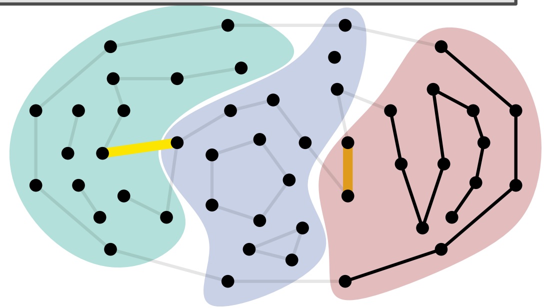
$\Pr[\neg A \cap \neg B] = \Pr[\neg(A \cup B)]$

Application: Dependent Independent Set (2nd Try)

Theorem: Let $G=(V,E)$ be a graph with max-degree Δ. For any partition $V_1 \cup \ldots \cup V_t = V$ such that $|V_i| \geq 8\Delta$, there exists an independent set containing one vertex from each V_i.

Proof

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.



Random Process

- Assume $|V_i| = k = 8\Delta$ for all i (otherwise remove vertices from too large V_i)
- Obtain S by ind. choosing one vertex unif. at random from each V_i

Positive Probability

- To show: $\Pr[\text{“}S \text{ independent”}] > 0$
- S is independent iff no edge $e = \{u, v\}$ has $\overbrace{e \subseteq S}^{\text{(both endpoints in } S)}$,

  $u \downarrow V_i$, $v \downarrow V_j$

  Let A_e be the event that $e \subseteq S$

  $\Pr[\text{“}S \text{ independent”}] = \Pr[\bigcap_{e \in E} \neg A_e]$

  $\Pr[A_e] \leq \frac{1}{k^2} =: p$

  $D_e = \{A_{e'} \mid e' \cap (V_i \cup V_j) \neq \emptyset\}$

This is like isolating V_i, V_j from the remainder of the graph

No matter the outcome of A_f for $f \in E \setminus D_e$, the probability for a node in V_i or V_j to be chosen remains the same $\Rightarrow A_e$ is independent of all events but D_e

LLL: Events $E_1, \ldots, E_n$

- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, \ldots, E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

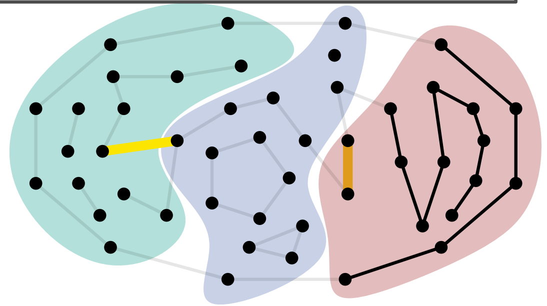
Then, $Pr[\bigcap_{i \in [n]} \neg E_i] > 0$.

Application: Dependent Independent Set (2nd Try)

Theorem: Let $G = (V, E)$ be a graph with max-degree Δ. For any partition $V_1 \cup \ldots \cup V_t = V$ such that $|V_i| \geq 8\Delta$, there exists an independent set containing one vertex from each V_i.

Proof

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.



Random Process

- Assume $|V_i| = k = 8\Delta$ for all i (otherwise remove vertices from too large V_i)
- Obtain S by ind. choosing one vertex unif. at random from each V_i

Positive Probability

- To show: $\Pr[\text{“}S \text{ independent”}] > 0$
- S is independent iff no edge $e = \{\underset{V_i}{u}, \underset{V_j}{v}\}$ has $\overbrace{e \subseteq S}^{\text{(both endpoints in } S\text{)}}$,

  Let A_e be the event that $e \subseteq S$

  $\Pr[\text{“}S \text{ independent”}] = \Pr[\bigcap_{e \in E} \neg A_e] > 0$ ✓

  $\Pr[A_e] \leq \frac{1}{k^2} =: p$

  $D_e = \{A_{e'} \mid e' \cap (V_i \cup V_j) \neq \emptyset\} \rightarrow |D_e| \leq \underset{|V_i|}{k}\Delta + \underset{|V_j|}{k}\Delta \leq 2k\Delta =: d$

LLL: Events $E_1, \ldots, E_n$

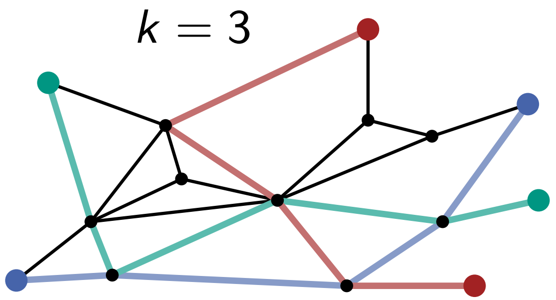
- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, \ldots, E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Then, $Pr[\bigcap_{i \in [n]} \neg E_i] > 0$.

$$4dp = 4 \cdot 2k\Delta \cdot \frac{1}{k^2} = \frac{8\Delta}{k} = 1$$

Application: Independent Paths

- Given a network and k vertex pairs that want to communicate
- Each $i \in [k]$ has a set S_i of candidate communication paths
- Does there exist a choice of paths (one P_i from each S_i) that are pairwise edge-disjoint? (NP-complete to decide)



Theorem: Let $m = \min_{i \in [k]}\{|S_i|\}$. Then, there exists a valid choice if any path in S_i shares edges with at most $\ell \leq m/(8k)$ paths in S_j for $i \neq j$.

Proof

Random Process: Ind., unif. at random choose P_i from S_i

Positive Probability

- Let E_{ij} be the event that P_i and P_j share an edge

$\Pr[\bigcap_{i<j} \neg E_{ij}] \overset{?}{>} 0$

$\Pr[E_{ij}] \leq \frac{\ell}{m} =: p$

- E_{ij} independent of every other event but not of *all* others
- If $E_{i1}, \ldots, E_{i\ell}$ occur, then $Pr[E_{ij}] = 0$ for $j > \ell$

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

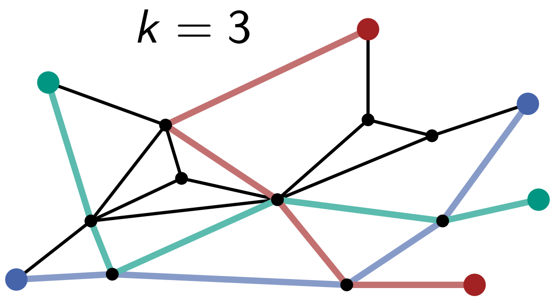
LLL: Events $E_1, \ldots, E_n$

- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, \ldots, E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Then, $Pr[\bigcap_{i \in [n]} \neg E_i] > 0$.

Application: Independent Paths

- Given a network and k vertex pairs that want to communicate
- Each $i \in [k]$ has a set S_i of candidate communication paths
- Does there exist a choice of paths (one P_i from each S_i) that are pairwise edge-disjoint? (NP-complete to decide)



Theorem: Let $m = \min_{i \in [k]} \{|S_i|\}$. Then, there exists a valid choice if any path in S_i shares edges with at most $\ell \leq m/(8k)$ paths in S_j for $i \neq j$.

Proof

Random Process : Ind., unif. at random choose P_i from S_i

Positive Probability

- Let E_{ij} be the event that P_i and P_j share an edge

$$\Pr[\bigcap_{i<j} \neg E_{ij}] \overset{?}{>} 0$$

$$\Pr[E_{ij}] \leq \frac{\ell}{m} =: p$$

$$D_{ij} = \{E_{st} \mid \{s,t\} \cap \{i,j\} \neq \emptyset\}$$

Removing D_{ij} is discarding all events that could tell us something about whether P_i and P_j can intersect

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

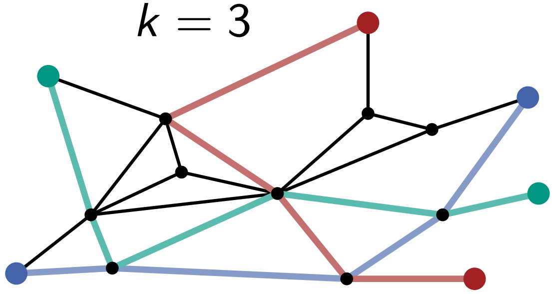
LLL: Events $E_1, \ldots, E_n$

- $p = \max_{i \in [n]} \Pr[E_i]$
- E_i independent of $\{E_1, \ldots, E_n\} \setminus \bigcup_{j \in \{i\} \cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Then, $Pr[\bigcap_{i \in [n]} \neg E_i] > 0$.

Application: Independent Paths

- Given a network and k vertex pairs that want to communicate
- Each $i \in [k]$ has a set S_i of candidate communication paths
- Does there exist a choice of paths (one P_i from each S_i) that are pairwise edge-disjoint? (NP-complete to decide)



Theorem: Let $m = \min_{i\in[k]}\{|S_i|\}$. Then, there exists a valid choice if any path in S_i shares edges with at most $\ell \leq m/(8k)$ paths in S_j for $i \neq j$.

Proof

Random Process: Ind., unif. at random choose P_i from S_i

Positive Probability

- Let E_{ij} be the event that P_i and P_j share an edge

$\Pr[\bigcap_{i<j} \neg E_{ij}] > 0$ ✓

$\Pr[E_{ij}] \leq \frac{\ell}{m} =: p$

$D_{ij} = \{E_{st} \mid \{s,t\} \cap \{i,j\} \neq \emptyset\}$

$|D_{ij}| = (k-1) + (k-1) - 1 < 2k =: d$

$4dp = 4 \cdot 2k \cdot \frac{\ell}{m} = \ell \cdot \frac{8k}{m} \leq 1$

Probabilistic Method: Show that something exists by proving that it has a *positive* probability of occuring from a random process.

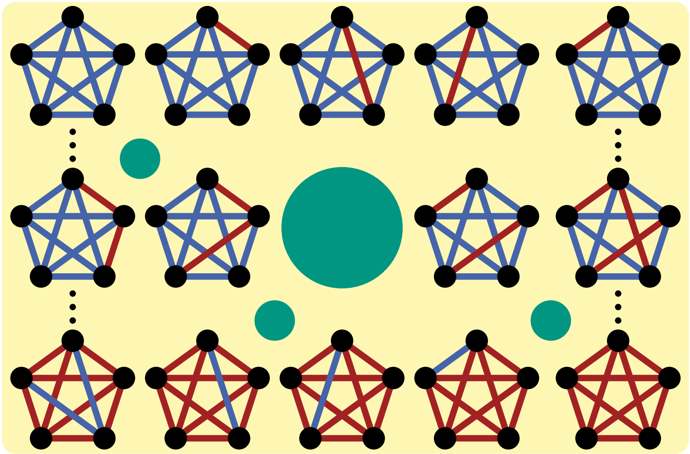
LLL: Events $E_1, \ldots, E_n$

- $p = \max_{i\in[n]} \Pr[E_i]$
- E_i independent of $\{E_1, \ldots, E_n\} \setminus \bigcup_{j\in\{i\}\cup D_i} E_j$ with $|D_i| \leq d$
- $4dp \leq 1$

Then, $Pr[\bigcap_{i\in[n]} \neg E_i] > 0$.

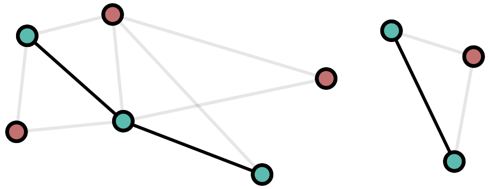
Conclusion

Probabilistic Method
- Show that something exists *deterministically*, by showing that it occurs with positive probability from a random process
- Reasoning: At least one object in the sample space has the desired property



Expectation Argument
- Useful tool when applying probabilistic method
- $\Pr[X \geq \mathbb{E}[X]] > 0$ and $\Pr[X \leq \mathbb{E}[X]] > 0$.

Sample via Modification
- Example Vertex Cover: remove vertices/edges at random



Lovász Local Lemma
- Show that something exists by showing that all events that prevent its existence do not occur, with positive probability
- Lemma works as long as there are not too many dependencies

Probability & Computing

Continuous Probability Spaces & Random Geometric Graphs

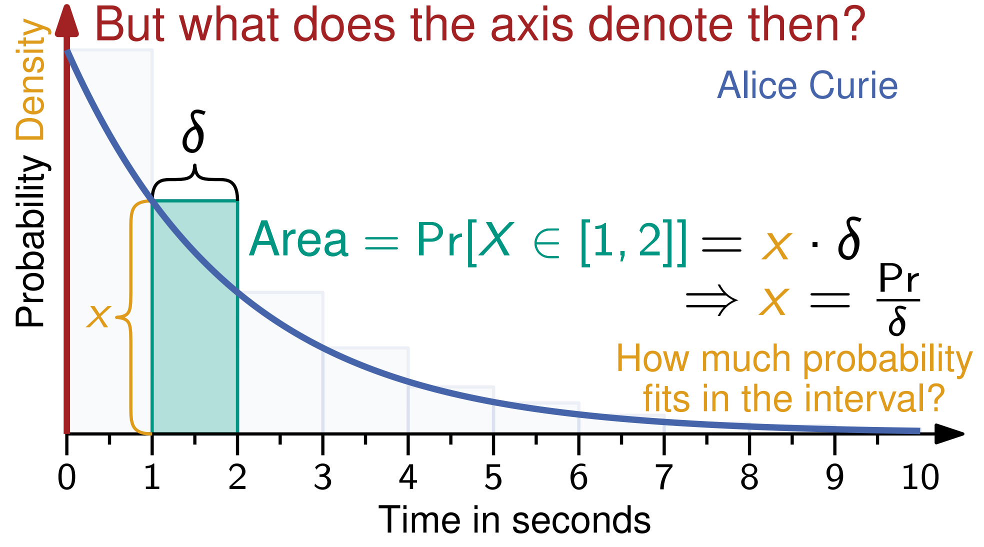
Motivation – Radioactive Decay

- Two physicists study radioactive material that emits particles every now and then
- Both compete to get the most accurate model describing the emission
- "We could do this forever!" Could they really?
- They measure with infinite precision...
  - What is $Pr[X = 2.71828182846]$?
  - What is $Pr[X = 2.7182818284\mathbf{7}]$?

  $\Big\} > 0?$ Emission could happen at any time...
  - But then the "sum" over uncountably infinite non-zero values is ∞ This is not a probability distribution!
- For continuous spaces we need to adjust how we measure probabilities

  We assign probabilities to *intervals* instead of individual values!

  The probability is the *area* of the bar, *not* the height
- As bars get thinner, areas (probabilities) decrease
- We describe distributions using probability density functions



youtube.com/watch?v=ZA4JkHKZM50

Working in Continuous Probability Spaces

Discrete Random Variable X

- Cumulative distribution function
  $F_X(x) = \Pr[X \leq x] = \sum_{y \leq x} f_X(y)$
- Probability mass function
  $f_X(x) = \Pr[X = x] \geq 0$ $\quad \sum_x \Pr[X = x] = 1$
- Expectation
  $\mathbb{E}[X] = \sum_x x \cdot \Pr[X = x]$

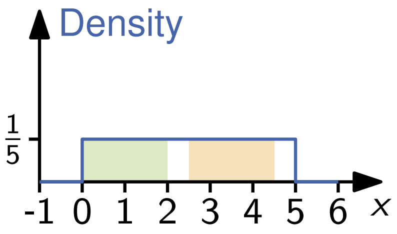
Continuous Random Variable X

- Cumulative distribution function
  $F_X(x) = \Pr[X \leq x] = \int_{-\infty}^{x} f_X(y)\mathrm{d}y$
- Probability density function
  $f_X(x) \geq 0$ $\quad \int_{-\infty}^{\infty} f_X(x)\mathrm{d}x = 1$
- Expectation
  $\mathbb{E}[X] = \int x \cdot f_X(x)\mathrm{d}x$

Example: Uniform Distribution $\longrightarrow$ Over $[0, 5]$

- You build a fence that is at least 2m tall at each point
- In the hardware store they have 5m planks
- The staff member cutting your planks wears hearing protection and cuts uniformly at random
- What is the probability that you get two $\geq$ 2m boards out of one 5m plank?

$$f_X(x) = \begin{cases} \frac{1}{5}, \text{ if } x \in [0, 5] \\ 0, \text{ o.w.} \end{cases}$$



$$\int_{-\infty}^{\infty} f_X(x)\mathrm{d}x = \int_0^5 \frac{1}{5}\mathrm{d}x = \left[\frac{x}{5}\right]_0^5 = 1 \checkmark$$

$$\int_a^b f_X(x)\mathrm{d}x = \left[\frac{x}{5}\right]_a^b = \frac{1}{5}(b-a) \checkmark$$

for $a \leq b \in [0, 5]$

Working in Continuous Probability Spaces

Discrete Random Variable X

- Cumulative distribution function
  $F_X(x) = \Pr[X \leq x] = \sum_{y \leq x} f_X(y)$
- Probability mass function
  $f_X(x) = \Pr[X = x] \geq 0$ $\qquad \sum_x \Pr[X=x]=1$
- Expectation
  $\mathbb{E}[X] = \sum_x x \cdot \Pr[X = x]$

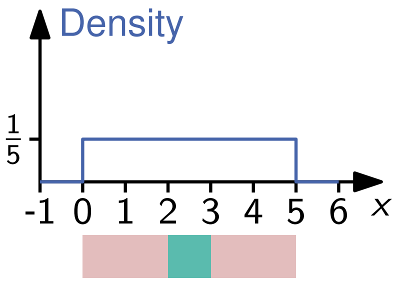
Continuous Random Variable X

- Cumulative distribution function
  $F_X(x) = \Pr[X \leq x] = \int_{-\infty}^{x} f_X(y)\mathrm{d}y$
- Probability density function
  $f_X(x) \geq 0$ $\qquad \int_{-\infty}^{\infty} f_X(x)\mathrm{d}x = 1$
- Expectation
  $\mathbb{E}[X] = \int x \cdot f_X(x)\mathrm{d}x$

Example: Uniform Distribution → Over $[0, 5]$

- You build a fence that is at least 2m tall at each point
- In the hardware store they have 5m planks
- The staff member cutting your planks wears hearing protection and cuts uniformly at random
- What is the probability that you get two $\geq$ 2m boards out of one 5m plank?

$$f_X(x) = \begin{cases} \frac{1}{5}, \text{if } x \in [0, 5] \\ 0, \text{o.w.} \end{cases}$$



$$\Pr[X \in [2, 3]] = \Pr[X \leq 3] - \Pr[X \leq 2]$$
$$= \int_0^3 \tfrac{1}{5}\mathrm{d}x - \int_0^2 \tfrac{1}{5}\mathrm{d}x$$
$$= \left[\tfrac{x}{5}\right]_0^3 - \left[\tfrac{x}{5}\right]_0^2 = \tfrac{3}{5} - \tfrac{2}{5} = \tfrac{1}{5} \checkmark$$

Working in Continuous Probability Spaces

Discrete Random Variable X

- Cumulative distribution function

  $F_X(x) = \Pr[X \le x] = \sum_{y \le x} f_X(y)$
- Probability mass function

  $f_X(x) = \Pr[X = x] \ge 0$ $\quad \sum_x \Pr[X = x] = 1$
- Expectation

  $\mathbb{E}[X] = \sum_x x \cdot \Pr[X = x]$

Continuous Random Variable X

- Cumulative distribution function

  $F_X(x) = \Pr[X \le x] = \int_{-\infty}^{x} f_X(y)\mathrm{d}y$
- Probability density function

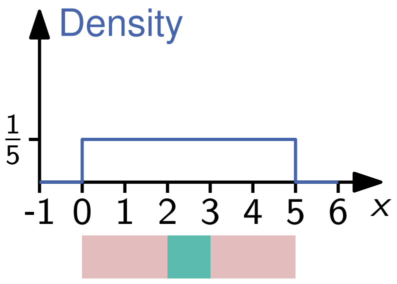
  $f_X(x) \ge 0$ $\quad \int_{-\infty}^{\infty} f_X(x)\mathrm{d}x = 1$
- Expectation

  $\mathbb{E}[X] = \int x \cdot f_X(x)\mathrm{d}x$

Example: Uniform Distribution $\longrightarrow$ Over $[0, 5]$

- You build a fence that is at least 2m tall at each point
- In the hardware store they have 5m planks
- The staff member cutting your planks wears hearing protection and cuts uniformly at random
- What is the probability that you get two $\ge$ 2m boards out of one 5m plank?

$$f_X(x) = \begin{cases} \frac{1}{5}, \text{if } x \in [0, 5] \\ 0, \text{o.w.} \end{cases}$$



- In general: $X \sim \mathcal{U}([a, b])$

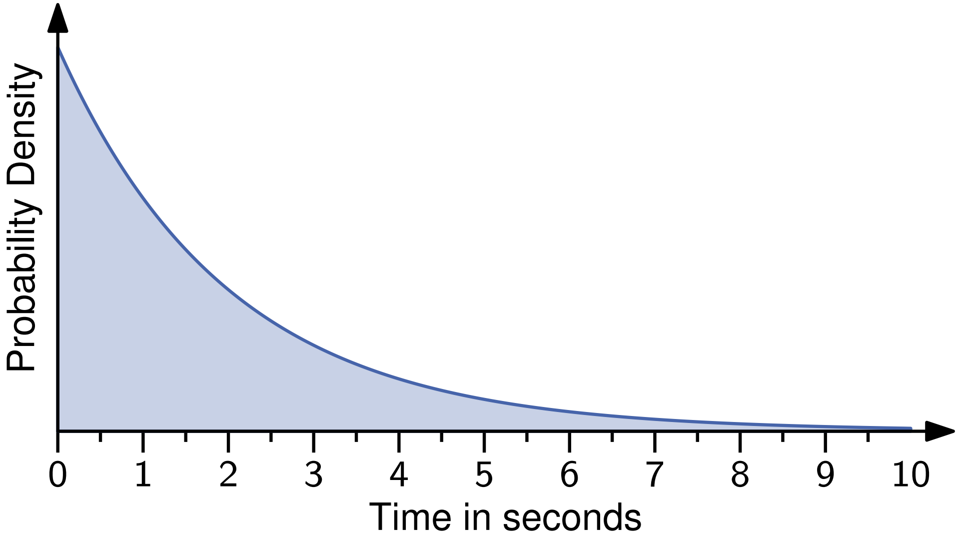
  $\Pr[X \in [c, d] \subseteq [a, b]] = \frac{d-c}{b-a}$

Example: Radioactive Decay

Exponential Distribution $X \sim \mathrm{Exp}(\lambda)$

- "Rate" parameter $\lambda > 0$
- Continuous equivalent to geometric distribution
- "Time until first success"
- Probability density function $f_X(x) = \begin{cases} \lambda e^{-\lambda x}, \text{if } x \geq 0 \\ 0, \text{o.w.} \end{cases}$
- Cumulative distribution function

  $F_X(x) = \int_{-\infty}^{x} f_X(y)\mathrm{d}y = 1 - e^{-\lambda x}$



Integration by Parts
$\int uv'\mathrm{d}x = uv - \int u'v\mathrm{d}x$

Characterization via Moments (n-th moment: $\mathbb{E}[X^n]$)

- $\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)\mathrm{d}x = \lambda \int_0^\infty xe^{-\lambda x}\mathrm{d}x = \lambda\left(\left[x \cdot \frac{1}{-\lambda}e^{-\lambda x}\right]_0^\infty - \int_0^\infty \frac{1}{-\lambda}e^{-\lambda x} \cdot 1\mathrm{d}x\right)$

$$= \cancel{\lambda}\left(\frac{1}{\cancel{\lambda}}\left[xe^{-\lambda x}\right]_\infty^0 + \frac{1}{\cancel{\lambda}}\int_0^\infty e^{-\lambda x}\mathrm{d}x\right)$$

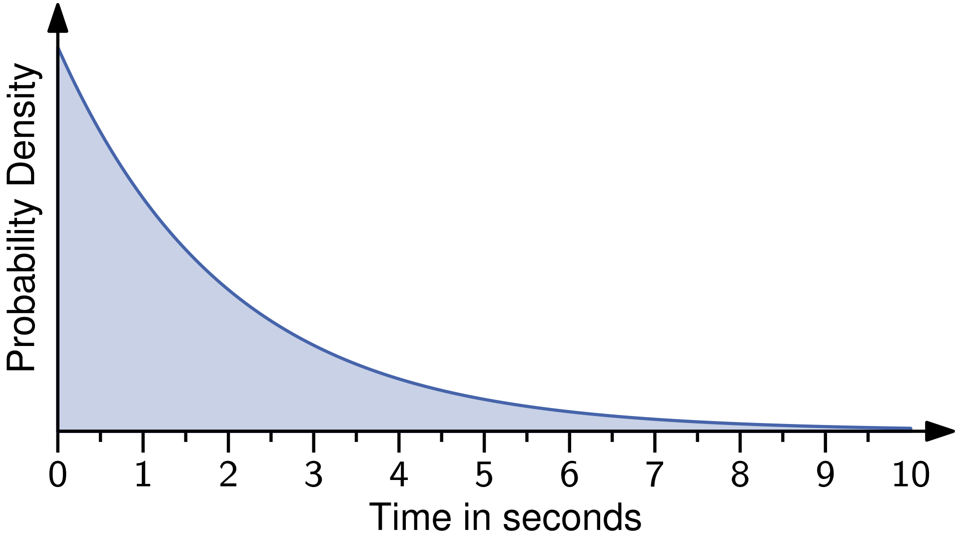
$$= 0 + 0 + \frac{1}{-\lambda}\left[e^{-\lambda x}\right]_0^\infty = \frac{1}{\lambda}\left[e^{-\lambda x}\right]_\infty^0 = \frac{1}{\lambda}[1-0] = \frac{1}{\lambda}$$

Example: Radioactive Decay

Exponential Distribution $X \sim \mathrm{Exp}(\lambda)$

- "Rate" parameter $\lambda > 0$
- Continuous equivalent to geometric distribution
- "Time until first success"
- Probability density function $f_X(x) = \begin{cases} \lambda e^{-\lambda x}, \text{ if } x \geq 0 \\ 0, \text{ o.w.} \end{cases}$
- Cumulative distribution function

  $F_X(x) = \int_{-\infty}^{x} f_X(y)\mathrm{d}y = 1 - e^{-\lambda x}$



Integration by Parts
$\int uv'\mathrm{d}x = uv - \int u'v\mathrm{d}x$

Characterization via Moments (*n*-th moment: $\mathbb{E}[X^n]$)

- $\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f_X(x)\mathrm{d}x = \lambda \int_0^{\infty} x e^{-\lambda x}\mathrm{d}x = \frac{1}{\lambda}$
- $\mathbb{E}[X^2] = \int_{-\infty}^{\infty} x^2 \cdot f_X(x)\mathrm{d}x = \lambda \int_0^{\infty} x^2 e^{-\lambda x}\mathrm{d}x$

$$= \lambda \left(\left[x^2 \frac{1}{-\lambda} e^{-\lambda x}\right]_0^{\infty} - \frac{2}{-\lambda} \overbrace{\int_0^{\infty} x \cdot e^{-\lambda x}\mathrm{d}x}^{\frac{1}{\lambda^2}} \right) = \lambda([0+0] + \frac{2}{\lambda^3}) = \frac{2}{\lambda^2}$$

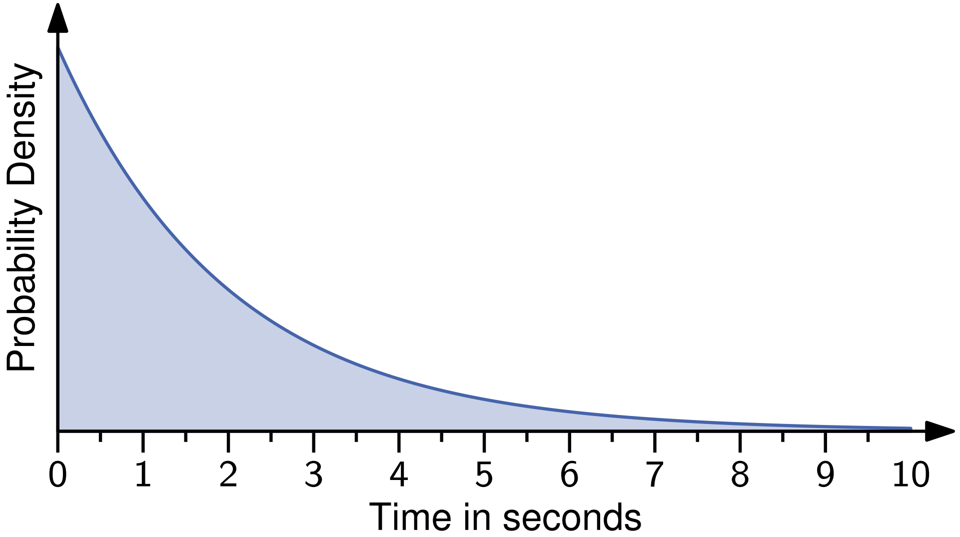
Example: Radioactive Decay

Karlsruhe Institute of Technology

Exponential Distribution $X \sim \text{Exp}(\lambda)$

- "Rate" parameter $\lambda > 0$
- Continuous equivalent to geometric distribution
- "Time until first success"
- Probability density function $f_X(x) = \begin{cases} \lambda e^{-\lambda x}, \text{if } x \geq 0 \\ 0, \text{o.w.} \end{cases}$
- Cumulative distribution function

  $F_X(x) = \int_{-\infty}^{x} f_X(y) \mathrm{d}y = 1 - e^{-\lambda x}$



Integration by Parts
$\int uv' \mathrm{d}x = uv - \int u'v \mathrm{d}x$

Characterization via Moments (*n*-th moment: $\mathbb{E}[X^n]$)

- $\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f_X(x) \mathrm{d}x = \lambda \int_0^{\infty} x e^{-\lambda x} \mathrm{d}x = \frac{1}{\lambda}$
- $\mathbb{E}[X^2] = \int_{-\infty}^{\infty} x^2 \cdot f_X(x) \mathrm{d}x = \lambda \int_0^{\infty} x^2 e^{-\lambda x} \mathrm{d}x = \frac{2}{\lambda^2}$
- $\text{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \frac{2}{\lambda^2} - (\frac{1}{\lambda})^2 = \frac{1}{\lambda^2}$

Exponential Distribution: Memorylessness

Motivation

- What is the probability of having to wait longer than an additional time $s > 0$ after already having waited time $t > 0$?

$$\Pr[X > s + t \mid X > t] = \frac{\Pr[X > s + t \wedge X > t]}{\Pr[X > t]} \qquad X > s + t \Rightarrow X > t$$

$$= \frac{\Pr[X > s + t]}{\Pr[X > t]} = \frac{1 - \Pr[X \leq s + t]}{1 - \Pr[X \leq t]}$$

$$= \frac{e^{-\lambda(s+t)}}{e^{-\lambda t}} = e^{-\lambda s} = \Pr[X > s]$$

$X \sim \mathrm{Exp}(\lambda)$

$f_X(x) = \lambda e^{-\lambda x}$

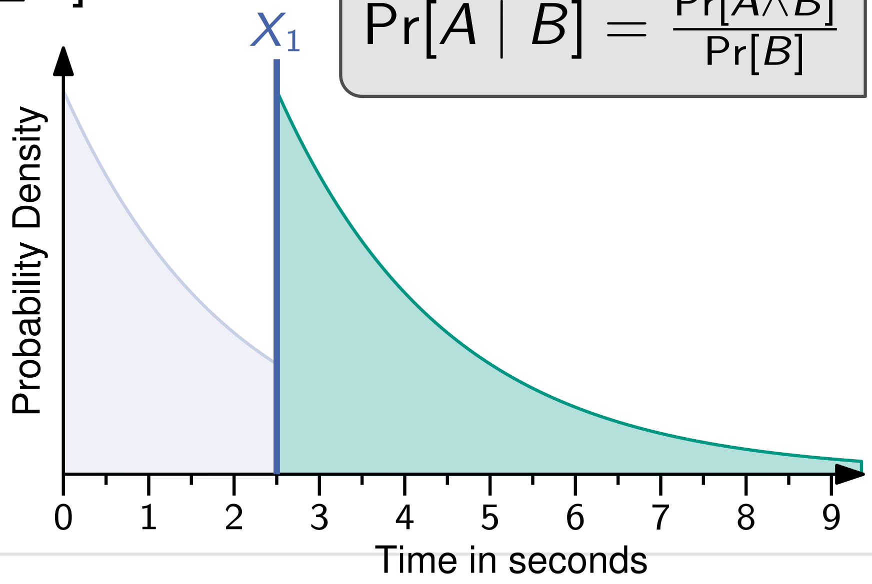
$F_X(x) = 1 - e^{-\lambda x}$

$\Pr[A \mid B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$

- No matter how long we already waited, waiting time is distributed as if we just started

Observing Multiple Particles

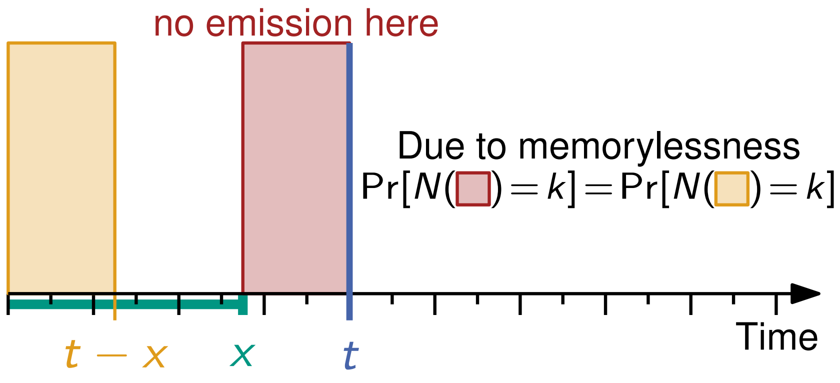
- How long do we have to wait for the second particle after having just seen the first?



Counting Decays

Motivation

- Count number of particles emitted within a given time t
- Let $X_1, X_2, X_3, \ldots \sim \mathrm{Exp}(\lambda)$ be independent waiting times
- Let $N(a, b)$ be the number of emissions in $[a, b]$
- Let $N_t = N(0, t)$ be the number of emissions until t



Specific Values

Law of Total Probability: $\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] \cdot f_X(x) \mathrm{d}x$

$\boldsymbol{X \sim \mathrm{Exp}(\lambda)}$

$f_X(x) = \lambda e^{-\lambda x} \mathbb{1}_{x \geq 0}$

$F_X(x) = 1 - e^{-\lambda x}$

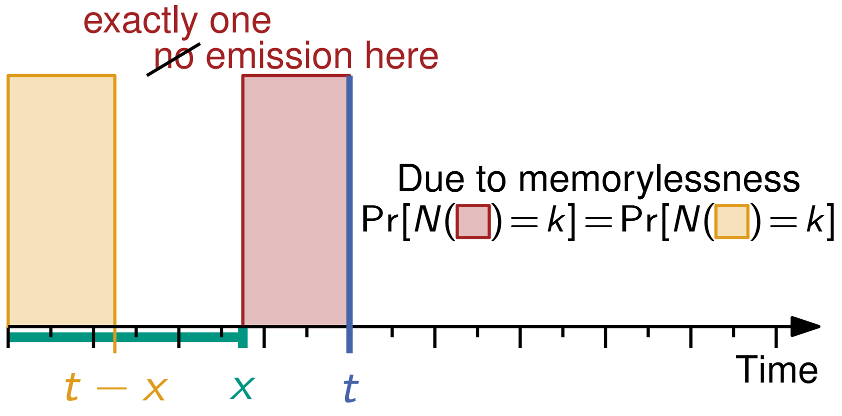
$$\Pr[N_t = 0] = e^{-\lambda t}$$

$$\begin{aligned}
\Pr[N_t = 1] &= \int_{-\infty}^{\infty} \Pr[X_1 \leq t \wedge N(x,t) = 0 \mid X_1 = x] f_{X_1}(x) \mathrm{d}x \\
&= \int_{-\infty}^{\infty} \Pr[X_1 \leq t \wedge N(x,t) = 0 \mid X_1 = x] \lambda e^{-\lambda x} \mathbb{1}_{x \geq 0} \mathrm{d}x \\
&= \int_0^t \Pr[\cancel{X_1 \leq t} \wedge N(x,t) = 0 \mid X_1 = x] \lambda e^{-\lambda x} \cancel{\mathbb{1}_{x \geq 0}} \mathrm{d}x \\
&= \int_0^t \Pr[N(x,t) = 0 \mid \cancel{X_1 = x}] \lambda e^{-\lambda x} \mathrm{d}x \\
&= \int_0^t \Pr[N(x,t) = 0] \lambda e^{-\lambda x} \mathrm{d}x = \int_0^t \overbrace{\Pr[N_{t-x} = 0]}^{e^{-\lambda(t-x)}} \lambda e^{-\lambda x} \mathrm{d}x = \int_0^t e^{-\lambda(t-x)} \cdot \lambda e^{-\lambda x} \mathrm{d}x \\
&= \lambda e^{-\lambda t} \int_0^t 1 \mathrm{d}x = \lambda t e^{-\lambda t}
\end{aligned}$$

Counting Decays

Motivation

- Count number of particles emitted within a given time t
- Let $X_1, X_2, X_3, \ldots \sim \mathrm{Exp}(\lambda)$ be independent waiting times
- Let $N(a, b)$ be the number of emissions in $[a, b]$
- Let $N_t = N(0, t)$ be the number of emissions until t



Specific Values

Law of Total Probability: $\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] \cdot f_X(x)\mathrm{d}x$

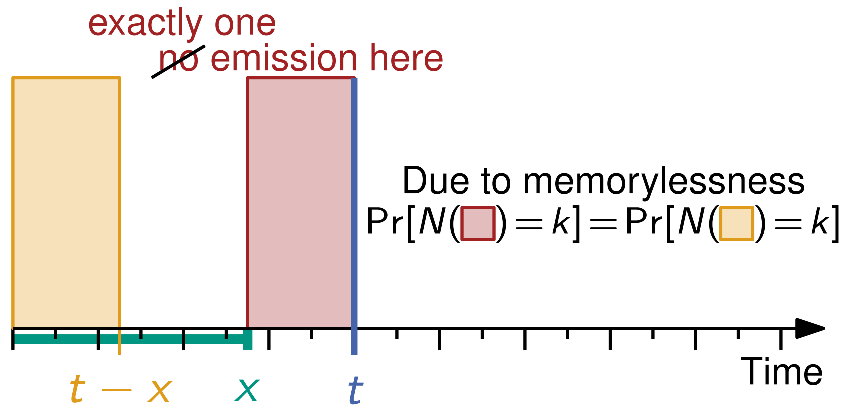
$X \sim \mathrm{Exp}(\lambda)$

$f_X(x) = \lambda e^{-\lambda x} \mathbb{1}_{x \geq 0}$

$F_X(x) = 1 - e^{-\lambda x}$

$\Pr[N_t = 0] = e^{-\lambda t}$ $\quad$ $\Pr[N_t = 1] = \lambda t e^{-\lambda t}$ $\quad$ $\Pr[N_t = 2] = \lambda^2 e^{-\lambda t} \cdot \frac{1}{2} t^2$

Counting Decays



Motivation

- Count number of particles emitted within a given time t
- Let $X_1, X_2, X_3, \ldots \sim \mathrm{Exp}(\lambda)$ be independent waiting times
- Let $N(a, b)$ be the number of emissions in $[a, b]$
- Let $N_t = N(0, t)$ be the number of emissions until t

Specific Values

Law of Total Probability: $\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] \cdot f_X(x)\mathrm{d}x$

$X \sim \mathrm{Exp}(\lambda)$

$f_X(x) = \lambda e^{-\lambda x} \mathbb{1}_{x \geq 0}$

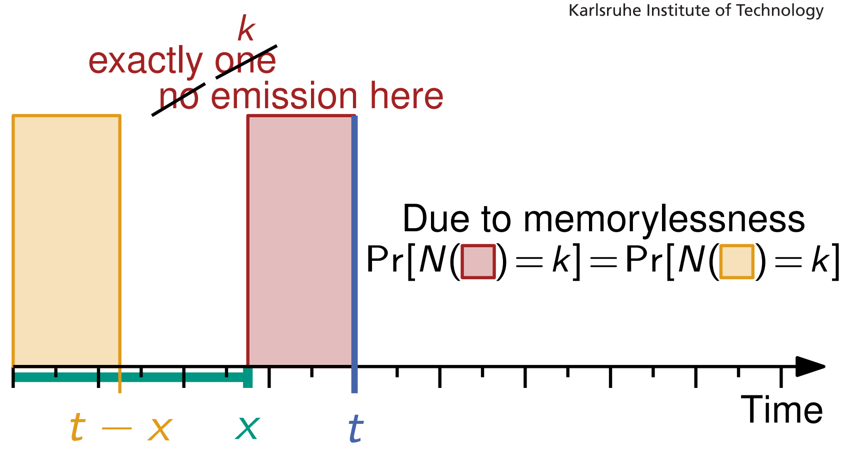
$F_X(x) = 1 - e^{-\lambda x}$

$$\Pr[N_t = 0] = \underbrace{e^{-\lambda t}}_{\frac{(\lambda t)^0 e^{-\lambda t}}{0!}} \quad \Pr[N_t = 1] = \underbrace{\lambda t e^{-\lambda t}}_{\frac{(\lambda t)^1 e^{-\lambda t}}{1!}} \quad \Pr[N_t = 2] = \underbrace{\lambda^2 e^{-\lambda t} \cdot \tfrac{1}{2} t^2}_{\frac{(\lambda t)^2 e^{-\lambda t}}{2!}}$$

Counting Decays

Motivation

- Count number of particles emitted within a given time t
- Let $X_1, X_2, X_3, \ldots \sim \mathrm{Exp}(\lambda)$ be independent waiting times
- Let $N(a, b)$ be the number of emissions in $[a, b]$
- Let $N_t = N(0, t)$ be the number of emissions until t



Law of Total Probability: $\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] \cdot f_X(x)\mathrm{d}x$

$\boldsymbol{X \sim \mathrm{Exp}(\lambda)}$

$f_X(x) = \lambda e^{-\lambda x} \mathbb{1}_{x \geq 0}$

$F_X(x) = 1 - e^{-\lambda x}$

Specific Values

$\Pr[N_t = 0] = e^{-\lambda t} \quad \Pr[N_t = 1] = \lambda t e^{-\lambda t} \quad \Pr[N_t = 2] = \lambda^2 e^{-\lambda t} \cdot \frac{1}{2} t^2$

General Form $\Pr[N_t = k] = \dfrac{(\lambda t)^k e^{-\lambda t}}{k!}$ *(proof via induction)*

$\hookrightarrow N_t \sim \mathrm{Pois}(\lambda t)$

Integration by Substitution $u = g(x)$

$$\int_a^b f(g(x))\mathrm{d}x = \int_{g(a)}^{g(b)} \frac{f(u)}{\left(\frac{\mathrm{d}g(x)}{\mathrm{d}x}\right)}\mathrm{d}u$$

$\Pr[N_t = k + 1]$

$$= \int_0^t \Pr[N_{t-x} = k] \cdot \lambda e^{-\lambda x}\mathrm{d}x$$

$$= \int_0^t \frac{(\lambda(t-x))^k e^{-\lambda(t\cancel{-x})}}{k!} \cdot \lambda \cancel{e^{-\lambda x}}\mathrm{d}x = \frac{\lambda^{(k+1)}e^{-\lambda t}}{k!}\int_0^t (t-x)^k \mathrm{d}x = \frac{\lambda^{(k+1)}e^{-\lambda t}}{k!}\int_t^0 \frac{u^k}{-1}\mathrm{d}u$$

$f(u) = u^k$

$u = g(x) = (t - x)$

$\frac{\mathrm{d}g(x)}{\mathrm{d}x} = -1$

$$= \frac{\lambda^{(k+1)}e^{-\lambda t}}{k!}\left[-\frac{1}{k+1}u^{(k+1)}\right]_t^0 = \frac{\lambda^{(k+1)}e^{-\lambda t}}{(k+1)!}\left[u^{(k+1)}\right]_0^t = \frac{(\lambda t)^{(k+1)}e^{-\lambda t}}{(k+1)!} \checkmark$$

Poisson Process

Definition: A **Poisson process** with *intensity* λ is a collection of random variables $X_1, X_2, \ldots \in \mathbb{R}$ such that, if $N(a,b) = |\{i \mid X_i \in [a,b]\}|$, then

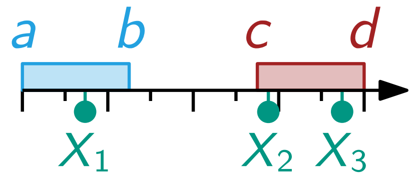
- $N(a,b) \sim \mathrm{Pois}(\lambda(b-a))$ (homogeneity)
- $a < b < c < d$: $N(a,b)$ and $N(c,d)$ are independent (independence)

- Assuming we know how many X_i are in $[\cancel{a}, b]$, *where* are they within the interval?

0 due to memorylessness

$$\Pr[N(a,b) = k] = \frac{(\lambda(b-a))^k e^{-\lambda(b-a)}}{k!}$$

- Simple case: $N(0,b) = 1$, where is X_1?



For $t \le b$: $\Pr[X_1 \le t \mid N(0,b) = 1] = \frac{\Pr[X_1 \le t \wedge N(0,b)=1]}{\Pr[N(0,b)=1]}$

exactly one in $[0,b]$ *and* it is $\le t$

$$= \frac{\Pr[N(0,t)=1 \wedge N(t,b)=0]}{\Pr[N(0,b)=1]}$$

independence of disjoint intervals

$$= \frac{\Pr[N(0,t)=1] \cdot \Pr[N(t,b)=0]}{\Pr[N(0,b)=1]}$$

for $X \sim \mathcal{U}([0,b])$

$$= \frac{(\cancel{\lambda} t)\cancel{e^{-\lambda t}} \cdot \cancel{e^{-\lambda(b-t)}}}{(\cancel{\lambda} b)\cancel{e^{-\lambda b}}} = \frac{t}{b} = F_X(t)$$

- In general: the positions of the points are distributed uniformly in an interval

Continuous Spaces: Joint Distributions

Definition: For two random variables X, Y the **joint cumulative distribution function** is

$$F_{X,Y}(a, b) = \Pr[X \le a \wedge Y \le b].$$

The **joint density function** $f_{X,Y}(a, b)$ satisfies $F_{X,Y}(a, b) = \int_{-\infty}^{a} \int_{-\infty}^{b} f_{X,Y}(x, y) \mathrm{d}y \mathrm{d}x$.

Definition: The **marginal density** of X is $f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y) \mathrm{d}y$.

Definition: The **conditional density** of X with respect to an event A is

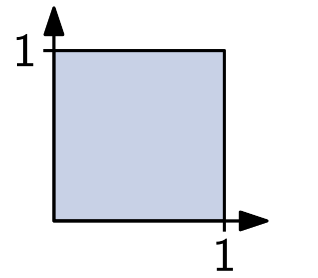
$$f_{X|A}(x) = \begin{cases} f_X(x)/\Pr[A], \text{ if } x \in A, \\ 0, \text{ otherwise.} \end{cases}$$

- For continuous Y, we specifically get $f_{X|Y=y}(x) = f_{X,Y}(x, y)/f_Y(y)$
- We can then write $f_{X,Y}(x, y) = f_{X|Y=y}(x) \cdot f_Y(y)$ (like the chain rule for probabilities)

Definition: Random variables X, Y are **independent** if $F_{X,Y}(x, y) = F_X(x) \cdot F_Y(y)$.

Example: $\mathcal{U}([0,1]^2)$

Uniform Distribution on the Unit Square



- We want to draw a point P uniformly at random from $[0,1]^2$
- Let X, Y be the x- and y-coordinates of P, respectively
- $f_P(x,y) = f_{X,Y}(x,y) = 1$ for $(x,y) \in [0,1]^2$ and $f_P(x,y) = 0$, otherwise

Marginal Distributions

Marginal Density
$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\mathrm{d}y$

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\mathrm{d}y = \int_0^1 1\mathrm{d}y = \big[y\big]_0^1 = 1 \qquad f_Y(y) = 1$$

- Note that $X \sim \mathcal{U}([0,1])$ and $Y \sim \mathcal{U}([0,1])$

Independence

X, Y independent if
$F_{X,Y}(x,y) = F_X(x) \cdot F_Y(y)$

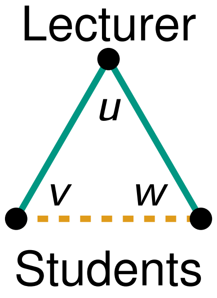
$$\begin{aligned} F_{X,Y}(a,b) = \int_{-\infty}^{a}\int_{-\infty}^{b} f_{X,Y}(x,y)\mathrm{d}y\mathrm{d}x &= \int_0^a \overbrace{\int_0^b 1\mathrm{d}y}^{\text{constant w.r.t. } x}\mathrm{d}x \\ &= \int_0^b 1\mathrm{d}y \cdot \int_0^a 1\mathrm{d}x \\ &= \int_0^b f_Y(y)\mathrm{d}y \cdot \int_0^a f_X(x)\mathrm{d}x = F_Y(b) \cdot F_X(a) \ \checkmark \end{aligned}$$

- Sample $P = (X,Y) \sim \mathcal{U}([0,1]^2)$ by independently sampling $X, Y \sim \mathcal{U}([0,1])$!

Application: Random Geometric Graphs

Motivation

- Average-case analysis: analyze models that represent the real world
- So far: Erdős-Rényi random graphs (connect two vertices independently with equal prob)
- Problem: In real networks, edges do *not* form independently
  - Two vertices are more likely to be adjacent if they have a common neighbor
    ↳ This property is called *locality* or *clustering*
  - ER-graph: $\Pr[\{v,w\} \in E \mid \{u,v\} \in E \wedge \{u,w\} \in E] = \Pr[\{v,w\} \in E]$ ✗



Idea

- Vertices are likelier to connect if their distance is already small

  $\Rightarrow$ Define vertex distances in advance by introducing geometry

> **Definition**: A **random geometric graph** is obtained by distributing vertices in a metric space and connecting any two with a probability that depends on their distance.

How many? Which space? Which metric? Which distribution? Which probability?

Simple & Realistic!

Application: Simple Random Geometric Graphs

Karlsruhe Institute of Technology

- Number: n vertices
- Space: 2-dimensional torus $\mathbb{T}^2$ (unit square with opposite sides identified)
- Metric: for $p = (p_1, p_2), q = (q_1, q_2)$: $d_i = |p_i - q_i|$
  ↳ L_∞ norm: $d(p,q) = \max_{i \in \{1,2\}} \min\{d_i, 1 - d_i\}$
- Distribution: For each v independently: $P_v \sim \mathcal{U}([0,1]^2)$
- Probability

$$\Pr[\{u,v\} \in E] = \begin{cases} 1, \text{ if } d(P_u, P_v) \le r \leftarrow \text{threshold parameter} \\ 0, \text{ otherwise} \end{cases}$$

Random Geometric Graph
Nodes distributed in metric space
Connection probability depends on distance

Expected Degree of *v*

- Neighbors of v are in $\boxed{N(v)}$ (here $N(v)$ denotes the *region* in the ground space)
- $\mathbb{E}[\deg(v)] = \mathbb{E}[\sum_{u \in V \setminus \{v\}} \mathbb{1}_{\{P_u \in N(v)\}}] = \sum_{u \in V \setminus \{v\}} \Pr[d(P_u, P_v) \le r]$
- Draw $P_u = (X, Y)$ as independent $X, Y \sim \mathcal{U}([0,1])$

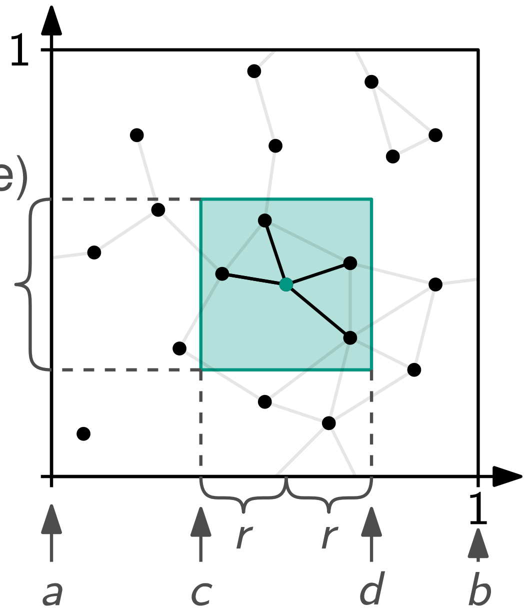
and y-coordinate of u in here

$$= \sum_{u \in V \setminus \{v\}} \frac{2r}{1-0} \cdot \frac{2r}{1-0}$$

$$= (n-1) \cdot \underbrace{4r^2}$$

(area of the region $N(v)$)

$X \sim \mathcal{U}([a,b]) : \Pr[X \in [c,d] \subseteq [a,b]] = \frac{d-c}{b-a}$



Simple Random Geometric Graphs – Locality

Locality

Realistic assumption: $r = \Theta(n^{-1/2})$ such that $\mathbb{E}[\deg(v)] = \Theta(1)$

Convention: $v = P_v$

- Two vertices v and w are likelier to connect if they have a common neighbor u

$$\Pr[\{v, w\} \in E] = \Pr[v \in N(w)] = 4r^2 = \Theta(1/n)$$

w.l.o.g assume $u = (r, r)$

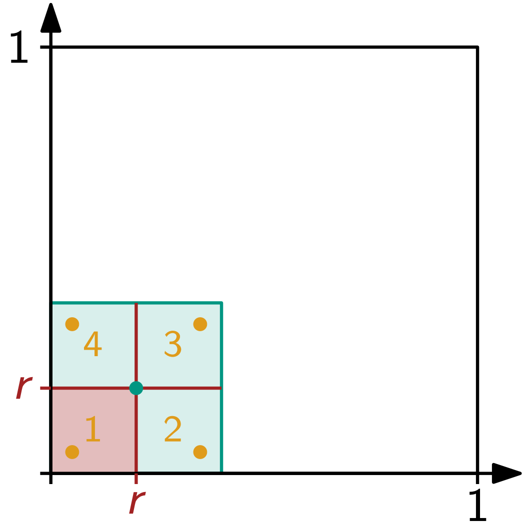
$$\Pr[\{v, w\} \in E \mid \{u, v\} \in E \wedge \{u, w\} \in E]$$

$$= \Pr[w \in N(v) \mid v \in N(u) \wedge w \in N(u)] = \frac{\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]}{\Pr[v \in N(u) \wedge w \in N(u)]}$$

Numerator $\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]$

$$= \int_{\mathbb{R}^2} \Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u) \mid v = (x, y)] f_{X,Y}(x, y) \mathrm{d}y \mathrm{d}x$$

$$= \int_0^{2r} \int_0^{2r} \Pr[w \in N(v) \wedge w \in [0, 2r]^2 \mid v = (x, y)] \mathrm{d}y \mathrm{d}x$$



Due to symmetry the area of the intersection is the same for these 4 positions of v.

$\Rightarrow$ Integrate only one quarter and multiply by 4

Law of Total Probability
$\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] f_X(x) \mathrm{d}x$

$(X, Y) \sim \mathcal{U}([0, 1]^2)$
$f_{X,Y}(x, y) = \mathbb{1}_{\{(x,y) \in [0,1]^2\}}$

Simple Random Geometric Graphs – Locality

Locality

Realistic assumption: $r = \Theta(n^{-1/2})$ such that $\mathbb{E}[\deg(v)] = \Theta(1)$

Convention: $v = P_v$

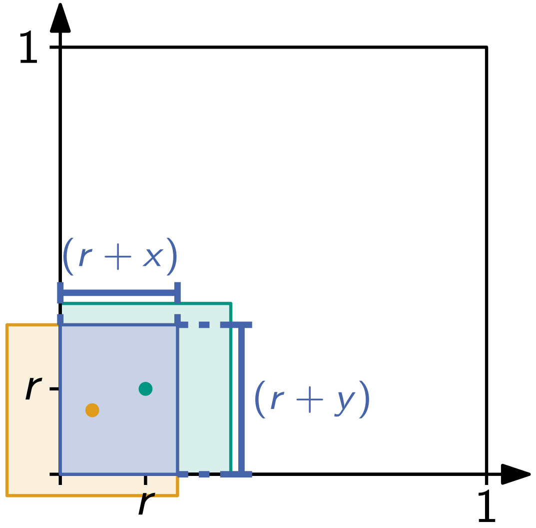
- Two vertices v and w are likelier to connect if they have a common neighbor u

$$\Pr[\{v,w\} \in E] = \Pr[v \in N(w)] = 4r^2 = \Theta(1/n)$$

w.l.o.g assume $u = (r, r)$

$$\Pr[\{v,w\} \in E \mid \{u,v\} \in E \wedge \{u,w\} \in E]$$

$$= \Pr[w \in N(v) \mid v \in N(u) \wedge w \in N(u)] = \frac{\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]}{\Pr[v \in N(u) \wedge w \in N(u)]}$$

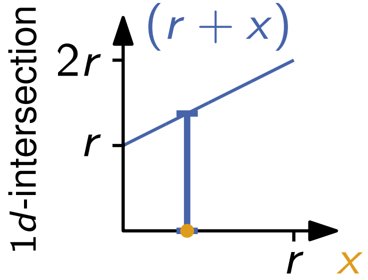


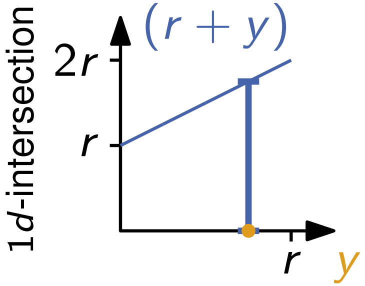
Numerator $\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]$

$$= \int_{\mathbb{R}^2} \Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u) \mid v = (x,y)] f_{X,Y}(x,y) \mathrm{d}y \mathrm{d}x$$

$$= 4 \int_0^r \int_0^r \Pr[w \in N(v) \wedge w \in [0,2r]^2 \mid v = (x,y)] \mathrm{d}y \mathrm{d}x$$

Consider size of intersection in *one* dimension depending on position of v





2d-intersection is product of 1d-intersections

$(r+x) \cdot (r+y)$

Law of Total Probability

$$\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] f_X(x) \mathrm{d}x$$

$$(X, Y) \sim \mathcal{U}([0,1]^2)$$

$$f_{X,Y}(x,y) = \mathbb{1}_{\{(x,y) \in [0,1]^2\}}$$

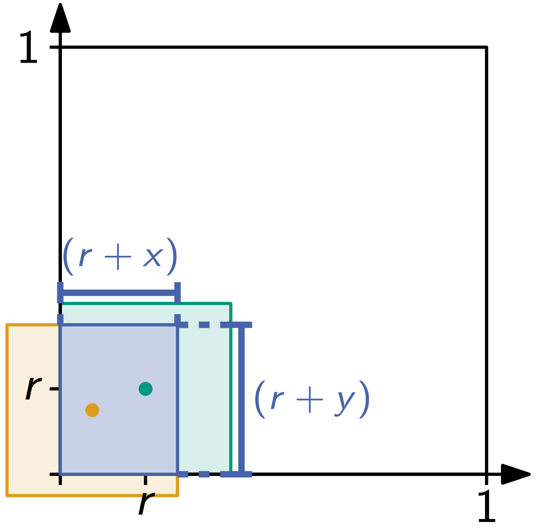
Simple Random Geometric Graphs – Locality

Locality Realistic assumption: $r = \Theta(n^{-1/2})$ such that $\mathbb{E}[\deg(v)] = \Theta(1)$ Convention: $v = P_v$

- Two vertices v and w are likelier to connect if they have a common neighbor u

$$\Pr[\{v,w\} \in E] = \Pr[v \in N(w)] = 4r^2 = \Theta(1/n)$$

w.l.o.g assume $u = (r, r)$



$$\Pr[\{v,w\} \in E \mid \{u,v\} \in E \wedge \{u,w\} \in E]$$

$$= \Pr[w \in N(v) \mid v \in N(u) \wedge w \in N(u)] = \frac{\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]}{\Pr[v \in N(u) \wedge w \in N(u)]}$$

Numerator $\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]$

$$= \int_{\mathbb{R}^2} \Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u) \mid v = (x,y)] f_{X,Y}(x,y) \mathrm{d}y\mathrm{d}x$$

$$= 4 \int_0^r \int_0^r \Pr[w \in N(v) \wedge w \in [0, 2r]^2 \mid v = (x,y)] \mathrm{d}y\mathrm{d}x$$

$$= 4 \int_0^r \int_0^r (r+x) \cdot (r+y) \mathrm{d}y\mathrm{d}x$$

$$= 4 \int_0^r (r+x) \cdot \int_0^r (r+y) \mathrm{d}y\mathrm{d}x$$

$$= 4 \int_0^r (r+y) \mathrm{d}y \cdot \int_0^r (r+x) \mathrm{d}x$$

$$= 4 \left(\int_0^r (r+x) \mathrm{d}x \right)^2$$

$$= 4 \left(\int_0^r r \mathrm{d}x + \int_0^r x \mathrm{d}x \right)^2$$

$$= 4 \left(r \left[x \right]_0^r + \left[\tfrac{1}{2} x^2 \right]_0^r \right)^2$$

$$= 4 \left(r^2 + \tfrac{1}{2} r^2 \right)^2 = 4 \left(\tfrac{3}{2} r^2 \right)^2 = \cancel{4} \frac{9}{\cancel{4}} r^4$$

$$= 9r^4$$

Law of Total Probability
$$\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] f_X(x) \mathrm{d}x$$

$$\boldsymbol{(X, Y) \sim \mathcal{U}([0,1]^2)}$$
$$f_{X,Y}(x,y) = \mathbb{1}_{\{(x,y) \in [0,1]^2\}}$$

Simple Random Geometric Graphs – Locality

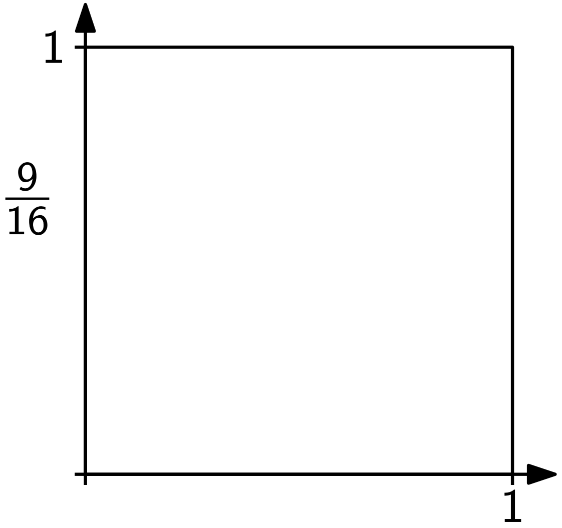
Locality Realistic assumption: $r = \Theta(n^{-1/2})$ such that $\mathbb{E}[\deg(v)] = \Theta(1)$ Convention: $v = P_v$

- Two vertices v and w are likelier to connect if they have a common neighbor u

$$\Pr[\{v,w\} \in E] = \Pr[v \in N(w)] = 4r^2 = \Theta(1/n)$$ $>$

$$\Pr[\{v,w\} \in E \mid \{u,v\} \in E \wedge \{u,w\} \in E] = \Theta(1)$$ ✓

$$= \Pr[w \in N(v) \mid v \in N(u) \wedge w \in N(u)] = \frac{\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)]}{\Pr[v \in N(u) \wedge w \in N(u)]} = \frac{9}{16}$$



Numerator $\Pr[w \in N(v) \wedge v \in N(u) \wedge w \in N(u)] = 9r^4$

Denominator

$$\Pr[v \in N(u) \wedge w \in N(u)] = \Pr[v \in N(u)] \cdot \Pr[w \in N(u)]$$

positions are drawn independently

distribution identical for all vertices

$$= (\Pr[v \in N(u)])^2$$

$$= (4r^2)^2 = 16r^4$$

Law of Total Probability
$\Pr[A] = \int_{-\infty}^{\infty} \Pr[A \mid X = x] f_X(x) \mathrm{d}x$

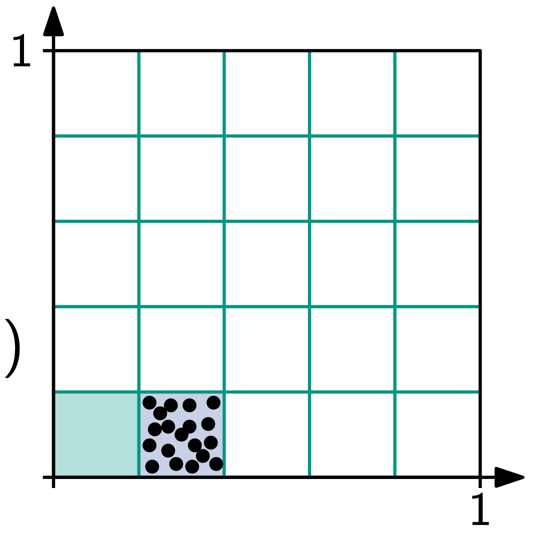
$(X,Y) \sim \mathcal{U}([0,1]^2)$
$f_{X,Y}(x,y) = \mathbb{1}_{\{(x,y) \in [0,1]^2\}}$

Application: Simple RGGs – Fair Distribution

- Discretize the space into equally sized grid cells, such that the expected number of vertices in each cell is $\log(n)$
- Each cell C_i has width and height $\sqrt{\log(n)/n}$
- Let X_i denote the number of vertices in C_i

$$\mathbb{E}[X_i] = \mathbb{E}[\textstyle\sum_{v \in V} \mathbb{1}_{\{v \in C_i\}}] = n \cdot \Pr[v \in C_i] = n \frac{\sqrt{\log(n)/n}}{1-0} \frac{\sqrt{\log(n)/n}}{1-0} = \log(n)$$

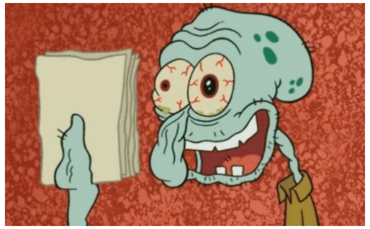
- What is the probability that each cell gets *exactly* $\log(n)$ vertices?



$$\Pr[X_1 = \log(n)] = \binom{n}{\log(n)} \left(\frac{\log(n)}{n}\right)^{\log(n)} \left(1 - \frac{\log(n)}{n}\right)^{n-\log(n)}$$

- Same distribution for all X_i: $\Pr[\forall i : X_i = \log(n)] = \cancel{\prod_i \Pr[X_i = \log(n)]}$ $\neq$
- X_1 and X_2 are *not* independent $\Pr[X_1 = \log(n) \mid X_2 = n] = 0$
- Chain rule of probability:

$$\Pr[\forall i : X_i = \log(n)]$$
$$= \Pr[X_1 = \log(n)] \cdot \Pr[X_2 = \log(n) \mid X_1 = \log(n)] \cdot \Pr[X_3 = \log(n) \mid X_1 = \log(n) \wedge X_2 = \log(n)] \cdot \ldots$$



https://i.imgflip.com/1pln6k.jpg?a471949

Poissonization

Idea

- Avoid dependencies by replacing uniform point sampling with a Poisson point process

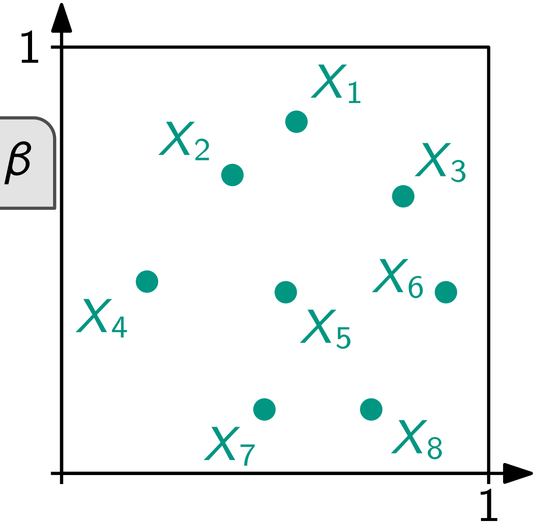
Definition: A **Poisson *Point* process** with *intensity* λ is a collection of random variables $X_1, X_2, \ldots \in \mathbb{R}^2$ such that, if $|A|$ is the area of A and $N(A) = |\{i \mid X_i \in A\}|$, then
- $N(A) \sim \mathrm{Pois}(\lambda |A|)$ (homogeneity)
- $A \cap B = \emptyset$: $N(A)$ and $N(B)$ are independent (independence)

(Generalizes to arbitrary dimension, $1d$ is the Poisson process seen earlier)

- Note: We do not know how many points we get!
- How do we choose λ?
  - We should at least *expect* n points in our ground space $[0,1]^2$

$N \sim \mathrm{Pois}(\beta) \Rightarrow \mathbb{E}[N] = \beta$

$$n = \mathbb{E}[|\{i \mid X_i \in [0,1]^2\}|] = \mathbb{E}[N([0,1]^2)] = \lambda |[0,1]^2| = \lambda$$



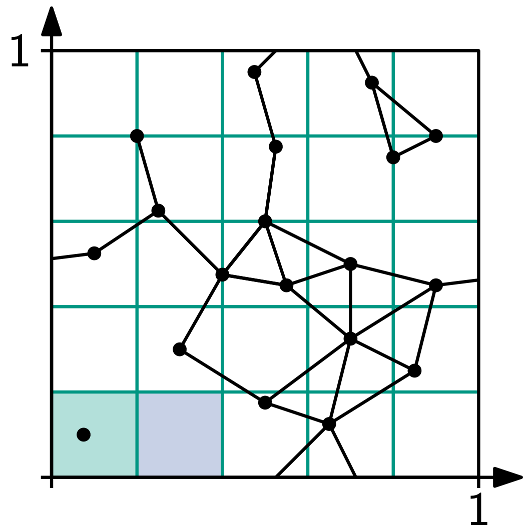
- Recall: conditioned on their number, points are distributed uniformly
- Simulate PPP: sample $N \sim \mathrm{Pois}(n)$, sample N points uniformly
- The resulting **Poissonized RGG** has n vertices in expectation

Application: Poissonized RGGs – Fair Distribution

?

- Vertices of RGG distributed using Poisson point process with intensity $\lambda = n$
- Discretize the space into equally sized grid cells, such that the expected number of vertices in each cell is $\log(n)$
- Each cell C_i has width and height $\sqrt{\log(n)/n} \Rightarrow |C_i| = \log(n)/n$
- Let X_i denote the number of vertices in $C_i \Rightarrow X_i \sim \mathrm{Pois}(\lambda|C_i|)$

$\mathbb{E}[X_i] = \lambda|C_i| = \log(n)$



- What is the probability that each cell gets *exactly* $\log(n)$ vertices?

$$\Pr[X_i = \log(n)] = \frac{(\lambda|C_i|)^{\log(n)} e^{-\lambda|C_i|}}{\log(n)!} = \frac{(\cancel{n}\frac{\log(n)}{\cancel{n}})^{\log(n)} e^{-\cancel{n}\frac{\log(n)}{\cancel{n}}}}{\log(n)!} = \frac{\log(n)^{\log(n)} e^{-\log(n)}}{\log(n)!}$$

$$\leq \frac{\cancel{\log(n)^{\log(n)}} \cancel{e^{-\log(n)}}}{e(\cancel{\frac{\log(n)}{e}})^{\log(n)}} = \frac{1}{e}$$

$N \sim \mathrm{Pois}(\lambda|A|)$

$\mathbb{E}[N] = \lambda|A|$

$\Pr[N = k] = \frac{(\lambda|A|)^k e^{-\lambda|A|}}{k!}$

$k! \geq e(k/e)^k$

there are $n/\log(n)$ cells

- Same distribution for all X_i: $\Pr[\forall i : X_i = \log(n)] = \prod_i \Pr[X_i = \log(n)]$

by definition, disjoint regions *are* independent $\leq e^{-n/\log(n)}$ ✓

but we cheated...

De-Poissonization

Karlsruhe Institute of Technology

Situation

- We started with a simple RGG (n, $\mathbb{T}^2$, L_∞-norm, $P_i \sim \mathcal{U}([0,1]^2)$, $\Pr[\{u,v\} \in E] = \mathbb{1}_{\{d(u,v) \leq r\}}$)
- Switched to Poissonized RGG (n is replaced by $\text{Pois}(n)$) and obtained $\Pr[\forall i : X_i = \log(n)] \leq e^{-n/\log(n)}$
- How can we translate this result to the original model?

Recall

- Conditioned on the number of points in area A, the points are distributed uniformly in A
- So we get from the poissonized RGG to the original, by conditioning on the fact that the number of points N in $[0,1]^2$ obtained in the Poisson point process is exactly $N = n$

$$\Pr[N = n] = \frac{(\lambda|A|)^n e^{-\lambda|A|}}{n!} = \frac{n^n e^{-n}}{n!} \leq \cancel{\left(\frac{n}{e}\right)^n} \cdot \frac{1}{\sqrt{2\pi n}\cancel{\left(\frac{n}{e}\right)^n}\underbrace{e^{\frac{1}{12n+1}}}_{\Theta(1)}} = \Theta(n^{-1/2})$$

$$\Pr_{\text{RGG}(n)}[\forall i : X_i = \log(n)] = \Pr_{\text{RGG}(\text{Pois}(n))}[\forall i : X_i = \log(n) \mid N = n]$$

$$= \frac{\Pr_{\text{RGG}(\text{Pois}(n))}[\forall i : X_i = \log(n) \wedge N = n]}{\Pr_{\text{RGG}(\text{Pois}(n))}[N = n]} \leq \frac{\Pr_{\text{RGG}(\text{Pois}(n))}[\forall i : X_i = \log(n)]}{\Pr_{\text{RGG}(\text{Pois}(n))}[N = n]} \leq \frac{e^{-n/\log(n)}}{\Theta(n^{-1/2})} \checkmark$$

$N \sim \text{Pois}(\lambda|A|)$

$$\Pr[N = k] = \frac{(\lambda|A|)^k e^{-\lambda|A|}}{k!}$$

Stirling

$$n! \geq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}}$$

RGG – The Bigger Picture

Random Geometric Graph
Nodes distributed in metric space
Connection probability depends on distance

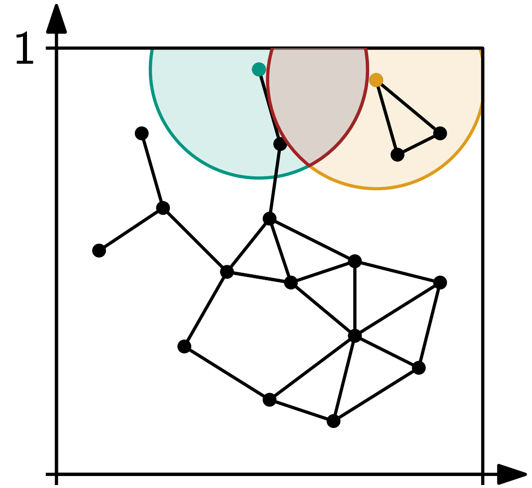
Seen so far

- Simple RGG
- n, $\mathbb{T}^2$, L_∞-norm, $P_i \sim \mathcal{U}([0,1]^2)$, $\Pr[\{u,v\} \in E] = \mathbb{1}_{\{d(u,v) \le r\}}$
- Expected degree of a vertex is $(n-1)4r^2$
- Probability to connect given common neighbor is constant

More commonly used model

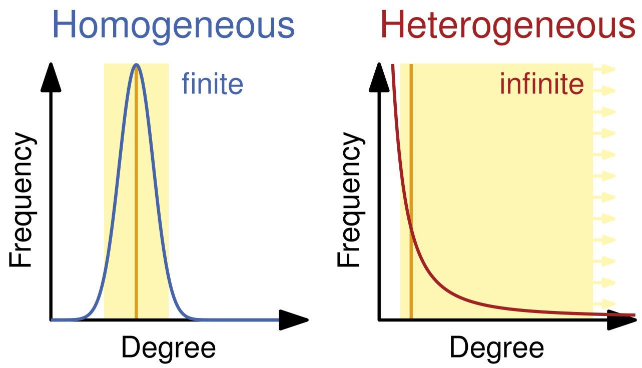
- n, $[0,1]^2$, L_2-norm, $P_i \sim \mathcal{U}([0,1]^2)$, $\Pr[\{u,v\} \in E] = \mathbb{1}_{d(u,v) \le r}$
- Complications
  - Vertices near the boundary / corners behave differently
  - Intersections of neighborhoods are lenses or parts thereof
- Still $\mathbb{E}[\deg(v)] = \Theta(nr^2)$
- Still probability to connect given common neighbor non-vanishing

$\boxed{N(v)}$ is a disk
No wrap-around!

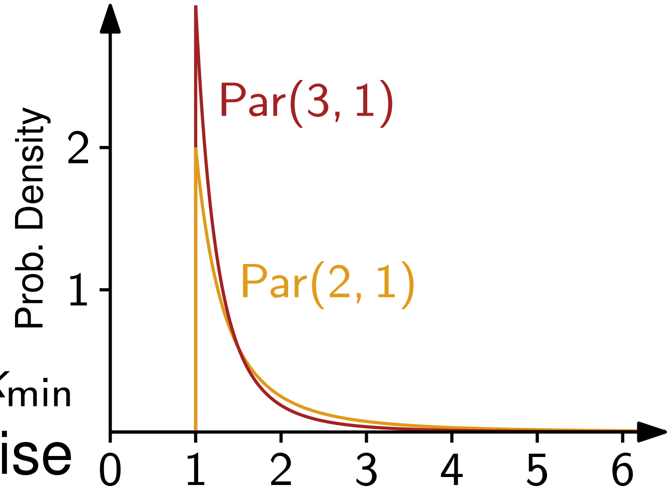


Problem: Homogeneous degree distribution does not match many real-world graphs

A Heterogeneous Distribution



Hard to distinguish!



Motivation

- Distributions seen so far have finite variance
- Graphs with corresponding degree distributions are *homogeneous*
  - $\Rightarrow$ For constant expected degree, it is very unlikely to find a high-degree vertex
- In real-world graphs high-degree vertices are not too rare (think of celebrities in a social network)

Pareto Distribution

- $X \sim \mathrm{Par}(\alpha, x_{\min})$

  α: shape parameter
  $x_{\min}$: minimum attainable value

- Probability density function: $f_X(x) = \begin{cases} \alpha x_{\min}^{\alpha} \cdot x^{-(\alpha+1)}, & \text{if } x \geq x_{\min} \\ 0, & \text{otherwise} \end{cases}$

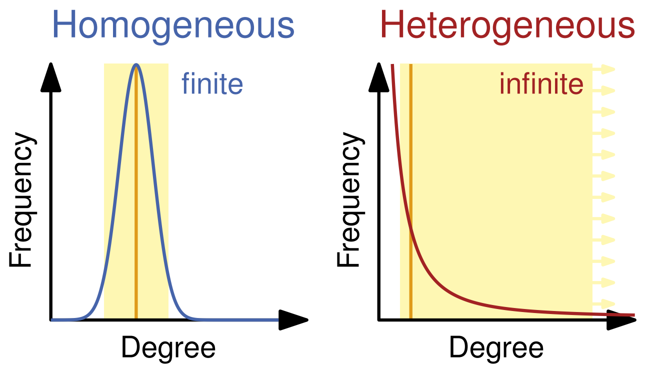
A Heterogeneous Distribution

Karlsruhe Institute of Technology

Motivation

- Distributions seen so far have finite variance
- Graphs with corresponding degree distributions are *homogeneous*

  $\Rightarrow$ For constant expected degree, it is very unlikely to find a high-degree vertex
- In real-world graphs high-degree vertices are not too rare (think of celebrities in a social network)



Pareto Distribution

- $X \sim \mathrm{Par}(\alpha, x_{\min})$

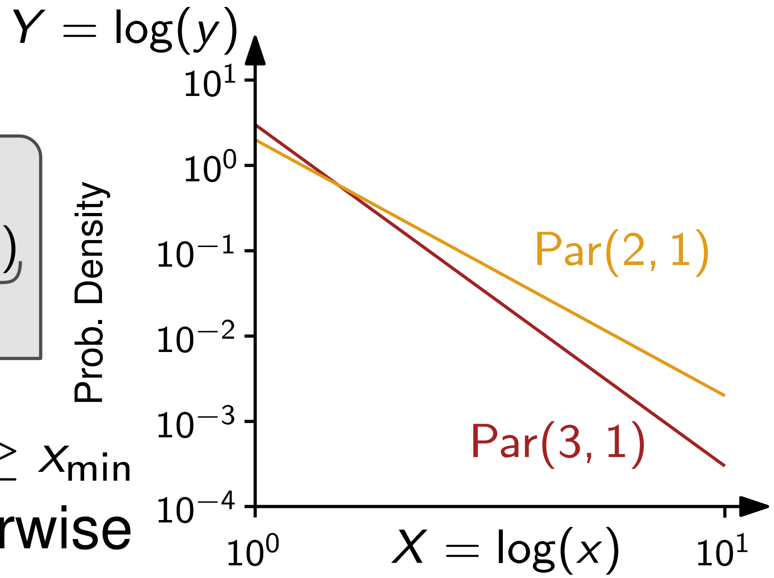
  $x_{\min}$: minimum attainable value

  α: shape parameter
- Probability density function: $f_X(x) = \begin{cases} \alpha x_{\min}^{\alpha} \cdot x^{-(\alpha+1)}, & \text{if } x \geq x_{\min} \\ 0, & \text{otherwise} \end{cases}$

Log-Log-Plot $y = bx^k$

$$\underbrace{\log(y)}_{Y} = \log(b) + k \underbrace{\log(x)}_{X}$$

$$Y = \log(b) + kX$$

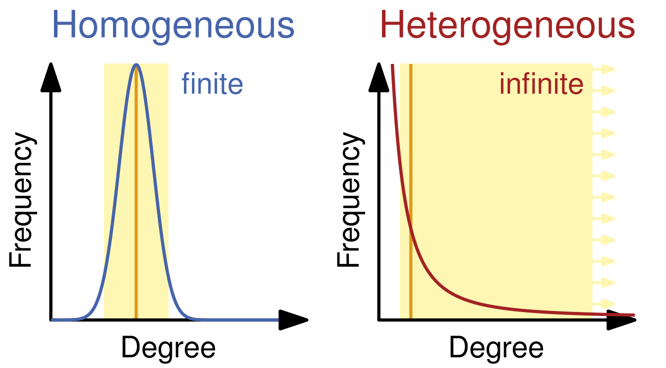


A Heterogeneous Distribution

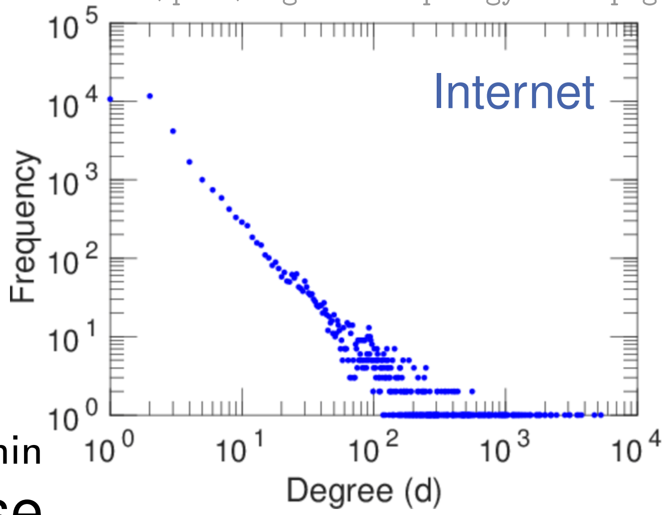
Motivation

- Distributions seen so far have finite variance
- Graphs with corresponding degree distributions are *homogeneous*

  $\Rightarrow$ For constant expected degree, it is very unlikely to find a high-degree vertex
- In real-world graphs high-degree vertices are not too rare (think of celebrities in a social network)



konect.cc/plot/degree.a.topology.full.png



Pareto Distribution

- $X \sim \mathrm{Par}(\alpha, x_{\min})$

  minimum attainable value

  shape parameter

Log-Log-Plot $y = bx^k$

$$\underbrace{\log(y)}_{Y} = \log(b) + k\,\underbrace{\log(x)}_{X}$$

$$Y = \log(b) + kX$$

- Probability density function: $f_X(x) = \begin{cases} \alpha x_{\min}^{\alpha} \cdot x^{-(\alpha+1)}, & \text{if } x \geq x_{\min} \\ 0, & \text{otherwise} \end{cases}$

Exercise: Determine for which values of α we have $\mathbb{E}[X] < \infty$ but $\mathrm{Var}[X] = \infty$

Conclusion

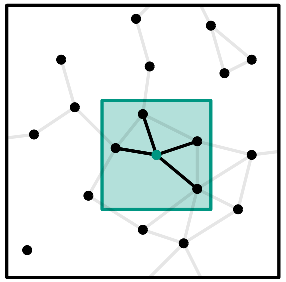
Continuous Distributions
- For our purposes they are handled like discrete versions (replacing sums with integrals)
- Seen today: Uniform distribution, exponential distribution, Pareto distribution, joint distributions

Poisson (Point) Process

(not discussed in lecture)

- Yields random point set with certain properties (homogeneity & independence)
- Number of points is a random variable
- Conditioned on certain number, points are distributed uniformly
- (De-)Poissonization to circumvent stochastic dependencies

We can simulate a PPP by drawing number according to Poisson distribution and distributing as many points uniformly

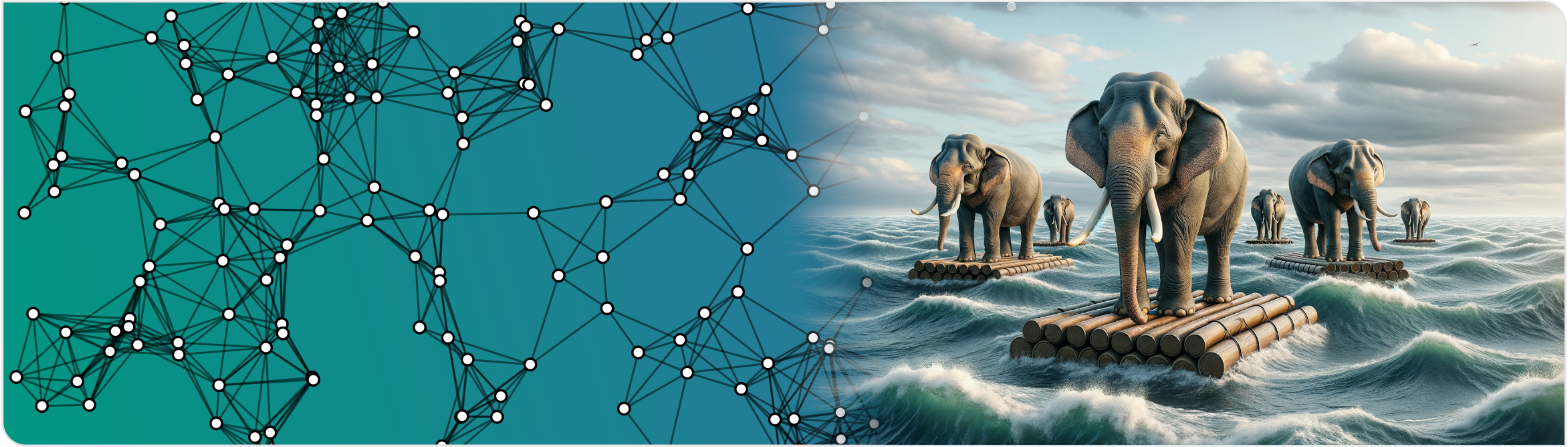


Random Geometric Graphs
- Vertices distributed at random in metric space
- Edges form with probability depending on distances
- Exhibit locality (edges tend to form between vertices with common neighbors)

Outlook: More realistic extension of RGGs featuring a heterogeneous degree distribution

Probability & Computing

Bounded Differences & Geometric Inhomogeneous Random Graphs



Recall: Concentration

Concentration Inequalities

- Bound the probability for a random variable to deviate from its expectation
- Markov: generally applicable, but not very strong
- Chebychev: stronger, but requires knowledge about variance
- Chernoff: even stronger, but requires knowledge about moment generating functions

Markov: X non-negative, $a > 0$: $\Pr[X \geq a] \leq \mathbb{E}[X]/a$.

(simpler variants work, e.g., for sums of independent random variables)

Today: similarly strong but beyond sums of independent Bernoulli random variables

Example

- k balls distributed uniformly at random over n bins
- Random variable X counts empty bins
- Let $X_i = \mathbb{1}_{\{\text{Bin } i \text{ is empty}\}}$ for $i \in [n] \Rightarrow X = \sum_{i=1}^{n} X_i$
- Concentration: $\Pr[X \geq \mathbb{E}[X] + 5\sqrt{k}]$
  - Markov: $\Pr[X \geq \mathbb{E}[X] + 5\sqrt{k}] \leq \frac{\mathbb{E}[X]}{\mathbb{E}[X]+5\sqrt{k}} = 1 - \frac{5\sqrt{k}}{\mathbb{E}[X]+5\sqrt{k}} \xrightarrow{n \to \infty} 1$ ✗
  - Chebychev: tedious... ✗
  - Chernoff: ✗ (our Bernoulli random variables are *not* independent)

$X_1 = 0 \quad X_2 = 1 \quad X_3 = 0 \quad X_4 = 1 \quad X_5 = 0 \quad X_6 = 0$

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = n \cdot \Pr[X_i = 1] = n \cdot \left(1 - \frac{1}{n}\right)^k \underset{n \to \infty}{\approx} n \cdot e^{-k/n}$$

Recall: Concentration

Concentration Inequalities

- Bound the probability for a random variable to deviate from its expectation
- Markov: generally applicable, but not very strong
- Chebychev: stronger, but requires knowledge about variance
- Chernoff: even stronger, but requires knowledge about moment generating functions

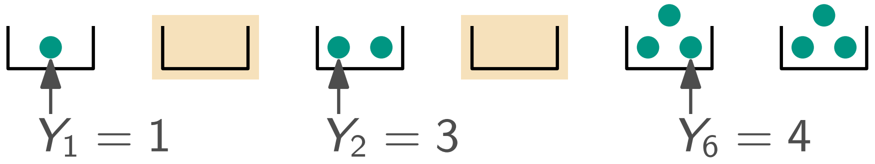
Markov: X non-negative, $a > 0$: $\Pr[X \geq a] \leq \mathbb{E}[X]/a$.

(simpler variants work, e.g., for sums of independent random variables)

Example

Today: similarly strong but beyond sums of independent Bernoulli random variables

- k balls distributed uniformly at random over n bins
- Random variable X counts empty bins
- Let independent $Y_j \sim \mathcal{U}([n])$ for $j \in [k]$ denote the bin of the j-th ball



$$\Rightarrow X = f(Y_1, \ldots, Y_k) = \sum_{i \in [n]} \mathbb{1}_{\{\nexists j : Y_j = i\}}$$ (summands not independent, but the Y_j are)

$$= \sum_{i \in [n]} \max_{j \in [k]} \{2 - |\{Y_j, i\}|\}$$ (“not” a sum Bernoulli random variables)

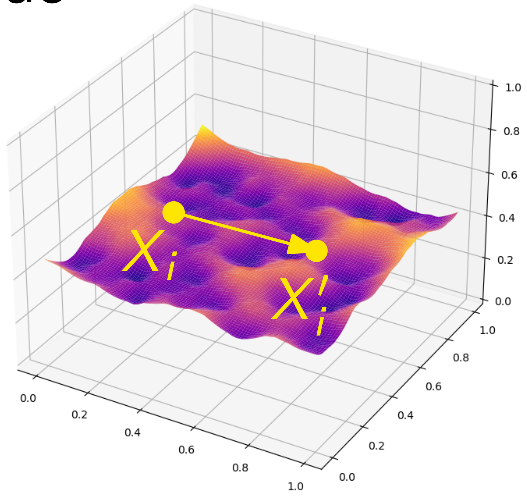
Can we show concentration for some arbitrary function of independent random variables?

... under certain conditions!

Method of Bounded Differences

Aka ... Bounded differences inequality, McDiarmid's inequality, Azuma-Hoeffding inequality

Idea If changing one of the random inputs of $f(X_1, \ldots, X_k)$ does not change $f(\cdot)$ much then a lot has to go wrong for $f(\cdot)$ to deviate from its expected value



Definition: A function $f : S^n \to \mathbb{R}$ satisfies the **bounded differences condition** ("Lipschitz condition") with parameters Δ_i, if $|f(X_1, \ldots, X_i, \ldots, X_n) - f(X_1, \ldots, X_i', \ldots, X_n)| \leq \Delta_i$ for all $i \in [n]$ and $X_i, X_i' \in S$.

Theorem: Let $X_1, \ldots, X_n$ be independent random variables taking values in a set S. Let $f : S^n \to \mathbb{R}$ satisfy the bounded differences condition with parameters Δ_i. Then, for $\Delta = \sum_{i \in [n]} \Delta_i^2$:

$$\Pr[|f - \mathbb{E}[f]| \geq t] \leq 2e^{-2t^2/\Delta}.$$ (write f for $f(X_1, \ldots, X_n)$)

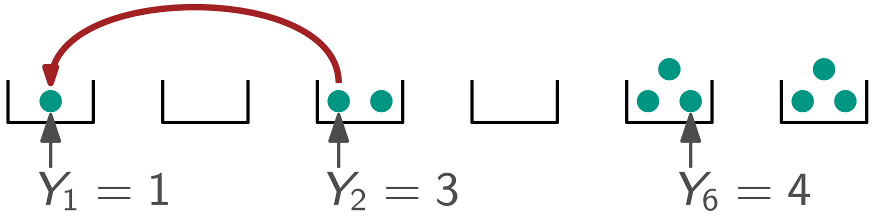
Lemma: $\Pr[f \geq \mathbb{E}[f] + t] \leq e^{-2t^2/\Delta}$.

also for $\Pr[f \leq \mathbb{E}[f] - t]$

Cor. $\mathbb{E}[f] \leq g(n)$: $\Pr[f \geq cg(n)] \leq e^{-2((c-1)g(n))^2/\Delta}$.

Application: Balls into Bins

- k balls distributed uniformly at random over n bins
- Random variable X counts empty bins



- Let independent $Y_j \sim \mathcal{U}([n])$ for $j \in [k]$ denote the bin of the j-th ball, and $X = f(Y_1, \ldots Y_k)$

Bounded differences condition

$|f(\ldots, Y_i, \ldots) - f(\ldots, Y_i', \ldots)| \leq \Delta_i$
for all i and Y_i, Y_i'

- Intuition: How much can the number of empty bins change if we move a ball from one bin to another?
  - A ball is moved from an almost empty bin to...
    - ... an empty bin $\Rightarrow +1 - 1 \Rightarrow \Delta_i = 0$
    - ... a non-empty bin $\Rightarrow +1 \Rightarrow \Delta_i = 1$
  - A ball is moved from a not almost empty bin to...
    - ... an empty bin $\Rightarrow -1 \Rightarrow \Delta_i = 1$
    - ... a non-empty bin $\Rightarrow \Delta_i = 0$

$\Bigg\}\ \Delta_i \leq 1$

Function $f(Y_1, \ldots, Y_k)$:
- $Y_1, \ldots, Y_k$ independent
- bounded differences Δ_i
- $\Delta = \sum_{i=1}^{k} \Delta_i^2$

Then $\Pr[f \geq \mathbb{E}[f] + t] \leq e^{-2t^2/\Delta}$

Concentration via bounded differences

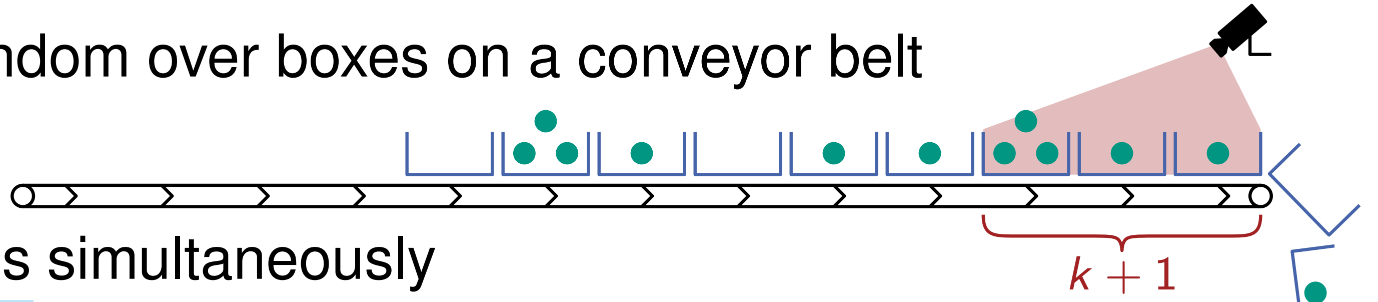
$$\Delta = \sum_{i=1}^{k} \Delta_i^2 \leq \sum_{i=1}^{k} 1^2 = k \quad \Rightarrow \Pr[f \geq \mathbb{E}[f] + 5\sqrt{k}] \leq e^{-2(5\sqrt{k})^2/k} = e^{-50}$$

Much better than Markov's $\to 1$

Application: The Factory

- Products are distributed uniformly at random over boxes on a conveyor belt

n products | $m = n/k$ boxes | $k = \log\log(n)$



- A camera scans $k+1$ consecutive boxes simultaneously
- Problem: Empty box in view $\Rightarrow$ reflection blinds camera, products remain unscanned
- Question: How many products avoid quality assurance? *Show*: $o(n)$ with prob. $1 - O(\frac{1}{n})$

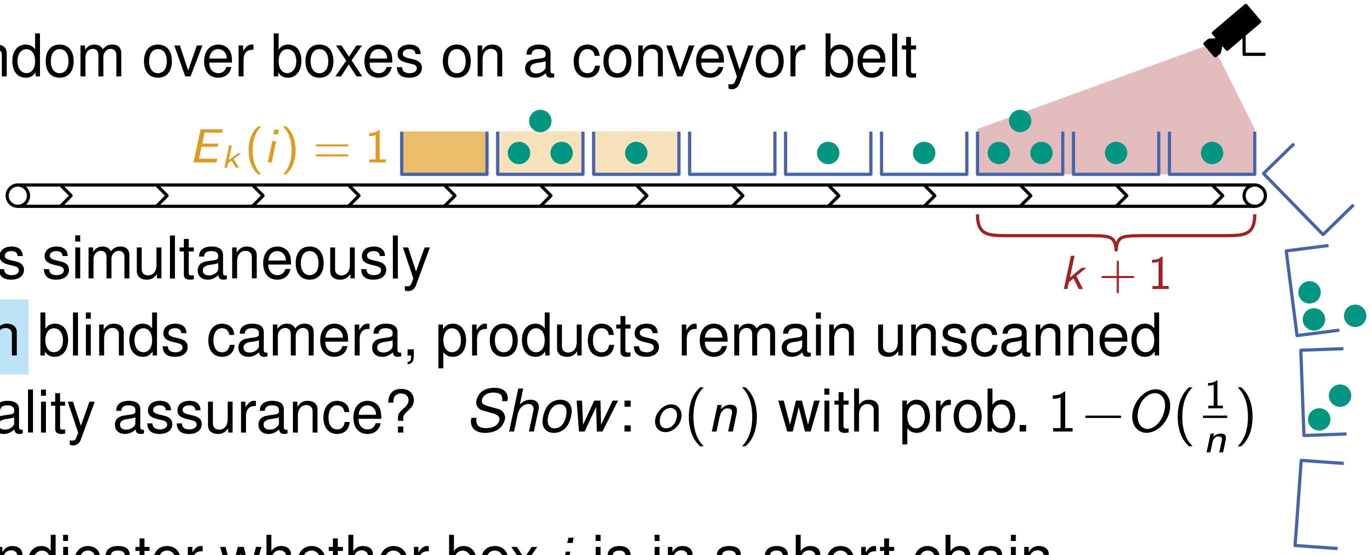
Formalize

- *chain*: consecutive sequence of non-empty boxes
- *short chain*: incl. max. chain of length $\leq k \Rightarrow$ *exactly* products in short chains unscanned
- X_i = number of products in box i, Y_i = indicator whether box i is in a short chain
- Then $X = \sum_{i=1}^{m} X_i \cdot Y_i$ is the number of unscanned products
- Problem: Dependencies (between X_i's, between X_i and Y_i)
- Solution: Relax dependencies and compute upper bound instead

Application: The Factory

- Products are distributed uniformly at random over boxes on a conveyor belt

n products $m = n/k$ boxes $k = \log\log(n)$



- A camera scans $k+1$ consecutive boxes simultaneously
- Problem: Empty box in view $\Rightarrow$ reflection blinds camera, products remain unscanned
- Question: How many products avoid quality assurance? *Show*: $o(n)$ with prob. $1 - O(\frac{1}{n})$

Relax and bound

- X_i = number of products in box i, Y_i = indicator whether box i is in a short chain
- Then $X = \sum_{i=1}^{m} X_i \cdot Y_i$ is the number of unscanned products
- $E_k(i)$ = number of empty boxes in box i and k closest (assuming k even)
  - ↳ Box i in short chain $\Rightarrow E_k(i) > 0$
- Y_i' = indicator whether $E_k(i) > 0 \Rightarrow Y_i \leq Y_i'$
  - ↳ $X = \sum_{i=1}^{m} X_i \cdot Y_i \leq \sum_{i=1}^{m} X_i \cdot Y_i' =: X'$

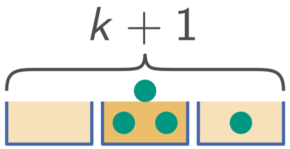
Expectation of X' (for n large enough)

$$\mathbb{E}[X'] = \sum_{i=1}^{m} \mathbb{E}[X_i \cdot Y_i']$$

(law of total expectation) $= \sum_{\ell=0}^{k+1} \mathbb{E}[X_i \cdot Y_i' \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$

$$= \sum_{\ell=0}^{k} \mathbb{E}[X_i \cdot Y_i' \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$

$$= \sum_{\ell=1}^{k} \underbrace{\mathbb{E}[X_i \mid E_k(i) = \ell]} \cdot \Pr[E_k(i) = \ell]$$



- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X' = \sum X_i \cdot Y_i'$
- X_i, products in box i
- $E_k(i)$, number empty boxes in box i and k closest
- Y_i', indicator $E_k(i) > 0$

Expected number of products in box i, knowing that exactly ℓ boxes are empty

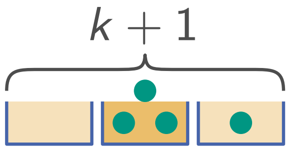
- Box i empty? $\Rightarrow X_i = 0$
- Else: n products distributed u.a.r. over $m' = m - \underbrace{\ell}_{\leq k}$ boxes

$\hookrightarrow \mathbb{E}[X_i \mid E_k(i) = \ell] = \frac{n}{m'} \leq 2 \log\log(n)$

$$m' \geq \frac{n}{\log\log(n)} - \log\log(n)$$

(for n large enough) $\geq \frac{1}{2} \frac{n}{\log\log(n)}$

Expectation of X' (for n large enough)



- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X' = \sum X_i \cdot Y_i'$
- X_i, products in box i
- $E_k(i)$, number empty boxes in box i and k closest
- Y_i', indicator $E_k(i) > 0$

$$\mathbb{E}[X'] = \sum_{i=1}^{m} \mathbb{E}[X_i \cdot Y_i']$$

(law of total expectation) $$= \sum_{\ell=0}^{k+1} \mathbb{E}[X_i \cdot Y_i' \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$

$$= \sum_{\ell=0}^{k} \mathbb{E}[X_i \cdot Y_i' \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$

$$= \sum_{\ell=1}^{k} \mathbb{E}[X_i \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$

$$\leq \sum_{\ell=1}^{k} 2\log\log(n) \cdot \Pr[E_k(i) = \ell]$$

$$= 2\log\log(n) \sum_{\ell=1}^{k} \Pr[E_k(i) = \ell]$$

$$\leq \Pr[\text{“Exists an empty box among } k+1\text{”}]$$

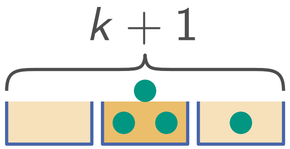
(union bound) $$\leq (k+1) \cdot \Pr[\text{“A given box is empty”}]$$

$$\leq 2k\left(1 - \frac{1}{m}\right)^n$$

$(1 + x \leq e^x)$

$$= 2k\left(1 - \frac{k}{n}\right)^n \leq 2k \cdot e^{-k} = 2\frac{\log\log(n)}{\log(n)}$$

Expectation of X' (for n large enough)



- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X' = \sum X_i \cdot Y_i'$
- X_i, products in box i
- $E_k(i)$, number empty boxes in box i and k closest
- Y_i', indicator $E_k(i) > 0$

$$\mathbb{E}[X'] = \sum_{i=1}^{m} \mathbb{E}[X_i \cdot Y_i']$$

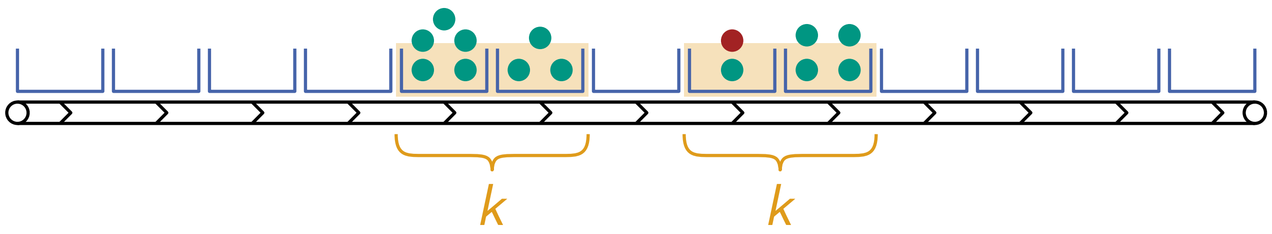
(law of total expectation)
$$= \sum_{\ell=0}^{k+1} \mathbb{E}[X_i \cdot Y_i' \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$
$$= \sum_{\ell=0}^{k} \mathbb{E}[X_i \cdot Y_i' \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$
$$= \sum_{\ell=1}^{k} \mathbb{E}[X_i \mid E_k(i) = \ell] \cdot \Pr[E_k(i) = \ell]$$
$$\leq \sum_{\ell=1}^{k} 2\log\log(n) \cdot \Pr[E_k(i) = \ell]$$
$$= 2\log\log(n) \sum_{\ell=1}^{k} \Pr[E_k(i) = \ell]$$
$$\leq 2\log\log(n) \cdot 2\frac{\log\log(n)}{\log(n)}$$
$$= 4\frac{\log\log(n)^2}{\log(n)}$$

$$\mathbb{E}[X'] = \sum_{i=1}^{m} 4\frac{\log\log(n)^2}{\log(n)} = m \cdot 4\frac{\log\log(n)^2}{\log(n)} = \frac{n}{\cancel{\log\log(n)}} \cdot 4\frac{\log\log(n)^{\cancel{2}}}{\log(n)} = n \cdot 4\frac{\log\log(n)}{\log(n)} = o(n) \checkmark$$

Concentration of X (for n large enough)

Bounded Differences

- View X as a function $f(Z_1, ..., Z_n)$ of independent rand. var. where Z_j for $j \in [n]$ denotes the box of the j-th product
- Bounded differences condition:
  - Worst change in number of products in short chains when moving a single product from one box to another
  - Consider chain of $2k + 1$ boxes containing *all* n products and one box contains only one of them



$\Rightarrow X = 0$, since no short chain and, thus, no products in short chains

- Move product to next box

$\Rightarrow X = n$, since all products in short chains now

$\Delta_j \leq n$

- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X = \sum X_i \cdot Y_i$
- X_i, products in box i
- Y_i, indicator i in short chain

$\mathbb{E}[X] \leq \mathbb{E}[X'] \leq 4n\frac{\log\log(n)}{\log(n)}$

$|f(..., Z_j, ...) - f(..., Z_j', ...)| \leq \Delta_j$ for all j and Z_j, Z_j'

Concentration of X (for n large enough)

Bounded Differences

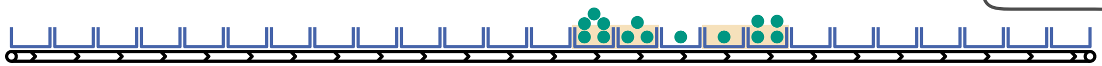
- View X as a function $f(Z_1, \ldots, Z_n)$ of independent rand. var. where Z_j for $j \in [n]$ denotes the box of the j-th product
- Bounded differences condition: $\Delta_j \leq n$
- Bounded differences inequality:

$$\Delta = \sum_{j=1}^{n} \Delta_j^2 \leq \sum_{j=1}^{n} n^2 = n^3 \qquad g(n) = 4n\frac{\log\log(n)}{\log(n)}$$

$$\Pr\left[X \geq c4n\frac{\log\log(n)}{\log(n)}\right] \leq \exp\left(-\frac{2(c-1)^2\left(4n\frac{\log\log(n)}{\log(n)}\right)^2}{n^3}\right)$$

$$= \exp\left(-\Theta\left(\underbrace{\frac{\log\log(n)^2}{n\log(n)^2}}_{=o(1)}\right)\right) \xrightarrow{n\to\infty} 1$$

This bound is useless, since worst-case changes are too big



But this case (all products in few boxes) is super unlikely...

- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X = \sum X_i \cdot Y_i$
- X_i, products in box i
- Y_i, indicator i in short chain

$\mathbb{E}[X] \leq \mathbb{E}[X'] \leq 4n\frac{\log\log(n)}{\log(n)}$

$|f(\ldots, Z_j, \ldots) - f(\ldots, Z_j', \ldots)| \leq \Delta_j$ for all j and Z_j, Z_j'

Function $f(Z_1, \ldots, Z_n)$:
- $Z_1, \ldots, Z_n$ independent
- bounded differences Δ_j
- $\Delta = \sum_{j=1}^{n} \Delta_j^2$
- $g(n) \geq \mathbb{E}[f]$

$\Pr[f \geq cg(n)] \leq e^{-2((c-1)g(n))^2/\Delta}$.

Method of Typical Bounded Differences

Definition: A function $f\colon S^n \to \mathbb{R}$ satisfies the **typical bounded differences condition** with respect to

- an event $A \subseteq S^n$ and
- parameters $\Delta_i^A \leq \Delta_i$ for $i \in [n]$,

if $$|f(X_1, \dots, X_i, \dots, X_n) - f(X_1, \dots, X_i', \dots, X_n)| \leq \begin{cases} \Delta_i^A, & \text{if } (X_1, \dots, X_i, \dots, X_n) \in A, \\ \Delta_i, & \text{otherwise} \end{cases}$$

for all $i \in [n]$ and $X_i, X_i' \in S$.

- Δ_i^A is worst-case change, assuming A held before the change

Theorem: Let $X_1, \dots, X_n$ be independent random variables taking values in a set S, let $A \subseteq S^n$ be an event, and let $f\colon S^n \to \mathbb{R}$ satisfy the typical bounded differences condition w.r.t. A and parameters $\Delta_i^A \leq \Delta_i$. Then, for $g(n) \geq \mathbb{E}[f]$, for all $\varepsilon_i \in (0, 1]$ and $\Delta = \sum_{i \in [n]} (\Delta_i^A + \varepsilon_i(\Delta_i - \Delta_i^A))^2$: $\Pr[f \geq cg(n)] \leq e^{-((c-1)g(n))^2/(2\Delta)} + \Pr[\neg A] \sum_{i \in [n]} \frac{1}{\varepsilon_i}$.

Corollary of "On the Method of Typical Bounded Differences", Warnke, Comb. Probab. Comput. 2015

Method of Typical Bounded Differences

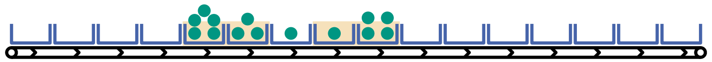
Theorem: Let $X_1, \ldots, X_n$ be independent random variables taking values in a set S, let $A \subseteq S^n$ be an event, and let $f\colon S^n \to \mathbb{R}$ satisfy the typical bounded differences condition w.r.t. A and parameters $\Delta_i^A \leq \Delta_i$. Then, for $g(n) \geq \mathbb{E}[f]$, for all $\varepsilon_i \in (0,1]$ and $\Delta = \sum_{i\in[n]}(\Delta_i^A + \varepsilon_i(\Delta_i - \Delta_i^A))^2$: $\Pr[f \geq cg(n)] \leq e^{-((c-1)g(n))^2/(2\Delta)} + \Pr[\neg A]\sum_{i\in[n]}\frac{1}{\varepsilon_i}$.

- Function of independent random variables as before
- A is the good, typical event that should be very likely to occur
- Δ is sum of squared worst-case changes as before
  - We still consider general worst-case changes as before
  - But we can use the ε_i to mitigate the worst-case effects
  - And focus on the worst-case changes, assuming A held before the change
- But we have to pay for the mitigation!
  - With the probability that the good event A does not occur
  - Multiplied with the inverse mitigators

The more we need to mitigate, the higher the price!

Not too bad if A is very likely to occur!

Application: The Factory (2nd Try)

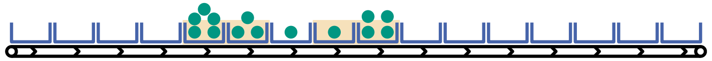


- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X = \sum X_i \cdot Y_i$
- X_i, products in box i
- Y_i, indicator i in short chain

$\mathbb{E}[X] \leq \mathbb{E}[X'] \leq 4n\frac{\log\log(n)}{\log(n)}$

- View X as a function $f(Z_1, ..., Z_n)$ of independent rand. var. where Z_j for $j \in [n]$ denotes the box of the j-th product
- Bounded differences condition: $\Delta_j \leq n$
  - When all n products fall into $2k+1 = O(\log\log(n))$ boxes
  - But expected number of products in a single box i: $\mathbb{E}[B_i] = \frac{n}{m} = \frac{n}{\frac{n}{\log\log(n)}} = \log\log(n)$
  - And, thus, expected number in sequence of $2k+1$ boxes $\mathbb{E}[S] = \sum_{i=1}^{2k+1} \mathbb{E}[B_i] = O(\log\log(n)^2) \leq \delta\log(n) =: g(n)$ (for any $\delta > 0$ and suffciently large n)
  - So *typically* a sequence should contain way fewer than n products
- Typical event $A = \{$"Every sequence of $2k+1$ boxes contains $O(\log(n))$ products"$\}$
  - See S as sum of independent Bernoulli rand. var. (whether j-th product is in sequence)
  - Chernoff: For $g(n) \geq \mathbb{E}[S]$: $\Pr[S \geq (1+\varepsilon)g(n)] \leq e^{-\varepsilon^2/3 \cdot g(n)} = e^{-\varepsilon^2/3 \cdot \delta\log(n)} = n^{-\delta\varepsilon^2/3}$
  - Union bound over $\leq n$ sequences: $\Pr[\neg A] \leq n^{-\delta\varepsilon^2/3+1} \leq n^{-\lambda}$ (for arbitrarily large λ)

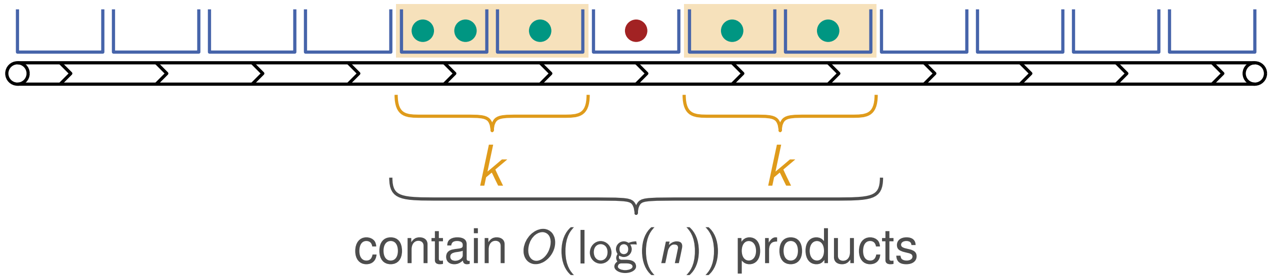
Application: The Factory (2nd Try)



- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X = \sum X_i \cdot Y_i$
- X_i, products in box i
- Y_i, indicator i in short chain

$\mathbb{E}[X] \leq \mathbb{E}[X'] \leq 4n\frac{\log\log(n)}{\log(n)}$

- View X as a function $f(Z_1, ..., Z_n)$ of independent rand. var. where Z_j for $j \in [n]$ denotes the box of the j-th product
- Bounded differences condition: $\Delta_j \leq n$
- Typical event $A = \{$“Every sequence of $2k+1$ boxes contains $O(\log(n))$ products”$\}$, $\Pr[\neg A] \leq n^{-\lambda}$ (for arbitrary λ)
- Typical bounded differences condition:
  - Worst change in f when moving a product from one box to another, assuming A held before the move



  - Moving one product empties at most one box $\Rightarrow$ at most two new short chains
  - Assuming A, these short chains combined contain $O(\log(n))$ products $\Rightarrow \Delta_j^A = O(\log(n))$

Application: The Factory (2nd Try)

- View X as a function $f(Z_1, \ldots, Z_n)$ of independent rand. var. where Z_j for $j \in [n]$ denotes the box of the j-th product
- Bounded differences condition: $\Delta_j \le n$
- Typical event $A = \{$"Every sequence of $2k+1$ boxes contains $O(\log(n))$ products"$\}$, $\Pr[\neg A] \le n^{-\lambda}$ (for arbitrary λ)
- Typical bounded differences condition: $\Delta_j^A = O(\log(n))$
- Typical bounded differences inequality:

$$\Delta = \sum_{j=1}^{n} (\Delta_j^A + \varepsilon_j(\Delta_j \cancel{- \Delta_j^A}))^2 \qquad \varepsilon_j = \frac{1}{n}$$

$$\le \sum_{j=1}^{n} (\Delta_j^A + \varepsilon_j \Delta_j)^2$$

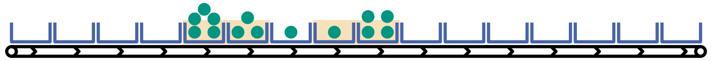
Mitigators, arbitrary $\in (0, 1]$!

$$\le \sum_{j=1}^{n} (O(\log(n)) + \varepsilon_j n)^2$$

$$= \sum_{j=1}^{n} (O(\log(n)) + 1)^2$$

$$= O(n \log(n)^2)$$

Much better than n^3 from before!



- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X = \sum X_i \cdot Y_i$
- X_i, products in box i
- Y_i, indicator i in short chain

$\mathbb{E}[X] \le \mathbb{E}[X'] \le 4n \frac{\log\log(n)}{\log(n)}$

Function $f(Z_1, \ldots, Z_n)$:

- $Z_1, \ldots, Z_n$ independent
- typical event A
- bounded differences $\Delta_j^A \le \Delta_j$
- $\Delta = \sum_{j=1}^{n} (\Delta_j^A + \varepsilon_j(\Delta_j - \Delta_j^A))^2$
- $g(n) \ge \mathbb{E}[f]$

$\Pr[f \ge c g(n)] \le e^{-((c-1)g(n))^2/(2\Delta)} + \Pr[\neg A] \sum_{j=1}^{n} \frac{1}{\varepsilon_j}$

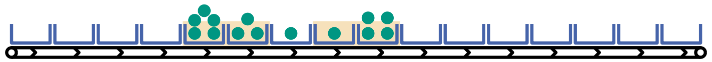
Application: The Factory (2nd Try)

- View X as a function $f(Z_1, \ldots, Z_n)$ of independent rand. var. where Z_j for $j \in [n]$ denotes the box of the j-th product
- Bounded differences condition: $\Delta_j \leq n$
- Typical event $A = \{$"Every sequence of $2k+1$ boxes contains $O(\log(n))$ products"$\}$, $\Pr[\neg A] \leq n^{-\lambda}$ (for arbitrary λ)
- Typical bounded differences condition: $\Delta_j^A = O(\log(n))$
- Typical bounded differences inequality:

$$\Delta = O(n \log(n)^2) \quad g(n) = 4n\frac{\log\log(n)}{\log(n)} \qquad \varepsilon_j = \frac{1}{n}$$

$$\Pr\left[X \geq c4n\frac{\log\log(n)}{\log(n)}\right] \leq \underbrace{\exp\left(-\Omega\left(n\frac{\log\log(n)^2}{\log(n)^4}\right)\right)}_{=O(1/n)} + \underbrace{\Pr[\neg A]\sum_{j=1}^{n}\frac{1}{\varepsilon_j}}_{\leq n^{-\lambda}\cdot n^2 = O(1/n) \text{ for } \lambda = 3}$$

✓



- n products
- $m = n/k$ boxes, $k = \log\log(n)$
- $X = \sum X_i \cdot Y_i$
- X_i, products in box i
- Y_i, indicator i in short chain

$$\mathbb{E}[X] \leq \mathbb{E}[X'] \leq 4n\frac{\log\log(n)}{\log(n)}$$

Function $f(Z_1, \ldots, Z_n)$:
- $Z_1, \ldots, Z_n$ independent
- typical event A
- bounded differences $\Delta_j^A \leq \Delta_j$
- $\Delta = \sum_{j=1}^{n}(\Delta_j^A + \varepsilon_j(\Delta_j - \Delta_j^A))^2$
- $g(n) \geq \mathbb{E}[f]$

$$\Pr[f \geq cg(n)] \leq e^{-((c-1)g(n))^2/(2\Delta)} + \Pr[\neg A]\sum_{j=1}^{n}\frac{1}{\varepsilon_j}$$

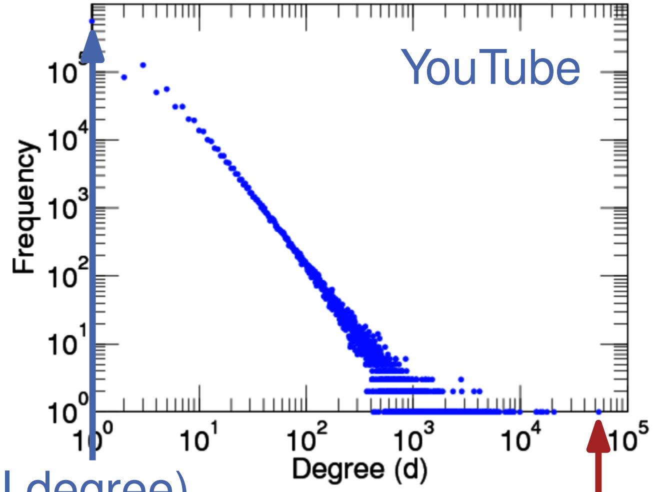
Geometric Inhomogeneous Random Graphs

Karlsruhe Institute of Technology

Motivation

- Average-case analysis: analyze models that represent the real world
- Models seen so far
  - Erdős-Rényi random graphs: simple but no locality
  - Random geometric graphs: locality but $\underbrace{\text{no heterogeneity}}_{\text{Not realistic: celebrities are very-high-degree vertices in social networks}}$ (all vertices roughly same degree)

konect.cc/plot/degree.a.youtube-links.full.png



- Realistic representation: power-law distribution

"Scale-free networks well done", Voitalov, van der Hoorn, van der Hofstad, Krioukov, Phys. Rev. Research. 2019

- Pareto distribution: $X \sim \mathrm{Par}(\alpha, x_{\min})$

$$f_X(x) = \begin{cases} \alpha x_{\min}^{\alpha} \cdot x^{-(\alpha+1)}, & \text{if } x \geq x_{\min} \\ 0, & \text{otherwise} \end{cases}$$

(most vertices small degree)

(few vertices very high degree)

Idea

- Add Pareto distribution to RGGs

Geometric Inhomogeneous Random Graphs

Definition

- Consider n vertices
- For each vertex v independently:
  - Draw a *position* x_v uniformly on $\mathbb{T}^d$
  - Draw a *weight* w_v from $\mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3) \Rightarrow f_{w_v}(w) = (\tau - 1)w^{-\tau}$

“Power-Law Exponent”

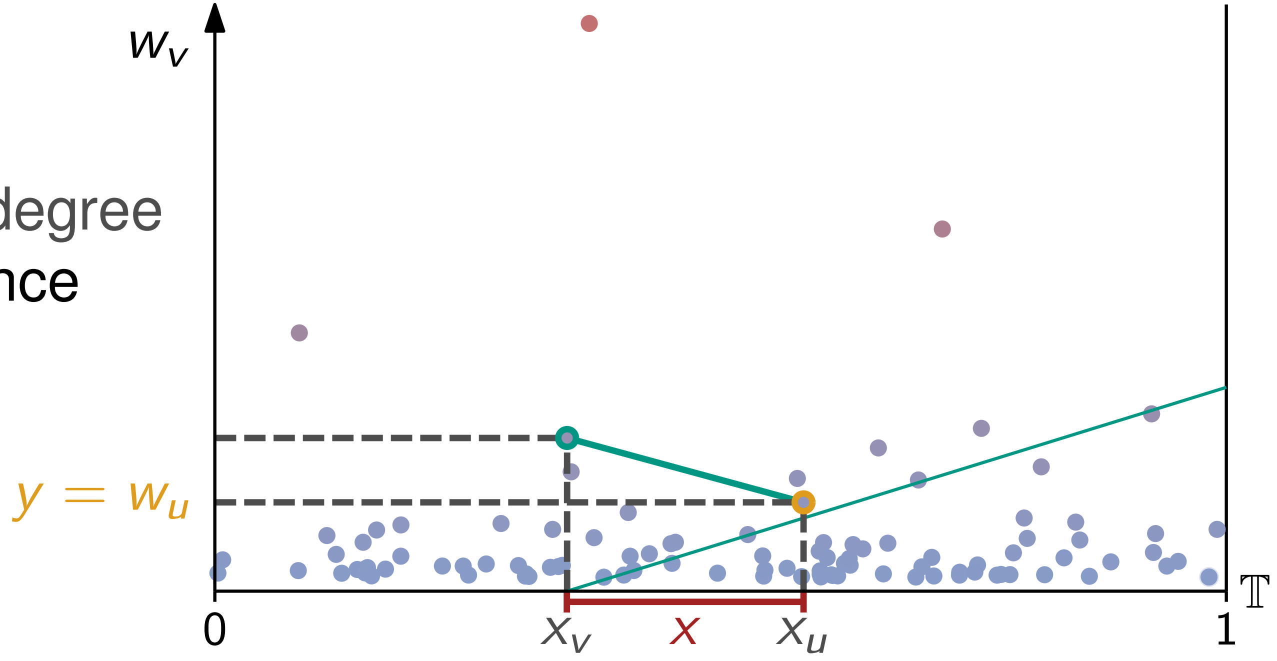
- Connect u and v with an edge, iff

$$\underbrace{\mathrm{dist}(x_u, x_v)}_{L_\infty\text{-norm}} \leq \left(\lambda \frac{w_u \cdot w_v}{n}\right)^{1/d}$$

const. controls the avg. degree

- For $d = 1$, linear relation between distance and weight $y = w_u, x = \mathrm{dist}(x_u, x_v)$

$$x \leq \lambda \frac{w_v \cdot y}{n} \Leftrightarrow y \geq \frac{n}{\lambda w_v} x$$



Geometric Inhomogeneous Random Graphs

Definition

- Consider n vertices
- For each vertex v independently:
  - Draw a *position* x_v uniformly on $\mathbb{T}^d$
  - Draw a *weight* w_v from $\mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3) \Rightarrow f_{w_v}(w) = (\tau - 1)w^{-\tau}$

"Power-Law Exponent"

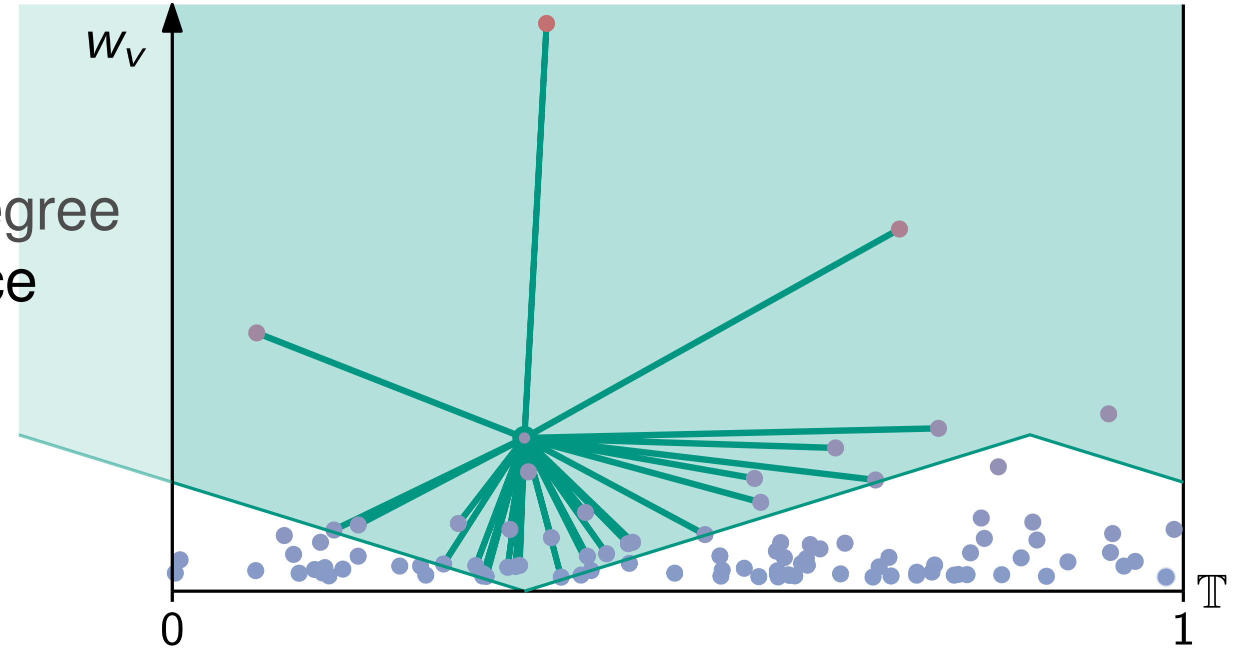
- Connect u and v with an edge, iff

$$\underbrace{\mathrm{dist}(x_u, x_v)}_{L_\infty\text{-norm}} \leq \left(\lambda \frac{w_u \cdot w_v}{n}\right)^{1/d}$$

const. controls the avg. degree

- For $d = 1$, linear relation between distance and weight $\quad y = w_u, x = \mathrm{dist}(x_u, x_v)$

$$x \leq \lambda \frac{w_v \cdot y}{n} \Leftrightarrow y \geq \frac{n}{\lambda w_v} x$$



Geometric Inhomogeneous Random Graphs

Definition

- Consider n vertices
- For each vertex v independently:
  - Draw a *position* x_v uniformly on $\mathbb{T}^d$
  - Draw a *weight* w_v from $\mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3) \Rightarrow f_{w_v}(w) = (\tau - 1)w^{-\tau}$

“Power-Law Exponent”

- Connect u and v with an edge, iff

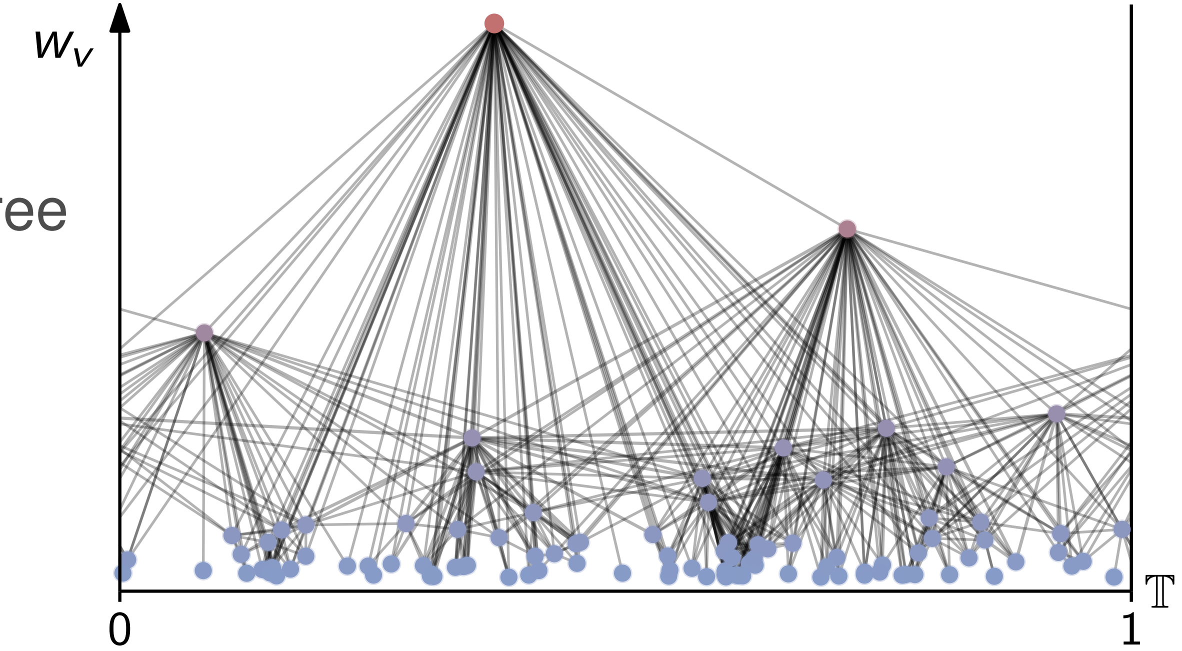
$$\underbrace{\mathrm{dist}(x_u, x_v)}_{L_\infty\text{-norm}} \leq \left(\lambda \frac{w_u \cdot w_v}{n}\right)^{1/d}$$

const. controls the avg. degree

- For $d = 1$, linear relation between distance and weight $\quad y = w_u, x = \mathrm{dist}(x_u, x_v)$

$$x \leq \lambda \frac{w_v \cdot y}{n} \Leftrightarrow y \geq \frac{n}{\lambda w_v} x$$

- The lower w_v, the steeper the wedge
  ↳ The lower the degree



Institute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithms

Expected Degree ($d = 1$)

- Consider vertex v with weight w_v
- We want to compute $\mathbb{E}[\deg(v) \mid w_v]$
- Consider X_u for $u \in V \setminus \{v\}$ indicating whether $\{u, v\} \in E$

$$\deg(v) = \sum_{u \in V \setminus \{v\}} X_u$$

$$\mathbb{E}[\deg(v) \mid w_v] = \sum_{u \in V \setminus \{v\}} \mathbb{E}[X_u \mid w_v]$$

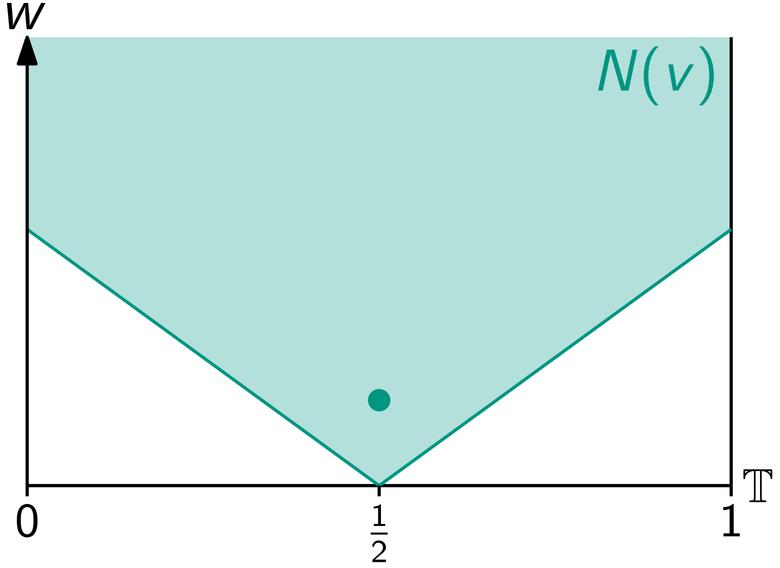
$$= \underbrace{\Theta(n \Pr[\underbrace{\{u, v\} \in E}_{u \in \boxed{N(v)}} \mid w_v])}_{}$$

This is *not* the area of the shape,
since weights are *not* distributed uniformly!

$\Rightarrow$ Use law of total probability to account for that

GIRG
- n independent vertices
- $x_v \sim \mathcal{U}([0,1])$
- $w_v \sim \mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2,3)$
  $f_{w_v}(w) = (\tau - 1) w^{-\tau}$
- u, v adjacent iff
  $\mathrm{dist}(x_u, x_v) \leq \lambda \frac{w_u \cdot w_v}{n}$

w.l.o.g $x_v = \frac{1}{2}$



Expected Degree ($d = 1$)

- Consider vertex v with weight w_v
- We want to compute $\mathbb{E}[\deg(v) \mid w_v]$
- Consider X_u for $u \in V \setminus \{v\}$ indicating whether $\{u, v\} \in E$

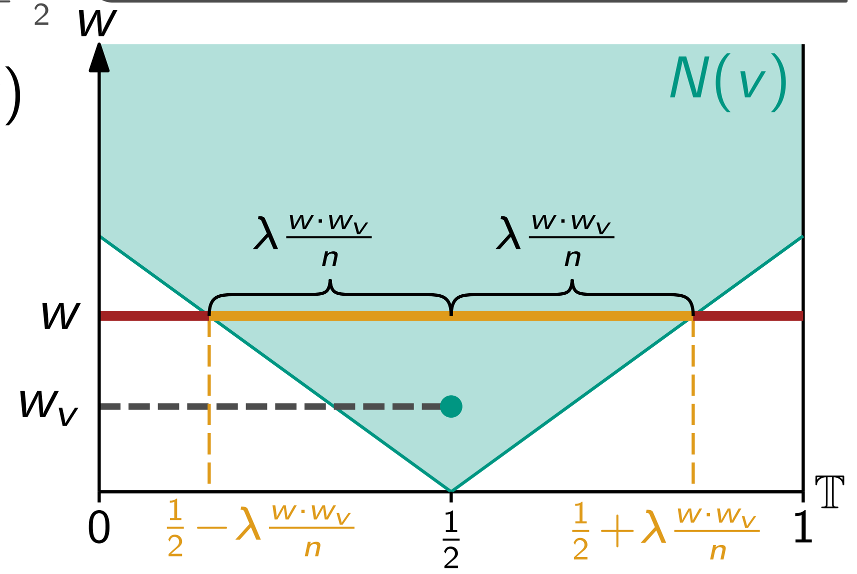
$$\deg(v) = \sum_{u \in V \setminus \{v\}} X_u$$

$$\begin{aligned}\mathbb{E}[\deg(v) \mid w_v] &= \sum_{u \in V \setminus \{v\}} \mathbb{E}[X_u \mid w_v] \\ &= \Theta(n \Pr[\{u, v\} \in E \mid w_v]) \\ &= \Theta(n \int_1^\infty \underbrace{\Pr[u \in N(v) \mid w_u = w, w_v]}_{= \Pr[x_u \in [\frac{1}{2} - \lambda\frac{w \cdot w_v}{n}, \frac{1}{2} + \lambda\frac{w \cdot w_v}{n}]]} f_{w_u}(w) \mathrm{d}w)\end{aligned}$$

w.l.o.g $x_v = \frac{1}{2}$

Case 1: $w \leq \frac{n}{2\lambda w_v} \Rightarrow \lambda\frac{w \cdot w_v}{n} \leq \frac{1}{2} \longrightarrow = 2\lambda\frac{w \cdot w_v}{n} = \Theta(\frac{w \cdot w_v}{n})$

GIRG
- n independent vertices
- $x_v \sim \mathcal{U}([0, 1])$
- $w_v \sim \mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3)$
  $f_{w_v}(w) = (\tau - 1)w^{-\tau}$
- u, v adjacent iff
  $\mathrm{dist}(x_u, x_v) \leq \lambda\frac{w_u \cdot w_v}{n}$



Expected Degree ($d = 1$)

- Consider vertex v with weight w_v
- We want to compute $\mathbb{E}[\deg(v) \mid w_v]$
- Consider X_u for $u \in V \setminus \{v\}$ indicating whether $\{u, v\} \in E$

$$\deg(v) = \sum_{u \in V \setminus \{v\}} X_u$$

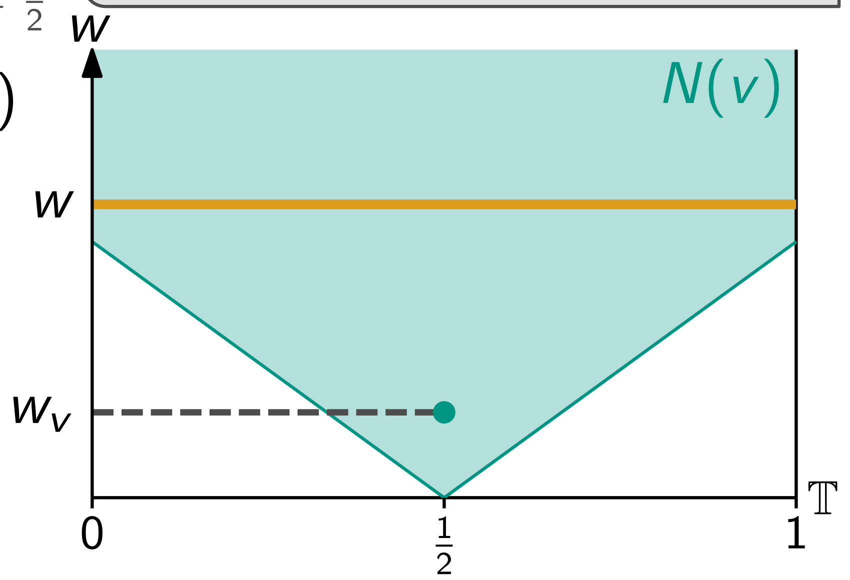
$$\begin{aligned}\mathbb{E}[\deg(v) \mid w_v] &= \sum_{u \in V \setminus \{v\}} \mathbb{E}[X_u \mid w_v] \\ &= \Theta(n \Pr[\{u, v\} \in E \mid w_v]) \\ &= \Theta(n \int_1^\infty \underbrace{\Pr[u \in N(v) \mid w_u = w, w_v]}_{= \Pr[x_u \in [\frac{1}{2} - \lambda \frac{w \cdot w_v}{n}, \frac{1}{2} + \lambda \frac{w \cdot w_v}{n}]]} f_{w_u}(w) \mathrm{d}w)\end{aligned}$$

w.l.o.g $x_v = \frac{1}{2}$

Case 1: $w \leq \frac{n}{2\lambda w_v} \Rightarrow \lambda \frac{w \cdot w_v}{n} \leq \frac{1}{2} \longrightarrow = 2\lambda \frac{w \cdot w_v}{n} = \Theta(\frac{w \cdot w_v}{n})$

Case 2: $w > \frac{n}{2\lambda w_v} \Rightarrow \lambda \frac{w \cdot w_v}{n} > \frac{1}{2} \longrightarrow = 1$

GIRG
- n independent vertices
- $x_v \sim \mathcal{U}([0, 1])$
- $w_v \sim \mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3)$
  $f_{w_v}(w) = (\tau - 1) w^{-\tau}$
- u, v adjacent iff
  $\mathrm{dist}(x_u, x_v) \leq \lambda \frac{w_u \cdot w_v}{n}$



Expected Degree ($d = 1$)

- Consider vertex v with weight w_v
- We want to compute $\mathbb{E}[\deg(v) \mid w_v] = \begin{cases} \Theta(n), \text{ if } w_v \geq \frac{n}{2\lambda} \\ \end{cases}$
- Consider X_u for $u \in V \setminus \{v\}$ indicating whether $\{u, v\} \in E$

$$\deg(v) = \sum_{u \in V \setminus \{v\}} X_u$$

$$\mathbb{E}[\deg(v) \mid w_v] = \sum_{u \in V \setminus \{v\}} \mathbb{E}[X_u \mid w_v]$$

$$= \Theta(n \Pr[\{u, v\} \in E \mid w_v])$$

w.l.o.g $x_v = \frac{1}{2}$

$$= \Theta(n \int_1^\infty \Pr[u \in N(v) \mid w_u = w, w_v] f_{w_u}(w) \mathrm{d}w)$$

$$= \Theta\left(n \left(\cancel{\int_1^{\boxed{\frac{n}{2\lambda w_v}}} \frac{w \cdot w_v}{n} f_{w_u}(w) \mathrm{d}w} + \underbrace{\int_{\boxed{\frac{n}{2\lambda w_v}}}^\infty 1 \cdot f_{w_u}(w) \mathrm{d}w}_{} \right)\right)$$

$$= \Pr[w_u \geq \boxed{\frac{n}{2\lambda w_v}}]$$

If $w_v \geq \frac{n}{2\lambda}$, then $\boxed{\frac{n}{2\lambda w_v}} \leq 1$

$$= \Pr[w_u \geq 1] = 1$$

$$= \Theta(n)$$

GIRG
- n independent vertices
- $x_v \sim \mathcal{U}([0, 1])$
- $w_v \sim \mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3)$
  $f_{w_v}(w) = (\tau - 1) w^{-\tau}$
- u, v adjacent iff
  $\mathrm{dist}(x_u, x_v) \leq \lambda \frac{w_u \cdot w_v}{n}$

Expected Degree ($d = 1$)

- Consider vertex v with weight w_v
- We want to compute $\mathbb{E}[\deg(v) \mid w_v] = \begin{cases} \Theta(n), \text{ if } w_v \geq \frac{n}{2\lambda} \\ \end{cases}$
- Consider X_u for $u \in V \setminus \{v\}$ indicating whether $\{u, v\} \in E$

$$\deg(v) = \sum_{u \in V \setminus \{v\}} X_u$$

$$\mathbb{E}[\deg(v) \mid w_v] = \sum_{u \in V \setminus \{v\}} \mathbb{E}[X_u \mid w_v]$$

$$= \Theta(n \Pr[\{u, v\} \in E \mid w_v])$$

w.l.o.g $x_v = \frac{1}{2}$

If $w_v < \frac{n}{2\lambda}$

$$= \Theta(n \int_1^\infty \Pr[u \in N(v) \mid w_u = w, w_v] f_{w_u}(w) \mathrm{d}w)$$

$$= \Theta\left(n\left(\int_1^{\frac{n}{2\lambda w_v}} \frac{w \cdot w_v}{n} f_{w_u}(w)\mathrm{d}w + \underbrace{\Pr\left[w_u \geq \frac{n}{2\lambda w_v}\right]}\right)\right)$$

$$\text{(via CDF of Par)} = \left(\frac{n}{2\lambda w_v}\right)^{-(\tau-1)}$$

$$= \Big(\underbrace{\frac{2\lambda w_v}{n}}_{<1}\Big)^{\tau-1}$$

$$= O\left(\frac{w_v}{n}\right)$$

GIRG

- n independent vertices
- $x_v \sim \mathcal{U}([0, 1])$
- $w_v \sim \text{Par}(\tau - 1, 1)$ for $\tau \in (2, 3)$
  $f_{w_v}(w) = (\tau - 1)w^{-\tau}$
- u, v adjacent iff
  $\text{dist}(x_u, x_v) \leq \lambda \frac{w_u \cdot w_v}{n}$

Expected Degree ($d = 1$)

- Consider vertex v with weight w_v
- We want to compute $\mathbb{E}[\deg(v) \mid w_v] = \begin{cases} \Theta(n), \text{ if } w_v \geq \frac{n}{2\lambda} \\ \Theta(w_v), \text{ otherwise} \end{cases}$ ✓
- Consider X_u for $u \in V \setminus \{v\}$ indicating whether $\{u, v\} \in E$

$$\deg(v) = \sum_{u \in V \setminus \{v\}} X_u$$

$$\mathbb{E}[\deg(v) \mid w_v] = \sum_{u \in V \setminus \{v\}} \mathbb{E}[X_u \mid w_v]$$

$$= \Theta(n \Pr[\{u, v\} \in E \mid w_v])$$

w.l.o.g $x_v = \frac{1}{2}$

If $w_v < \frac{n}{2\lambda}$

$$= \Theta(n \int_1^{\infty} \Pr[u \in N(v) \mid w_u = w, w_v] f_{w_u}(w) \mathrm{d}w)$$

$$= \Theta\left(n \left(\int_1^{\frac{n}{2\lambda w_v}} \frac{w \cdot w_v}{n} f_{w_u}(w) \mathrm{d}w + \Pr[w_u \geq \frac{n}{2\lambda w_v}]\right)\right)$$

$$= \Theta\left(n \int_1^{\frac{n}{2\lambda w_v}} \frac{w \cdot w_v}{n} f_{w_u}(w) \mathrm{d}w\right) + O(w_v)$$

$$= \Theta\left(n \frac{w_v}{n} \int_1^{\frac{n}{2\lambda w_v}} w \cdot (\tau - 1) w^{-\tau} \mathrm{d}w\right) + O(w_v)$$

$$= \Theta\left(w_v \int_1^{\frac{n}{2\lambda w_v}} w^{-(\tau-1)} \mathrm{d}w\right) + O(w_v)$$

$$= \Theta\left(w_v \left[\frac{1}{-(\tau-2)} w^{-(\tau-2)}\right]_1^{\frac{n}{2\lambda w_v}}\right) + O(w_v)$$

$$= \Theta\left(w_v \left[w^{-(\tau-2)}\right]_{\frac{n}{2\lambda w_v}}^1\right) + O(w_v)$$

$$= \Theta\left(w_v \left(1 - \underbrace{\left(\frac{n}{2\lambda w_v}\right)^{-(\tau-2)}}_{< 1 \text{ and } O(1)}\right)\right) + O(w_v)$$

$$= \Theta(w_v)$$

GIRG
- n independent vertices
- $x_v \sim \mathcal{U}([0, 1])$
- $w_v \sim \mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3)$
  $f_{w_v}(w) = (\tau - 1) w^{-\tau}$
- u, v adjacent iff
  $\mathrm{dist}(x_u, x_v) \leq \lambda \frac{w_u \cdot w_v}{n}$

Are GIRGs Realistic?

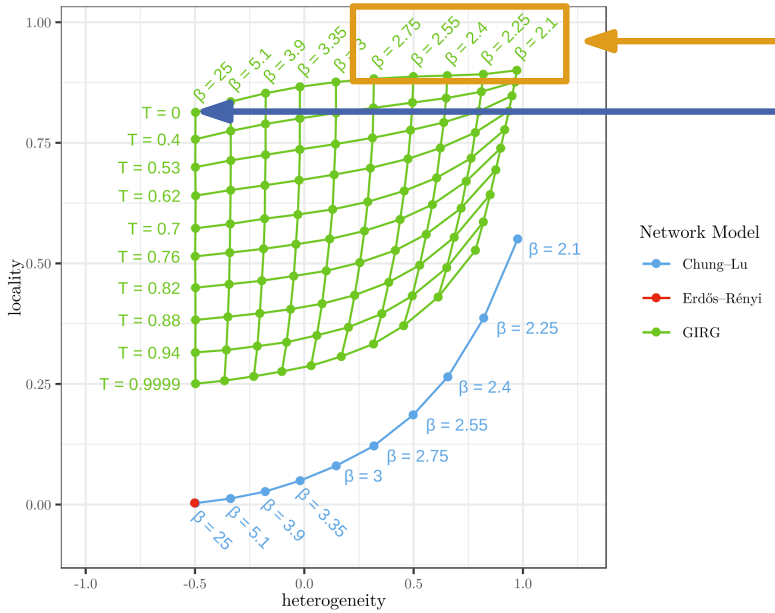
Structural Properties

- Heterogeneity: $\deg(v) \approx w_v$, $w_v \sim \mathrm{Par}(\tau - 1, 1) \rightsquigarrow$ power-law degree distribution ✓
  (also works with other weight distributions)
- Locality (not seen here) ✓

Algorithmic Properties

"On the External Validity of Average-Case Analyses of Graph Algorithms", Bläsius, Fischbeck, ACM Trans. Algorithms 2023

- Setup: GIRGs with varying degrees of heterogeneity and locality (each dot is a graph)



What we considered just now

Basically random geometric graphs

GIRGs without geometry / ER with weights

Are GIRGs Realistic?

Structural Properties

- Heterogeneity: $\deg(v) \approx w_v$, $w_v \sim \mathrm{Par}(\tau - 1, 1) \rightsquigarrow$ power-law degree distribution ✓
- Locality (not seen here) ✓

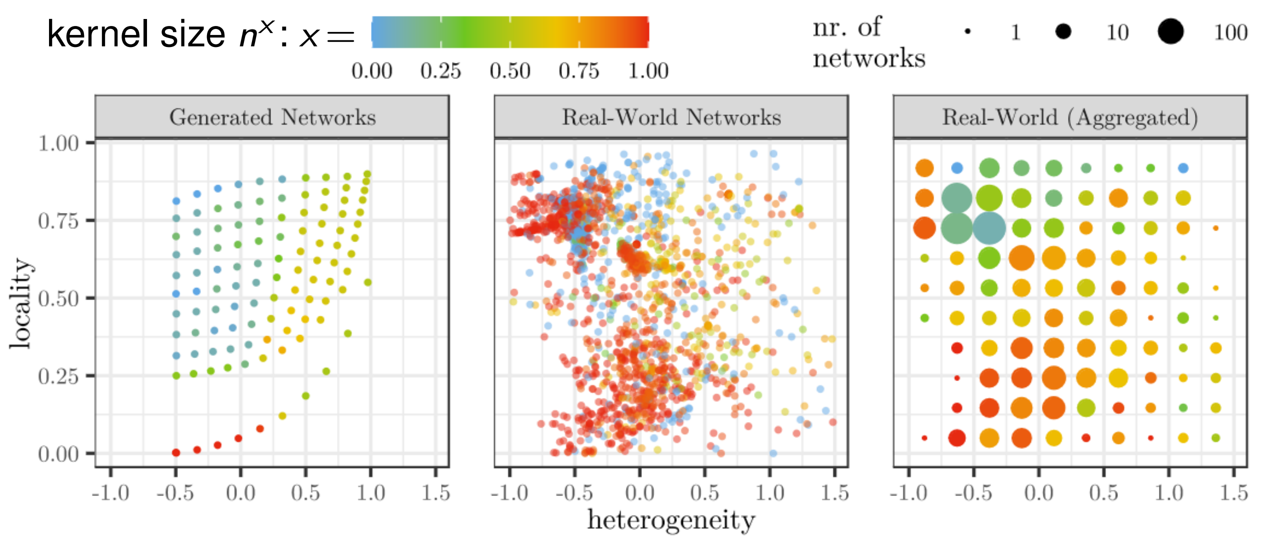
(also works with other weight distributions)

Algorithmic Properties

"On the External Validity of Average-Case Analyses of Graph Algorithms", Bläsius, Fischbeck, ACM Trans. Algorithms 2023

- Setup: GIRGs with varying degrees of heterogeneity and locality (each dot is a graph)
- Measure algorithmic properties on GIRGs and real graphs
  - Bidirectional breadth-first-search
  - Diameter computation via BFS
  - Vertex cover kernel size
  - Louvain clustering algorithm
  - Number of maximal cliques

    rather structural property
  - Chromatic number kernel size



Use GIRGs for average-case analysis!

Vertex Cover Approximation

Vertex Cover

- Given undirected graph $G = (V, E)$
- Find a smallest $S \subseteq V$ such that $\overbrace{G[V \setminus S]}^{\text{(induced subgraph)}}$ is edgeless
- NP-complete

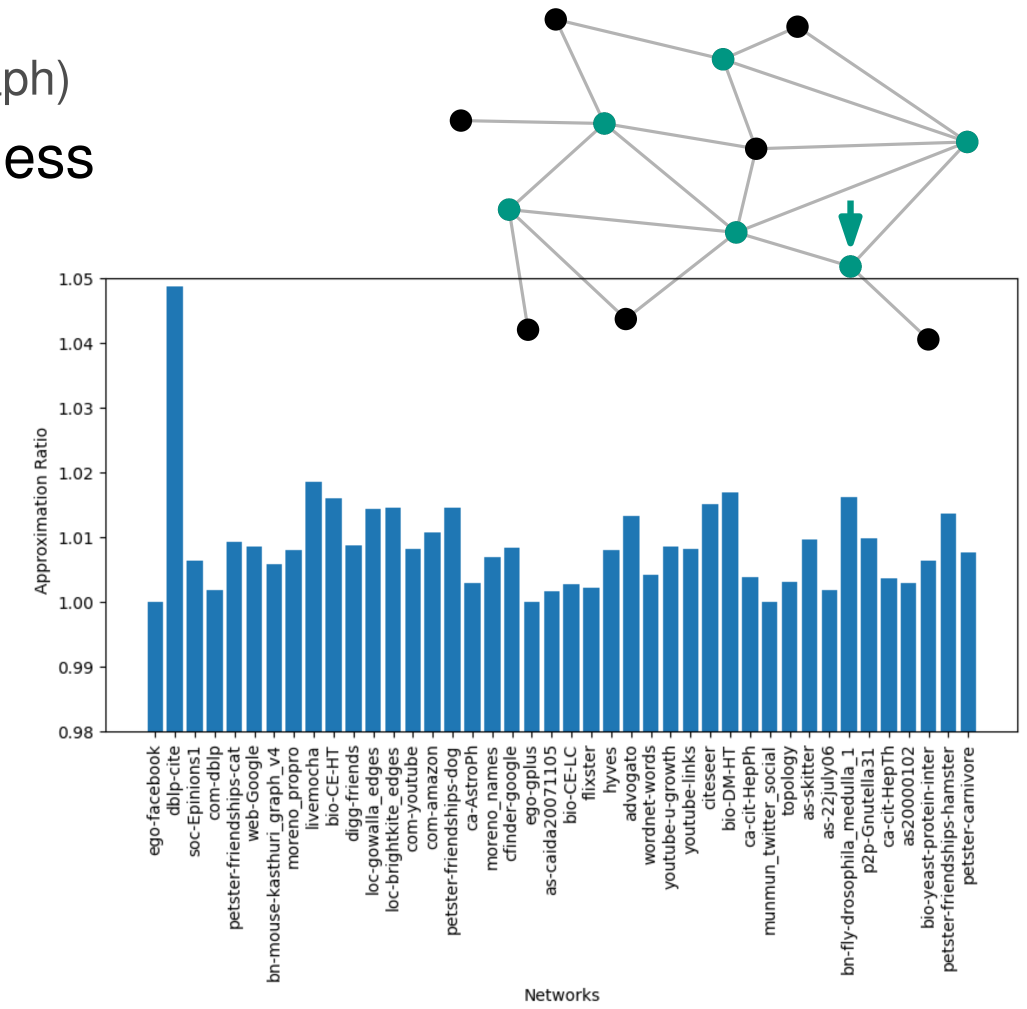
Vertex Cover Approximation

- Find a *small* vertex cover S' *fast*
- Approximation ratio: $r = |S'|/|S|$
- NP-hard to approximate with $r < \sqrt{2}$
- Believed to be NP-hard for $r < 2 - \varepsilon$ for const. ε

Practice

- Simple approximation algorithm repeatedly takes/deletes vertex of largest degree
- Close to optimal ratios on real graphs

"Vertex Cover on Complex Networks", Da Silva, Gimenez-Lugo, Da Silva, IJMPC 2013



Analsysis on GIRGs

(based on) "Efficiently Approximating Vertex Cover on Scale-Free Networks with Underlying Hyperbolic Geometry", Bläsius, Friedrich, K., Algorithmica 2023

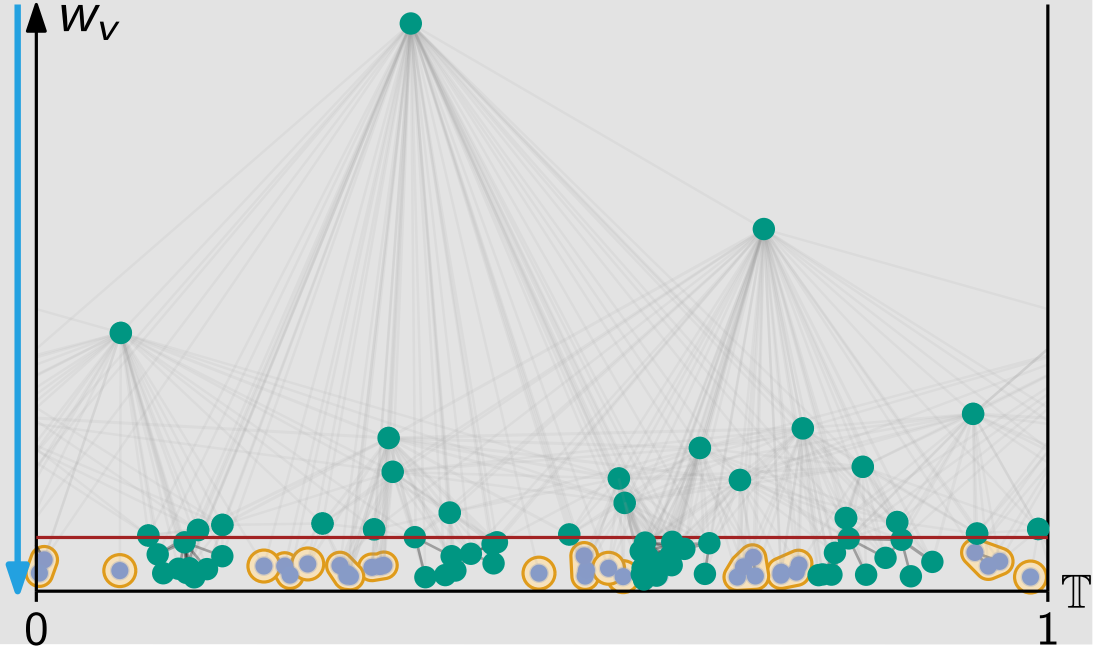
Keep it simple

- Consider vertices in order of decreasing degree in original graph
- Consider vertices in order of decreasing *weight*

Learn from the Model

- Once high-degree vertices are taken/removed, remaining vertices have roughly equal weight/degree
- Greedy algorithm picks vertices at random
- Improve quality by solving $\underbrace{\text{small}}_{\log\log(n)}$ separated components *exactly*
- Two variants
  - Search and solve small components after each greedily taken vertex
  - Take greedy until red line, solve small components exactly, take rest greedy too

This variant yields an upper bound on the quality of the other



Analysis on GIRGs – Approximation Ratio

Theorem: Let G be GIRG with n vertices and m edges. Then, an approximate vertex cover S' of G can be computed in time $O(m \log(n))$ such that the approximation ratio is $(1 + o(1))$ asymptotically almost surely.

Proof *Approximation Ratio*

- Differentiate greedily taken vertices S'_g from ones in exactly solved components S'_e
- For each small component, the optimal solution S cannot contain fewer vertices than S'_e does

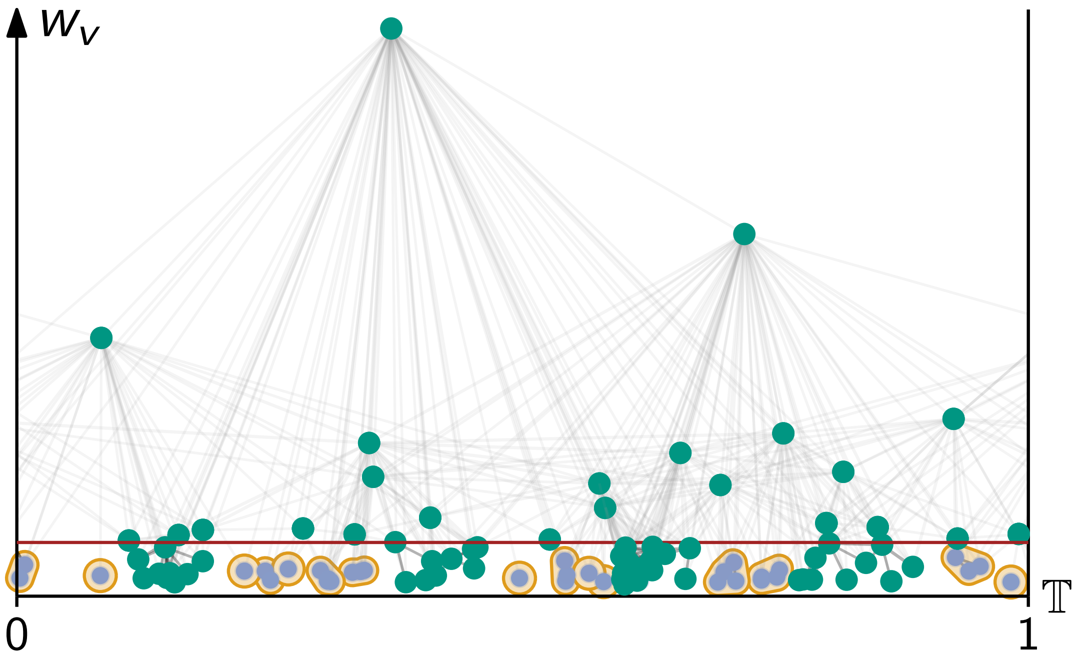
$$\Rightarrow |S'_e| \leq |S|$$

$$\Rightarrow r = \frac{|S'|}{|S|} = \frac{|S'_e| + |S'_g|}{|S|} \leq \frac{|S| + |S'_g|}{|S|} = 1 + \frac{|S'_g|}{|S|}$$

- $|S| = \Omega(n)$ with prob $1 - o(1)$

"Greed is Good for Deterministic Scale-Free Networks", Chauhan et al. FSTTCS 2016

Remains to show: $|S'_g| = o(n)$



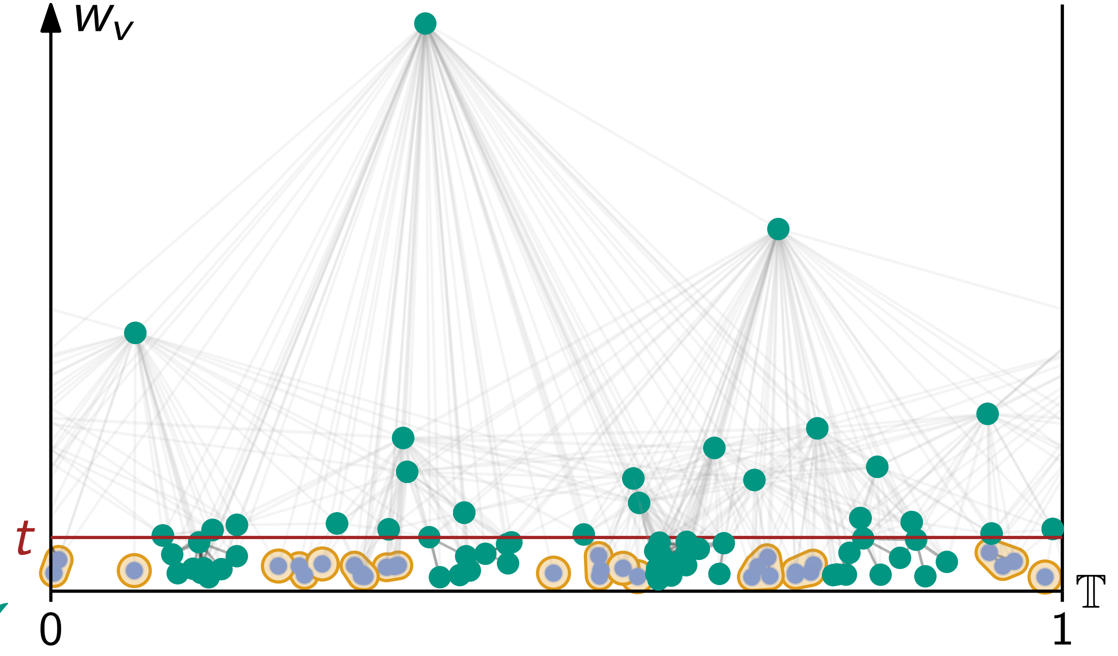
Analysis on GIRGs – Greedy Vertices $\geq t$

Lemma: Let G be a GIRG with n vertices, let $t = \omega(1)$, and let $N_{w \geq t}$ be the number of vertices with weight at least t. Then, $N_{w \geq t} = o(n)$ with probability $1 - O(1/n)$.

GIRG
- n independent vertices
- $w_v \sim \mathrm{Par}(\tau - 1, 1)$ for $\tau \in (2, 3)$

Proof
- Consider random variable $X_v = \mathbb{1}_{\{w_v \geq t\}}$
- $N_{w \geq t}$ is the sum of independent Bernoulli random variables
  $N_{w \geq t} = \sum_{v \in V} X_v$
- Expectation
  $\mathbb{E}[N_{w \geq t}] = \sum_{v \in V} \mathbb{E}[X_v] = n \Pr[w_v \geq t]$
  (via CDF of Par) $= nt^{-(\tau - 1)}$
  $(t = \omega(1), \tau \in (2,3)) = o(n)$
- Since there is a $g(n) \in o(n) \cap \Omega(\log(n))$ with $g(n) \geq \mathbb{E}[N_{w \geq t}]$, Chernoff gives concentration

✓

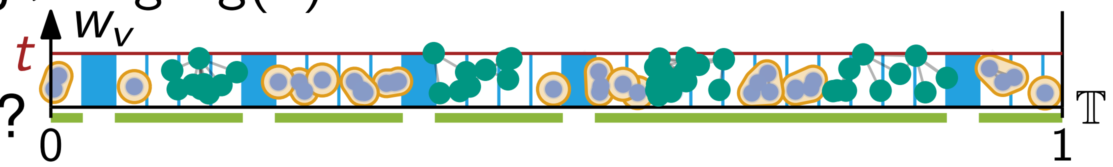


Analysis on GIRGs – Greedy Vertices $< t$

- After (the $o(n)$) vertices with weight $\geq t$ are removed, the graph decomposes into several components
  - Components of size $\leq \log\log(n)$ are solved exactly
  - Larger components are assumed to be taken greedily (need to show: these are $o(n)$)
- Hard to determine how likely it is for a vertex to be in a large component
- Make use of geometry! Overestimate components by counting how many vertices are geometrically very close

Idea

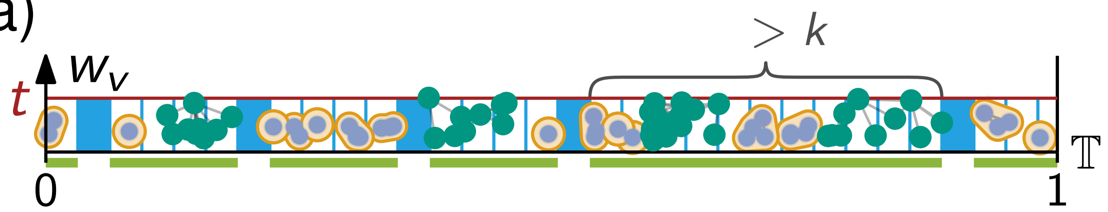
- Discretize ground space into cells such that edges cannot span empty cells
- Use empty cells as delimiters between components
- Regard chains of non-empty cells as one component
- Count all vertices that are in chains containing $> \log\log(n)$ vertices
  (also potentially counting small components)
- When does a chain contain too many vertices?



- After (the $o(n)$) vertices with weight $\geq t$ are removed, the graph decomposes into several components
  - Components of size $\leq \log\log(n)$ are solved exactly
  - Larger components are assumed to be taken greedily (need to show: these are $o(n)$)
- Hard to determine how likely it is for a vertex to be in a large component
- Make use of geometry! Overestimate components by counting how many vertices are geometrically very close

Case 1 Too many cells in long chains, say $> k$ cells

- Unlikely, if cells are small
- Proof via method of bounded differences!
  Total number of cells in long chains does not change much ($\leq 2k+1$) when one cell moves from empty to non-empty (or vice versa)
- Use Poissonization to get rid of dependencies



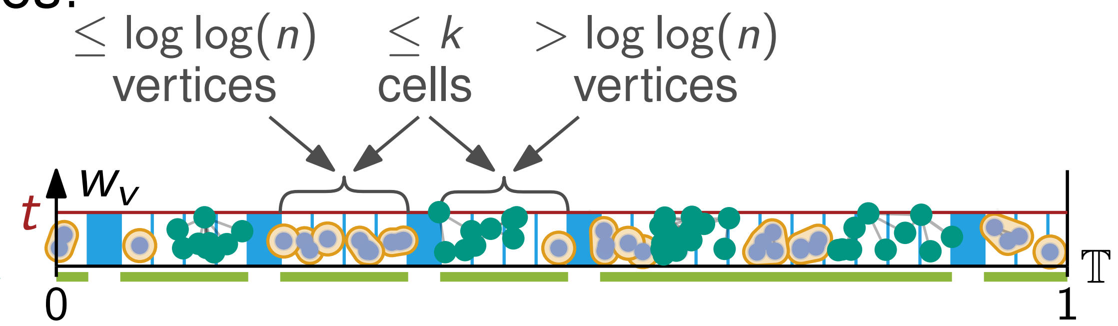
Analysis on GIRGs – Greedy Vertices $< t$

- After (the $o(n)$) vertices with weight $\geq t$ are removed, the graph decomposes into several components
  - Components of size $\leq \log\log(n)$ are solved exactly
  - Larger components are assumed to be taken greedily (need to show: these are $o(n)$)
- Hard to determine how likely it is for a vertex to be in a large component
- Make use of geometry! Overestimate components by counting how many vertices are geometrically very close

Case 2 Short chains ($\leq k$ cells) contain too many vertices

- Unlikely, if cells are small
- Proof via method of *typical* bounded differences!
  - Imagine cells as boxes on conveyor belt
  - Imagine vertices as products
  - Typically not many vertices in few cells

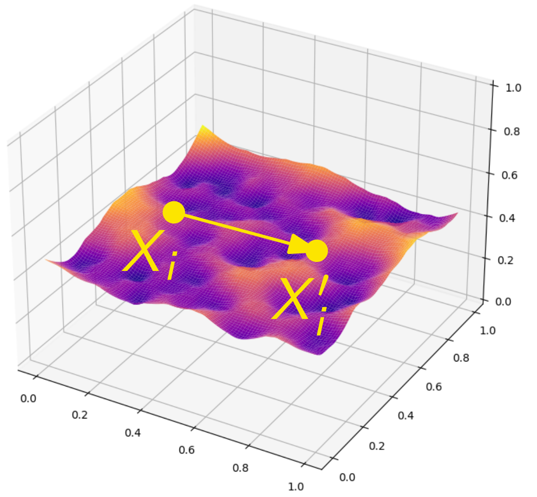
$\rightsquigarrow$ w.h.p., $o(n)$ vertices in large components ✓



Conclusion

Method of Bounded Differences

- Concentration for function of independent random variables
- Bounded differences (“Lipschitz”) condition
  - What is the worst that can happen when changing one input?
- Chernoff-like bound, weakened by sum of squared worst changes
- Useless if worst changes are too large



Method of Typical Bounded Differences

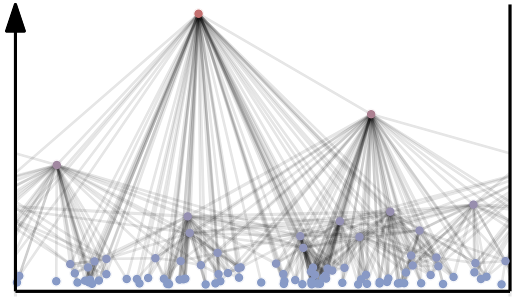
- Define typical event, distinguish worst changes depending on whether event occurred
- Use mitigators to weaken impact of general worst changes
- Pay with probability that typical event does not occur, multiplied with inverse mitigators

Geometric Inhomogeneous Random Graphs

- Pretty realistic graph model (heterogeneity, locality)
- Not too hard to analyze

(not discussed in lecture)

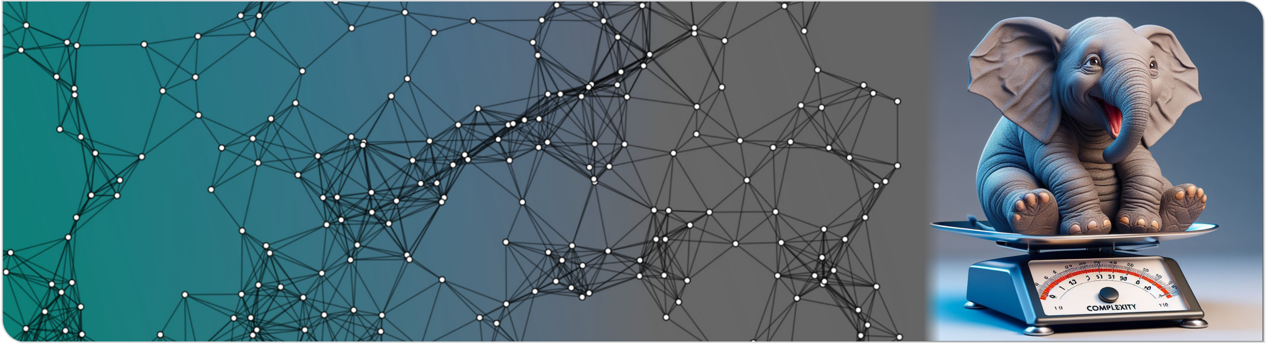
- Used for average-case analysis (e.g. vertex cover approximation)



Karlsruhe Institute of Technology

Probability and Computing – Randomised Complexity Classes

Stefan Walzer, Maximilian Katzmann, (Thomas Worsch) | WS 2023/2024



Outline & Script

The Second Half of the Semester

- Randomised Complexity Classes
  $\hookrightarrow$ full lecture notes by Thomas Worsch
- Game Theory and Yao's Principle
  $\hookrightarrow$ some lecture notes by Thomas Worsch
- Randomised Approximation
  $\hookrightarrow$ full lecture notes by Thomas Worsch
- Streaming Algorithms
- Randomised *Data Structures*
  - Hash Functions
    - application: linear probing hash table
    - application: linear chaining hash table
  - Bloom Filters
  - Cuckoo Hashing
  - The Peeling Algorithm
  - Applications of Peeling

What are you missing?

The lecture by Thomas Worsch also covered

- routing in hypercubes
- an expected $\mathcal{O}(n)$-time randomised MST algorithm
- online algorithms
- random walks
- Markov chains and Metropolis-Hastings
- pseudorandom number generation

Today: Decision Problems Only

Karlsruhe Institute of Technology

- ~~approximation algorithms~~
- ~~average case analysis~~
- ~~data structures~~
- ~~function problems~~
- **decision problems**
  - for some language L such as $L = \text{PRIMES}$
  - decide for input x the question “is $x \in L$?”
  - can you do it in polynomial time?
  - does randomisation help?

Turing machines

(Non-) deterministic Turing machine

- S: finite state set
- B: finite tape alphabet including blank symbol $\square$
- $A \subseteq B - \{\square\}$: input alphabet
- one tape, one head
- transition functions
  - *deterministic:* one
    $\delta : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$
  - *non-deterministic* two (or more)
    $\delta_0, \delta_1 : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$
    (alternatively: general transition *relation*)
  - in states YES and NO: "T halts"
- accepted language
  $L(T) = \{w \in A^+ \mid \exists\text{YES-computation for } w\}$

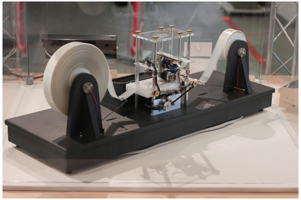


Photo: Rocky Acosta

Probabilistic Turing machine

- definition like non-deterministic TM
- uses δ_0 or δ_1 with probability $1/2$ in each step
- output $T(w)$ is random variable
- difference to NTM:
  - *quantified* non-determinism
  - can study e.g. *probability* of acceptance

When is a PTM polynomial time?

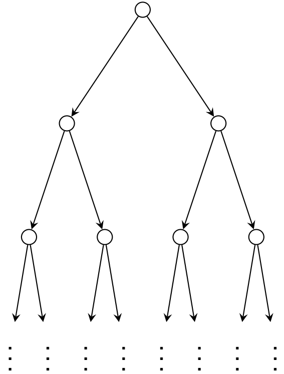
Karlsruhe Institute of Technology

Annoying

Running time for input x is random variable $T(x) \in \mathbb{N} \cup \{\infty\}$.

Simplification for Today: PTM in normal form

- For all inputs of length n, the PTM *halts* and does so after the *same number of steps* $t(n)$. $\hookrightarrow$ this is without loss of generality under weak conditions
- computation tree of a PTM in normal form is complete binary tree of depth $t(n)$.
- call $t(n)$ the *running time*
- PTM runs in *polynomial time*, if $t(n) \leq p(n)$ for a polynomial $p(n)$.
- acceptance probability is the number of accepting computations, divided by $2^{t(n)}$.



“Classic” Complexity Classes

class $\mathcal{C}$	requirement for $L \in \mathcal{C}$
$\mathbf{P}$	polynomial time DTM can decide L
$\mathbf{NP}$	polynomial time NTM can decide L
$\mathbf{PSPACE}$	polynomial space TM can decide L

Complement Classes

For class $\mathcal{C}$ let $\text{co}-\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\} = \{\bar{L} \mid L \in \mathcal{C}\}$, e.g.

- $\mathbf{P} = \text{co}-\mathbf{P}$
- $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co}-\mathbf{NP}$
- relationship between $\mathbf{NP}$ and $\text{co}-\mathbf{NP}$ unknown
- $\mathbf{NP} \cup \text{co}-\mathbf{NP} \subseteq \mathbf{PSPACE}$

Polynomial time reduction from L_1 to L_2

- in polynomial time computable function $f : A^+ \to A^+$, such that
- $\forall w \in A^+ : w \in L_1 \iff f(w) \in L_2$.

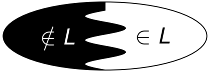
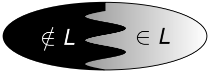
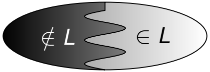
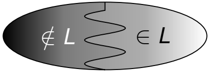
$\hookrightarrow$ then e.g. $L_2 \in \mathbf{NP}$ implies $L_1 \in \mathbf{NP}$.

Hardness

- A language H is *$\mathcal{C}$-hard*, if every language $L \in \mathcal{C}$ can be reduced to H in polynomial time.
- A language is *$\mathcal{C}$-complete*, if it is $\mathcal{C}$-hard and in $\mathcal{C}$.

Probabilistic Complexity Classes

A language L is in class $\mathbf{P}/\mathbf{RP}/\mathbf{BPP}/\mathbf{PP}$, if there exists a probabilistic polynomial time turing machine T such that...

class	name	requirement	visualisation	
$\mathbf{P}$	polynomial time	$\forall w \notin L : \Pr[T(w) = \textsc{yes}] = 0$ $\forall w \in L : \Pr[T(w) = \textsc{yes}] = 1$		no error
$\mathbf{RP}$	randomised polynomial time	$\forall w \notin L : \Pr[T(w) = \textsc{yes}] = 0$ $\forall w \in L : \Pr[T(w) = \textsc{yes}] \geq 1/2$		one-sided error
$\mathbf{BPP}$	bounded-error probabilistic polynomial time	$\forall w \notin L : \Pr[T(w) = \textsc{yes}] < 1/4$ $\forall w \in L : \Pr[T(w) = \textsc{yes}] > 3/4$		two-sided error
$\mathbf{PP}$	probabilistic polynomial time	$\forall w \notin L : \Pr[T(w) = \textsc{yes}] \leq 1/2$ $\forall w \in L : \Pr[T(w) = \textsc{yes}] > 1/2$		two-sided error

$\mathbf{ZPP} := \mathbf{RP} \cap \mathrm{co}{-}\mathbf{RP}$. zero error probabilistic polynomial time

$\hookrightarrow$ requires *two* Turing machines, one for $\mathbf{RP}$, one for $\mathrm{co}{-}\mathbf{RP}$.

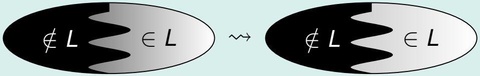
We say a polynomial time PTM is an $\mathbf{RP}$-PTM, $\mathbf{BPP}$-PTM or $\mathbf{PP}$-PTM if it is of the corresponding form.

Probability Amplification

Karlsruhe Institute of Technology

Theorem

Instead of "$1/2$" we can use "$1 - 2^{-q(n)}$" in the definition of $\mathbf{RP}$ without affecting the class.



Proof.

Let T be the Turing machine witnessing $L \in \mathbf{RP}$.
By running T independently $q(n)$ times the error probability is $2^{-q(n)}$.
Running time increases by polynomial factor $q(n)$. $\square$

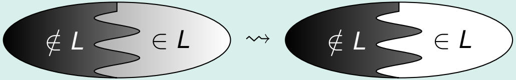
for $i = 1$ *to* $q(n)$ **do**
    if $T(w) = \text{YES}$ **then**
        return YES
return NO

Probability Amplification (2)

Karlsruhe Institute of Technology

Theorem

Instead of “$1/4$” and “$3/4$” we can use “$2^{-q(n)}$” and “$1-2^{-q(n)}$” in the definition of $\mathbf{BPP}$ without affecting the class.

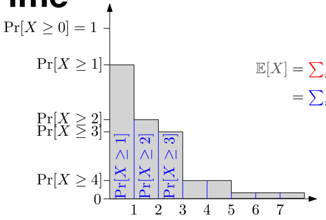


Proof.

Recommended (Bonus) Exercise.

$\hookrightarrow$ solution in lecture notes by Thomas Worsch

ZPP: Zero-Error-Probabilistic Polynomial Time



$$\mathbb{E}[X] = \sum_{i \geq 0} i \cdot \Pr[X = i] = \sum_{i \geq 1} \Pr[X \geq i]$$

Theorem: $L \in \mathbf{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \mathbf{ZPP} := \mathbf{RP} \cap \mathrm{co}-\mathbf{RP}$ then there exists a PTM that

- decides L with no error
- has *expected* polynomial running time
  - $\hookrightarrow$ this PTM is not in normal form

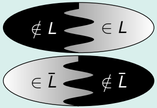
Proof

Let T be an $\mathbf{RP}$-PTM for L with running time $p(n)$.

$\hookrightarrow$ never errs for $x \notin L$

Let $\bar{T}$ be an $\mathbf{RP}$-PTM for $\bar{L}$ with running time $p(n)$.

$\hookrightarrow$ never errs for $x \notin \bar{L}$



repeat
  $r_1 \leftarrow T(w)$
  $r_2 \leftarrow$ **not** $\bar{T}(w)$
until $r_1 = r_2$
return r_1

- T and $\bar{T}$ never *both* answer incorrectly $\Rightarrow$ we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.

$$\mathbb{E}[\text{running time}] \leq 2p(|w|) \cdot \mathbb{E}[\#\text{rounds}] \leq 2p(n) \cdot \sum_{i \geq 1} \Pr[\#\text{rounds} \geq i] \leq 2p(n) \cdot \sum_{i \geq 1} 2^{-(i-1)} = 2p(n) \cdot \sum_{i \geq 0} 2^{-i} = 4p(n). \quad \square$$

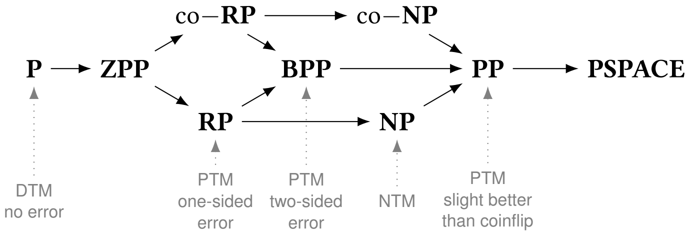
Complete Problems?

Remark

The classes $\mathbf{RP}$, $\mathrm{co}-\mathbf{RP}$ and $\mathbf{BPP}$ are not believed to have complete problems unless, e.g. $\mathbf{BPP} = \mathbf{P}$.
Underlying issue: "T is a $\mathbf{BPP}$-PTM" is undecidable.

Content

Beziehungen zwischen Komplexitätsklassen



Theorems

- $\mathbf{P} \subseteq \mathbf{ZPP}$
- $\mathbf{ZPP} \subseteq \mathbf{RP}$ and $\mathbf{ZPP} \subseteq \mathrm{co}{-}\mathbf{RP}$
- $\mathbf{RP} \subseteq \mathbf{NP}$ and $\mathrm{co}{-}\mathbf{RP} \subseteq \mathrm{co}{-}\mathbf{NP}$
- $\mathbf{RP} \subseteq \mathbf{BPP}$ and $\mathrm{co}{-}\mathbf{RP} \subseteq \mathbf{BPP}$
- $\mathbf{BPP} \subseteq \mathbf{PP}$

proved in the following (rest is exercise):

- $\mathbf{NP} \subseteq \mathbf{PP}$ and $\mathrm{co}{-}\mathbf{NP} \subseteq \mathbf{PP}$
- $\mathbf{PP} \subseteq \mathbf{PSPACE}$

“Typecasting” Turing Machines

DTM as NTM

Given DTM T with transition function δ, consider NTM T' with transition functions $\delta_0 = \delta_1 = \delta$.
$\hookrightarrow$ No change in behaviour: $T(w) = \text{YES} \Leftrightarrow T'(w) = \text{YES}$.

NTM as PTM

Given NTM T, we can reinterpret it as a PTM T':

$$T(w) = \text{YES} :\Leftrightarrow \exists\text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] > 0$$
$$T(w) = \text{NO} :\Leftrightarrow \nexists\text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] = 0$$

PTM as DTM

Given PTM T, we can view it as DTM T' with random bitstring $b = b_1 b_2 \ldots$ as additional input.
In step i transition function δ_{b_i} is used.

$$\Pr[T(w) = \text{YES}] = \Pr_{b_1, b_2, \ldots \sim Ber(1/2)}[T'(w, b) = \text{YES}].$$

Theorem: $\mathbf{NP} \subseteq \mathbf{PP}$ (analogously $\mathrm{co-}\mathbf{NP} \subseteq \mathbf{PP}$)

i.e. show that each $L \in \mathbf{NP}$ satisfies $L \in \mathbf{PP}$

Karlsruhe Institute of Technology

Have: NTM T certifying that $L \in \mathbf{NP}$

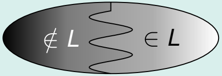
$w \in L \Leftrightarrow \exists$YES-computation for T and w

Use the NTM T as a PTM T':

$\forall w \notin L : \Pr[T'(w) = \text{YES}] = 0$

$\forall w \in L : \Pr[T'(w) = \text{YES}] > 0$

Want: PTM T'' certifying that $L \in \mathbf{PP}$



$\forall w \notin L : \Pr[T''(w) = \text{YES}] \leq 1/2$

$\forall w \in L : \Pr[T''(w) = \text{YES}] > 1/2$

T'' achieves this shift with a simple trick

$r \leftarrow T'(w)$ // T' is T as PTM

if $r = \text{YES}$ **then**

  return YES

else

  sample $b \sim \mathcal{U}(\{\text{YES}, \text{NO}\})$ // coinflip

  return b

Theorem: $\mathbf{PP} \subseteq \mathbf{PSPACE}$

i.e. show that each $L \in \mathbf{PP}$ satisfies $L \in \mathbf{PSPACE}$

Proof

- Let T a $\mathbf{PP}$-PTM for L with running time $p(n)$.
- Consider DTM T' that simulates T for given w and random choices $b_1 b_2 \dots b_{p(n)}$.
- Consider DTM T'' that for input w runs $T'(w, b_1 b_2 \dots b_{p(n)})$ for all $2^{p(n)}$ possible $b_1 b_2 \dots b_{p(n)}$. Return YES if T' returns YES in majority of cases.
- space complexity:
  - $p(n)$ bits for counter a
  - $p(n)$ bits for $b_1, \dots, b_k$
  - $\mathcal{O}(p(n))$ space for simulating T (can only use $p(n)$ space in its $p(n)$ steps)

$\hookrightarrow$ T'' decides L in space $\mathcal{O}(p(n))$ (and time $\Omega(2^{p(n)})$). $\square$

```
n ← |w|
k ← p(n)
a ← 0 // k-bit counter
for b_1 ... b_k ← 00...0 to 11...1 do
    r ← T'(w, b_1 ... b_k)
    if r = YES then
        a ← a + 1
if a > 2^(k−1) then
    return YES
else
    return NO
```

Conclusion

Karlsruhe Institute of Technology

$$\begin{tikzcd}
 & & \mathrm{co-}\mathbf{RP} \arrow[rr] \arrow[dr] & & \mathrm{co-}\mathbf{NP} \arrow[dr] & & \\
\mathbf{P} \arrow[r] & \mathbf{ZPP} \arrow[ur] \arrow[dr] & & \mathbf{BPP} \arrow[rr] & & \mathbf{PP} \arrow[r] & \mathbf{PSPACE} \\
 & & \mathbf{RP} \arrow[rr] \arrow[ur] & & \mathbf{NP} \arrow[ur] & &
\end{tikzcd}$$

What we learned – not much

- Only "obvious" inclusions known
  - $\hookrightarrow$ e.g. one-sided error vs. two-sided error
- since $\mathbf{P} \stackrel{?}{=} \mathbf{PSPACE}$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
  - obviously: in $\mathrm{co-}\mathbf{NP}$.
  - 1976: in $\mathrm{co-}\mathbf{RP}$ (Rabin).
  - 1987: in $\mathbf{RP}$, hence in $\mathbf{ZPP}$ (Adleman, Huang).
  - 2002: in $\mathbf{P}$ (Agrawal, Kayal, Saxena).

A boring topic?

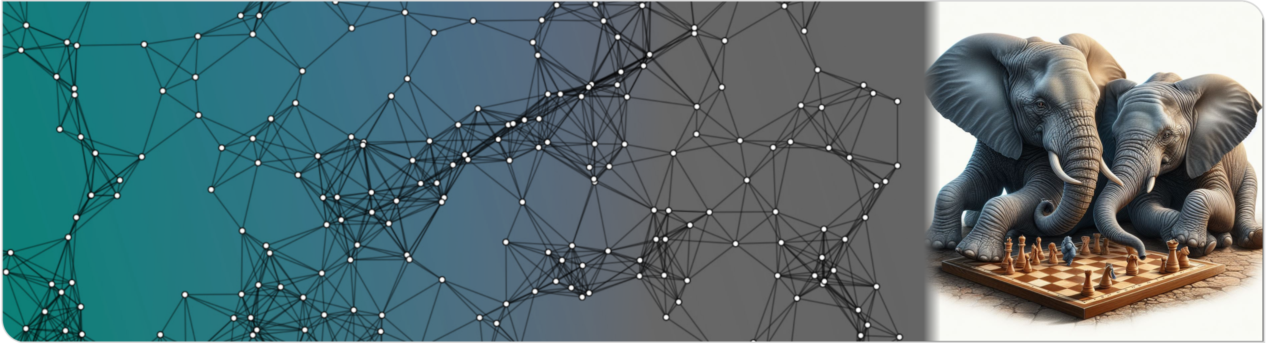
- People believe $\mathbf{BPP} = \mathbf{P}$
  - $\hookrightarrow$ "each $\mathbf{BPP}$ algorithm can be fully derandomised"
- $\mathbf{PP}$ is somewhat esoteric
  - $\hookrightarrow$ no interesting randomised classes remain?
- quantum computing may change the story. People suspect $\mathbf{NP} \not\subseteq \mathbf{BQP} \not\subseteq \mathbf{NP}$
  - $\hookrightarrow$ https://en.wikipedia.org/wiki/BQP

Anhang: Mögliche Prüfungsfragen

- Definiere: Was ist eine PTM? Was ist der Unterschied zu einer NTM?
- Definiere die Komplexitätsklassen $\mathbf{RP}$, $\mathrm{co}-\mathbf{RP}$ $\mathbf{BPP}$, $\mathbf{PP}$, $\mathbf{ZPP}$.
- Inwiefern spielen die Konstanten von $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$, die in den Definitionen vorkommen, einen Rolle? Inwiefern sind sie egal?
- Inwiefern steht die Klasse $\mathbf{ZPP}$ mit dem Konzept eines Las-Vegas Algorithmus in Verbindung? Wie sehen die Umwandlungen in die eine Richtung (Vorlesung) und in die andere Richtung (Übung) aus?
- Welche Inklusionsbeziehungen zwischen diesen Komplexitätsklassen sind bekannt?
- Begründe jede dieser Inklusionsbeziehungen. (In der tatsächlichen Prüfung würde man sich aus Zeitgründen nur eine oder zwei herausgreifen.)
- Gibt es Inklusionsbeziehungen von denen man weiß, dass sie strikt sind? Gibt es Klassen, von denen Experten vermuten, dass sie in Wirklichkeit identisch sind?

Probability and Computing – Lower bounds using Yao's Principle

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Lecture Notes by Worsch

Some of this lecture's content is covered in Thomas Worsch's notes from 2019.

Content

Content

Content

Prisoner's Dilemma

		👺	
		🤐	🗨
😈	🤐	-1\-1	-3\0
	🗨	0\-3	-2\-2

Setting

- *strategies* 🤐 and 🗨 available to both *players*
- table shows *payoffs* for players depending on chosen strategies
- here: always better to choose 🗨
  $\hookrightarrow$ pair (🗨, 🗨) is unique *equilibrium*

Definition: Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game

		🐱	
		🧀	🧶
🐭	🧀	-4\2	2\1
	🕳	0\0	0\1

Someone always regrets their decision

🐭	🐱	reaction
🧀	🧀	🐭 should have played 🕳
🕳	🧀	🐱 should have played 🧶
🕳	🧶	🐭 should have played 🧀
🧀	🧶	🐱 should have played 🧀

$\hookrightarrow$ No combination of *pure* strategies is an *equilibrium*.

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

Nash Equilibria

Karlsruhe Institute of Technology

What a Game *is*

- Finite sets S_1, S_2 of *pure strategies*.
- Utility functions $u_1, u_2 : S_1 \times S_2 \to \mathbb{R}$.

How a Game is played

- Players pick a strategy simultaneously
  $\hookrightarrow$ gives pair $(s_1, s_2) \in S_1 \times S_2$.
- Player 1 gets payoff $u_1(s_1, s_2)$ and player 2 gets $u_2(s_1, s_2)$.

Existence of Mixed-Strategy Nash Equilibria

There exist distributions $\mathcal{S}_1$ on S_1 and $\mathcal{S}_2$ on S_2, called *mixed strategies* such that $(\mathcal{S}_1, \mathcal{S}_2)$ is an equilibrium:

player 1 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1, s_2 \sim \mathcal{S}_2}[u_1(s_1, s_2)] = \max_{s_1 \in S_1} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[u_1(s_1, s_2)]$.

player 2 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1, s_2 \sim \mathcal{S}_2}[u_2(s_1, s_2)] = \max_{s_2 \in S_2} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[u_2(s_1, s_2)]$.

Remark: Theorem holds for $n \geq 3$ players as well.

Karlsruhe Institute of Technology

Nash Equilibrium in Cat & Mouse Game

		🐱	
		🧀	🧶
🐭	🧀	-4\2	2\1
	🕳	0\0	0\1

Equilibrium

$\mathcal{S}_{🐭} = \{$🧀 $: \frac{1}{2},$ 🕳 $: \frac{1}{2}\}$

$\mathcal{S}_{🐱} = \{$🧀 $: \frac{1}{3},$ 🧶 $: \frac{2}{3}\}$

Verification of Equilibrium Property: Calculating Expected Payoffs

for 🐭:

- playing 🧀 gives expected payoff $\frac{1}{3} \cdot (-4) + \frac{2}{3} \cdot 2 = 0$
- playing 🕳 gives expected payoff $\frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 0 = 0$
- playing $\mathcal{S}_{🐭}$ is a mix of both
  $\hookrightarrow$ also expected payoff 0.

for 🐱:

- playing 🧀 gives expected payoff $\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 0 = 1$
- playing 🧶 gives expected payoff $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$
- playing $\mathcal{S}_{🐱}$ is a mix of both
  $\hookrightarrow$ also expected payoff 1.

Content

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

		🤓 🪨	🤓 📃	🤓 ✂️
👧	🪨	0	-1	1
👧	📃	1	0	-1
👧	✂️	-1	1	0

Unique equilibrium of 🪨 📃 ✂️

$\mathcal{S}_1 = \mathcal{S}_2 = \{$ 🪨 $: \frac{1}{3},$ 📃 $: \frac{1}{3},$ ✂️ $: \frac{1}{3} \}$

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

	-1	1	-1
	1	-1	1

Equilibria of

Work it out yourself!

Nash Equilibria for Two-Player Zero-Sum Games

Karlsruhe Institute of Technology

Nash's Theorem (1950), Special Case for Two-Player Zero-Sum Games

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition: When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n \times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

(where $\mathcal{S}_1$ and $\mathcal{S}_2$ are distributions on $[n]$ and $[m]$, respectively)

Intuition

No first-mover disadvantage if

- first player plays mixed strategy
- second player (wlog) pure strategy

Next: Proof of Loomis' Theorem assuming Nash's Theorem.

Lemma: First Mover's Disadvantage

Karlsruhe Institute of Technology

Lemma Φ: Exchanging min and max

Let X and Y be sets and $u : X \times Y \to \mathbb{R}$ a function. In our setting[1]

$$\max_{y \in Y} \min_{x \in X} u(x, y) \leq \min_{x \in X} \max_{y \in Y} u(x, y)$$

[1] In general min and max may not be well-defined...

Proof.

$$\max_{y \in Y} \min_{x \in X} u(x, y) = \min_{x \in X} u(x, y^*) \leq \min_{x \in X} \max_{y \in Y} u(x, y).$$ □

Relevance

Being the second player to choose is never a disadvantage.

Lemma: When *pure* strategies are sufficient

Karlsruhe Institute of Technology

Lemma Δ: Minima over sets of distributions

Let X be a set and $u : X \to \mathbb{R}$ a function. Let $\mathbf{D}$ be the set of all distributions on X. In our setting:

$$\min_{x \in X} u(x) = \min_{\mathcal{X} \in \mathbf{D}} \mathbb{E}_{x \sim \mathcal{X}}[u(x)].$$

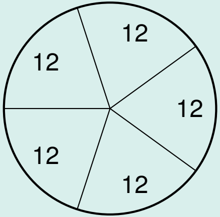
Relevance for us

The last player to choose a strategy may always chose a pure strategy.

Proof by Example

Here is a list of numers $X = \{12, 42, 73, 101\}$.

- Task 1: Find the minimum!
  $\hookrightarrow$ Duh, it's 12.
- Task 2: Design a wheel of fortune involving only numbers from X with minimum expectation!
  $\hookrightarrow$ Duh, take 12 everywhere.



Proof of Loomis's Theorem

Let $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$ be the mixed strategies from Nash's Theorem.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n\times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2\in[m]} \mathbb{E}_{s_1\sim\mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1\in[n]} \mathbb{E}_{s_2\sim\mathcal{S}_2}[M_{s_1,s_2}]$$

$$\begin{aligned}
&\min_{\mathcal{S}_2} \max_{s_1\in[n]} \mathbb{E}_{s_2\sim\mathcal{S}_2}[M_{s_1,s_2}] \\
\le\ & \max_{s_1\in[n]} \mathbb{E}_{s_2\sim\mathcal{S}_2^*}[M_{s_1,s_2}] \\
=\ & \min_{s_2\in[m]} \mathbb{E}_{s_1\sim\mathcal{S}_1^*}[M_{s_1,s_2}] && \text{Nash's Theorem} \\
=\ & \min_{\mathcal{S}_2} \mathbb{E}_{s_1\sim\mathcal{S}_1^*, s_2\sim\mathcal{S}_2}[M_{s_1,s_2}] && \text{Lemma } \Delta \\
\le\ & \max_{\mathcal{S}_1} \min_{\mathcal{S}_2} \mathbb{E}_{s_1\sim\mathcal{S}_1, s_2\sim\mathcal{S}_2}[M_{s_1,s_2}] \\
=\ & \max_{\mathcal{S}_1} \min_{s_2\in[m]} \mathbb{E}_{s_1\sim\mathcal{S}_1}[M_{s_1,s_2}] && \text{Lemma } \Delta \\
\le\ & \min_{\mathcal{S}_2} \max_{\mathcal{S}_1} \mathbb{E}_{s_1\sim\mathcal{S}_1, s_2\sim\mathcal{S}_2}[M_{s_1,s_2}] && \text{Lemma } \Phi\text{: moving first no } \textit{disadvantage} \\
=\ & \min_{\mathcal{S}_2} \max_{s_1\in[n]} \mathbb{E}_{s_2\sim\mathcal{S}_2}[M_{s_1,s_2}] && \text{Lemma } \Delta
\end{aligned}$$

Start and end with same term. Hence all "$\le$" are "$=$". Hence terms of interest are "$=$". □

Content

Algorithm Design as a 2-Player Zero-Sum Game

Karlsruhe Institute of Technology

Setting

For a given computational problem P let

- **Algos**: *finite* set of deterministic algorithms
- **Inputs**: *finite* set of inputs
- $C(A, I) \in \mathbb{R}$ cost of $A \in$ **Algos** on $I \in$ **Inputs**.

A Two-Player Zero-Sum Game

- Designer chooses (randomised) algorithm, i.e. a distribution on **Algos**.
  $\hookrightarrow$ Goal: Minimise (expected) cost.
- Adversary chooses (randomised) input, i.e. a distribution on **Inputs**.
  $\hookrightarrow$ Goal: Maximise (expected) cost.

Example: Sorting

For given $n \in \mathbb{N}$ (finite, though possibly $n \to \infty$ later)

- $P =$ "sort n numbers comparison-based"
- $C(A, I) =$ # of comparisons of A for input I
- **Inputs** $= S_n$ //permutations of $[n]$
- **Algos** $=$ e.g. suitable set of decision trees

Sorting (x, y, z)

Algorithm Designer	Adversary: $(1,2,3)$	$(3,1,2)$	$(2,3,1)$	...
$x < y$ then $y < z$ then $z < x$	2	3	3	
$y < z$ then $z < x$ then $x < y$	3	2	3	
...				

Recall: Exercise Sheet 0, Exercise 1.

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)]$$ designer moves first

$$\overset{\text{Loomis}}{=} \max_{\mathcal{I} \text{ dist. on } \textbf{Inputs}} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)]$$ adversary moves first

Yao's Principle: (Upper and) Lower Bounds on $\mathcal{C}$

Let $\mathcal{A}_0$ be a distribution on **Algos** and $\mathcal{I}_0$ a distribution on **Inputs**. Then

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \geq \mathcal{C} \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Tightness: Loomis implies that "$=$" is possible.

$\hookrightarrow$ Can attain lower bounds on $\mathcal{C}$ by thinking about deterministic algorithm only!

Content

Computational Problem: $\bar{\wedge}$-Tree-Evaluation

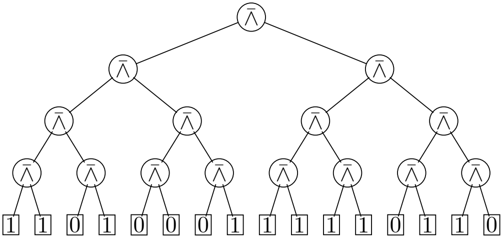
Problem: Evaluate $\bar{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$



Computational Problem: $\bar{\wedge}$-Tree-Evaluation

Karlsruhe Institute of Technology

Problem: Evaluate $\bar{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

Example and possible formalisation of **Algos** (that we won't use)

Each $A \in$ **Algos** corresponds to a *decision tree*. In the example:

- $C(A, (1, 0, 1, 0)) = 4$
- $C(A, (0, 1, 0, 1)) = 2$

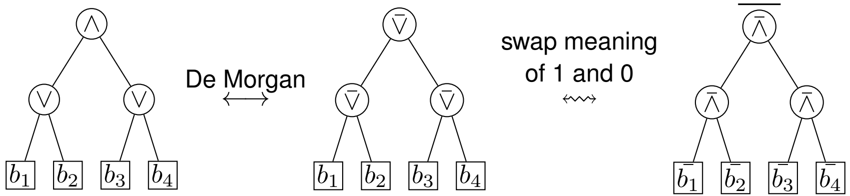
Each leaf queried at most once per path
$\Rightarrow$ depth $\leq n \Rightarrow |\textbf{Algos}| < \infty$

What we already know

Karlsruhe Institute of Technology

$\wedge$-$\vee$-trees are $\bar{\vee}$-trees are $\bar{\wedge}$-trees

See exercise sheet 1 (“Die Wälder von NORwegen”)



What we already know

Karlsruhe Institute of Technology

$\wedge$-$\vee$-trees are $\bar{\vee}$-trees are $\bar{\wedge}$-trees

See exercise sheet 1 ("Die Wälder von NORwegen")

Deterministic Query Complexity is n (Lecture 1, Slide 8)

For all $A \in$ **Algos** there exists $I \in$ **Inputs** such that $C(A, I) = n$.

Randomised Query Complexity is $\mathcal{O}(n^{\log_4(3)}) \approx \mathcal{O}(n^{0.792})$ (Lecture 1, Slide 10)

Let $\mathcal{A}$ be the randomised algorithm that evaluates one of the two depth $d-1$ subtrees at random (recursively) and, if that yields 1, also evaluates the other subtree (recursively).

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \mathcal{O}(3^{d/2}) = \mathcal{O}(n^{\log_4(3)}).$$

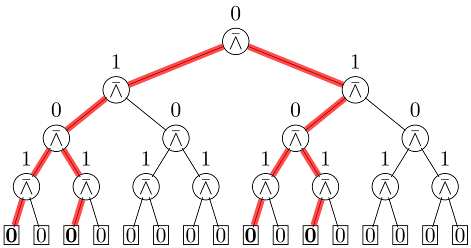
Goal: Show lower bound of $\Omega(\varphi^d) \approx \Omega(n^{0.694})$ using Yao's Principle (φ is the golden ratio).

Remark: actual complexity is $\Theta(n^{\log_4(3)})$, but that's more difficult.

Warm Up: A simple lower bound

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.



Proof

- in the end A knows that the root is 0.
- knowing a 0 requires knowing that both children are 1.
- Knowing a 1 requires knowing of one child that it is 0.

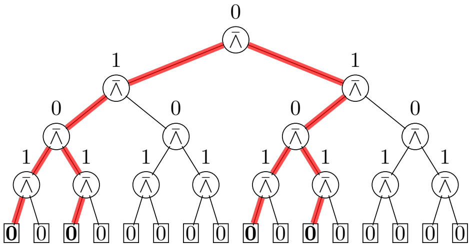
$\hookrightarrow$ A knows of $\geq 2^{d/2}$ leafs that they are 0 and must have checked them.

Warm Up: A simple lower bound

Karlsruhe Institute of Technology

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.

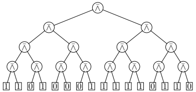


Corollary: Randomised Complexity is $\Omega(\sqrt{n})$

$$\begin{aligned}
\mathcal{C} &= \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \\
&\geq \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, (0, \ldots, 0))] \\
&\geq \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \mathbb{E}_{A \sim \mathcal{A}}[2^{d/2}] \\
&\geq 2^{d/2} = 2^{\log_2(n)/2} = n^{1/2}.
\end{aligned}$$

Note Yao's spirit: Lower bound on *randomised complexity* from result on *deterministic algorithms*.

A stronger lower bound



Karlsruhe Institute of Technology

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Lemma

If $p_0 = \frac{\sqrt{5}-1}{2}$ and φ is the golden ratio then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Corollary: $\mathcal{C} = \Omega(\varphi^d) \approx \Omega(n^{0.694})$

$$\mathcal{C} \overset{\text{Yao}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] \overset{\text{Tarsi}}{=} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$$

$$\overset{\text{Lemma}}{=} \varphi^d = \varphi^{\log_2 n} = n^{\log_2 \varphi} \approx n^{0.694}.$$

Independent Bernoulli Inputs

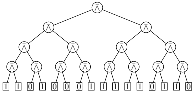
Let $\mathcal{I}_p = Ber(p)^n$ be the distribution where leafs are assigned independently values with distribution $Ber(p)$.

Deterministic Algorithm

Algorithm simpleEval(T)**:**

- **if** $T = \text{leaf}(b)$ **then**
  - **return** b
- **else**
  - $(T_\ell, T_r) \leftarrow T$
  - **if** *simpleEval(T_ℓ) = 0* **then**
    - **return** 1
  - **else**
    - **return** $\neg$simpleEval(T_r)

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Lemma

If $p_0 = \frac{\sqrt{5}-1}{2}$ and φ is the golden ratio then

$$\mathbb{E}_{I\sim\mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1+p_0)^d = \varphi^d.$$

Proof (cf. Exercise “Die Wälder von NORwegen”)

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim Ber(p_0)$ then $a \bar{\wedge} b \sim Ber(1 - p_0^2) = Ber(p_0)$.
- For $I \sim \mathcal{I}_{p_0}$ the probability that an *internal* tree node evaluates to 1 is p_0.
- Let $c_d := \mathbb{E}_{I\sim\mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$ for trees of depth d. Then
  - $c_0 = 1$ // tree of depth 0 is just the leaf
  - $c_d = c_{d-1} + p_0 \cdot c_{d-1} = (1+p_0)c_{d-1} \overset{\text{Ind.}}{=} (1+p_0)(1+p_0)^{d-1} = (1+p_0)^d$
    // Always one recursive call, with probability p a second one.

Deterministic Algorithm

Algorithm simpleEval(T)**:**
  if $T = \text{leaf}(b)$ **then**
    return b
  else
    $(T_\ell, T_r) \leftarrow T$
    if *simpleEval(T_ℓ) = 0* **then**
      return 1
    else
      return $\neg$simpleEval(T_r)

Content

Tarsi's Theorem

Karlsruhe Institute of Technology

Theorem (Tarsi 1984)

For any $p \in [0, 1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

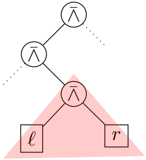
$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Proof idea:

- Take optimal Algorithm A.
- Transform A into simpleEval step by step.
- Show: Expected query complexity never increases.

Lemma: Evaluating Superleafs like simpleEval

Karlsruhe Institute of Technology



Definition: Superleafs

A *superleaf* consists of two sibling leafs and their parent.

Lemma

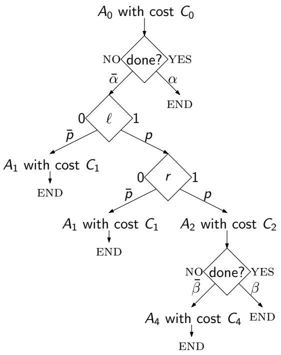
For any $p \in [0,1]$ and any $A \in$ **Algos** there exists $A' \in$ **Algos** such that

- $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$
- A' behaves on any superleaf $T = (\ell, r)$ like simpleEval:
  - i never visits r before ℓ
  - ii never visits r if $\ell = 0$
  - iii immediately visits r after visiting ℓ if $\ell = 1$

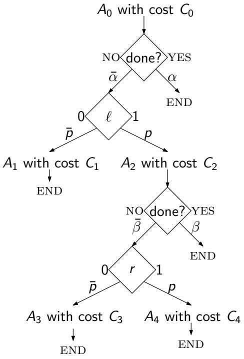
Proof Idea

- We fix every superleaf one by one. Let T be superleaf that needs fixing.
- Property i: Switch roles of ℓ and r if needed. Does not change the expected cost.
- Property ii: r does not contribute to result. Not visiting r *reduces* expected cost.
- Property iii: More difficult. See next slide.

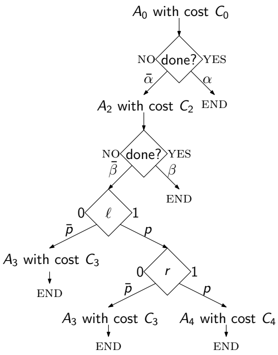
Modified Algorithm B



Original Algorithm A



Modified Algorithm D



$$C_A := \mathbb{E}[C(A,I)] = \mathbb{E}[C_0 + \bar{\alpha} \cdot (1 + \bar{p}C_1 + p \cdot (C_2 + \bar{\beta}(1 + \bar{p}C_3 + pC_4)))]$$
$$C_B := \mathbb{E}[C(B,I)] = \mathbb{E}[C_0 + \bar{\alpha} \cdot (1 + \bar{p}C_1 + p \cdot (1 + \bar{p}C_1 + p(C_2 + \bar{\beta}C_4)))]$$
$$C_D := \mathbb{E}[C(D,I)] = \mathbb{E}[C_0 + \bar{\alpha} \cdot (C_2 + \bar{\beta}(1 + \bar{p}C_3 + p(1 + \bar{p}C_3 + pC_4)))]$$

$$(C_B - C_A) + p \cdot (C_D - C_A) = \ldots = 0$$
$$\Rightarrow C_B - C_A \le 0 \vee C_D - C_A \le 0$$

$\Rightarrow$ B or D (or both) are at least as good as A and both visit superleaf (ℓ, r) as desired.

Theorem (Tarsi 1984)

For any $p \in [0, 1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in \textbf{Algos}$ be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in \textbf{Algos}$ that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

The expected cost for evaluating a superleaf is $1 + p$.
Hence

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] = (1 + p)\mathbb{E}[L']$$
$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)] = (1 + p)\mathbb{E}[L]$$

Finally we obtain:

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(\text{simpleEval}, I)] = (1 + p)\mathbb{E}[L] \leq (1 + p)\mathbb{E}[L']$$
$$= \mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Hence, simpleEval is optimal for $\mathcal{I}_p$. $\square$

Content

Ski Rental – A Prototypical Online Problem

Karlsruhe Institute of Technology

Setting: You are on a ski trip

Trip lasts for unknown number of days $I \in \mathbb{N}$
("as long as there is snow").
Every day, if no skis bought yet:

- RENT skis for one day for cost 1 *or*
- BUY skis for cost $B \in \mathbb{N}$.

Goal: Minimise Competitive Ratio

The *competitive ratio* of distribution $\mathcal{A}$ on **Algos** is

$$C_{\mathcal{A}} = \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)]}{\mathrm{OPT}(I)}.$$

Framing using Online Algorithms

- **Inputs** $= \mathbb{N}$: number of days (not known in advance)
- **Algos** $= \mathbb{N}$: specify day for choosing BUY
- cost for $A \in$ **Algos** on $I \in$ **Inputs**:

$$C(A, I) = \begin{cases} I & \text{if } I < A \\ A - 1 + B & \text{otherwise.} \end{cases}$$

- cost of optimum *offline* solution

$$\mathrm{OPT}(I) = \begin{cases} I & \text{if } I < B \\ B & \text{otherwise.} \end{cases}$$

Break-Even is the best deterministic algorithm

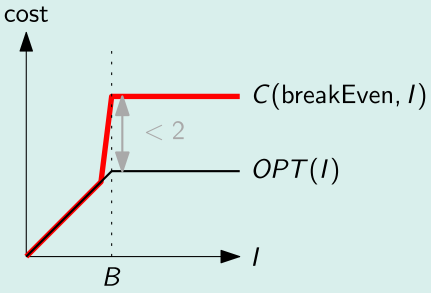
Observation

The algorithm breakEven $:= B$ has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for breakEven is attained for input $I = B$.

$$C_{\text{breakEven}} = \sup_{I \in \mathbb{N}} \frac{C(\text{breakEven}, I)}{\mathrm{OPT}(I)} = \frac{C(\text{breakEven}, B)}{\mathrm{OPT}(B)} = \frac{B - 1 + B}{B} = \frac{2B-1}{B}.$$

Break-Even is the best deterministic algorithm

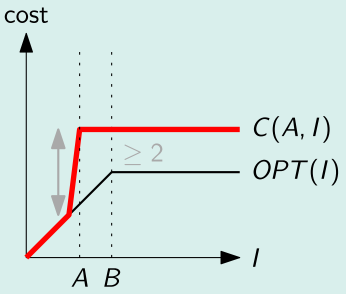
Observation

The algorithm breakEven $:= B$ has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for $A \in$ **Algos** with $A < B$ is attained for input $I = A$.

$$C_A = \sup_{I \in \mathbb{N}} \frac{C(A, I)}{\mathrm{OPT}(I)} = \frac{C(A, A)}{\mathrm{OPT}(A)} = \frac{A - 1 + B}{A} = 1 + \frac{B - 1}{A} \geq 1 + 1 = 2.$$

Break-Even is the best deterministic algorithm

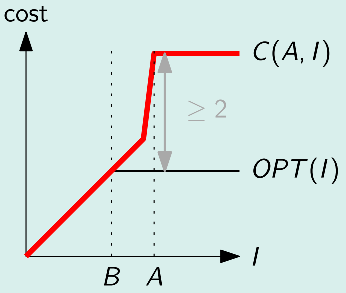
Observation

The algorithm breakEven $:= B$ has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for $A \in$ **Algos** with $A > B$ is attained for input $I = A$.

$$C_A = \sup_{I \in \mathbb{N}} \frac{C(A, I)}{\mathrm{OPT}(I)} = \frac{C(A, A)}{\mathrm{OPT}(A)} = \frac{A - 1 + B}{B} = 1 + \frac{A-1}{B} \geq 1 + 1 = 2.$$

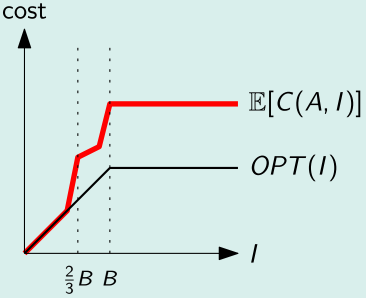
A randomised algorithm can beat break-even

Karlsruhe Institute of Technology

Observation (assuming wlog that B is a multiple of 3)

The randomised algorithm $\mathcal{A} = \mathcal{U}(\{\frac{2}{3}B, B\})$ has competitive ratio $\approx 1 + \frac{5}{6}$.

Proof



The competitive ratio of $\mathcal{A}$ "spikes" for inputs $\frac{2}{3}B$ and B. It is decreasing in between and constant after B.

$$\mathbb{E}_{A\sim\mathcal{A}}[C(A, \tfrac{2}{3}B)] = \underbrace{\tfrac{2}{3}B - 1}_{\text{rent}} + \underbrace{\tfrac{1}{2}(1 + B)}_{\text{rent or buy}} < \tfrac{7}{6}B, \qquad \mathrm{OPT}(\tfrac{2}{3}B) = \tfrac{2}{3}B,$$

$$\mathbb{E}_{A\sim\mathcal{A}}[C(A, B)] = \underbrace{B}_{\text{buy}} + \underbrace{\tfrac{2}{3}B - 1}_{\text{rent}} + \underbrace{\tfrac{1}{2}(\tfrac{1}{3}B)}_{\text{maybe rent}} < \tfrac{11}{6}B, \qquad \mathrm{OPT}(B) = B.$$

Hence $C_{\mathcal{A}} = \sup_{I\in\mathbb{N}} \frac{\mathbb{E}_{A\sim\mathcal{A}}[C(A, I)]}{\mathrm{OPT}(I)} \le \max\left\{\frac{7/6}{2/3}, \frac{11/6}{1}\right\} = \max\left\{\frac{7}{4}, \frac{11}{6}\right\} = \frac{11}{6}$.

What's next?

Goal: Lower bound

No randomised algorithm has competitive ratio better than ≈ 1.582.

Yao's Principle for Online Algorithms

Karlsruhe Institute of Technology

Theorem (see Online Optimization Lecture, Corollary 3.8, Prof. Yann Disser, Darmstadt, 2023)

For any distribution $\mathcal{A}_0$ on **Algos** and any distribution $\mathcal{I}_0$ on **Inputs** we have

$$C_{\mathcal{A}_0} \stackrel{\text{def}}{=} \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)]}{\mathrm{OPT}(I)} \geq \frac{\inf_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)]}{\mathbb{E}_{I \sim \mathcal{I}_0}[\mathrm{OPT}(I)]}.$$

Remark

- Yao's principle exists for other settings as well.
- Proof of "$\geq$" relatively easy to prove.
- Tightness typically follows from duality of optimisation problems or fixed point theorems.
  (though I'm not sure how it works here)

A hard distribution for Ski-Rental: Intuition

Karlsruhe Institute of Technology

$$\mathcal{I}_0 := Geo(\tfrac{1}{B}).$$

Why $\mathcal{I}_0$?

- distribution is **memoryless**.
  Assume no skis bought on day i: Minimising expected *future* cost is the same problem as on day 1.
  $\hookrightarrow$ wlog: either buy right away or not at all.
- **expectation** tuned such that

$$\mathbb{E}_{I\sim\mathcal{I}_0}[C(\text{never buy}, I)] = \mathbb{E}_{I\sim\mathcal{I}_0}[C(\text{immediately buy}, I)] = B.$$

$\hookrightarrow$ all strategies equally good

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := Geo(\frac{1}{B})$ and $q := 1 - \frac{1}{B} =$ "Pr[❄]". Then

- i $\mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)] = B$ *for all* $A \in \mathbb{N}$.
- ii $\mathbb{E}_{I \sim \mathcal{I}_0}[\mathrm{OPT}(I)] = B(1 - (1 - \frac{1}{B})^B)$.

Seen before:

Any random variable X with values in $\mathbb{N}$ satisfies
$$\mathbb{E}[X] = \sum_{j \geq 1} \Pr[X \geq j].$$

Proof of (i)

$$\mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_0}\left[\sum_{i \in \mathbb{N}} \text{cost on day } i\right] = \sum_{i \in \mathbb{N}} \mathbb{E}_{I \sim \mathcal{I}_0}[\text{cost on day } i] = \sum_{i=1}^{A-1} \underbrace{\Pr[I \geq i] \cdot 1}_{\text{rent}} + \underbrace{\Pr[I \geq A] \cdot B}_{\text{buy}}$$

$$= \sum_{i=1}^{A-1} q^{i-1} + q^{A-1} \cdot B = \sum_{i=0}^{A-2} q^i + q^{A-1} \cdot B = \frac{1 - q^{A-1}}{1 - q} + q^{A-1} \cdot B$$

$$= (1 - q^{A-1})B + q^{A-1} \cdot B = B.$$

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := Geo(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "Pr[❄]". *Then*

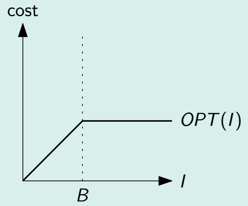
- i $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)] = B$ *for all* $A \in \mathbb{N}$.
- ii $\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)] = B(1-(1-\frac{1}{B})^B)$.

Seen before:

Any random variable X with values in $\mathbb{N}$ satisfies
$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Proof of (ii)

$$\begin{aligned}
\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)] &= \sum_{j\geq 1} \Pr[\mathrm{OPT}(I) \geq j] = \sum_{j=1}^{B} \Pr[\mathrm{OPT}(I) \geq j] \\
&= \sum_{j=1}^{B} \Pr[I \geq j] = \sum_{j=1}^{B} q^{j-1} = \sum_{j=0}^{B-1} q^j \\
&= \frac{1-q^B}{1-q} = B(1-(1-\tfrac{1}{B})^B).
\end{aligned}$$



Note: $\mathrm{OPT}(I) = I$ for $I \in [B]$.

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \mathit{Geo}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "Pr[❄]". *Then*

- i $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)] = B$ *for all* $A \in \mathbb{N}$.
- ii $\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)] = B(1-(1-\frac{1}{B})^B)$.

Seen before:

Any random variable X with values in $\mathbb{N}$ satisfies

$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Lower bound for Ski-Rental

By Yao's theorem any randomised algorithm $\mathcal{A}$ for ski-rental has competitive ratio at least

$$c_{\mathcal{A}} \geq \frac{\inf_{A\in\textbf{Algos}} \mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)]} = \frac{B}{B(1-(1-\frac{1}{B})^B)} = \frac{1}{1-(1-\frac{1}{B})^B}.$$

For large B the lower bound converges to $\lim_{B\to\infty} \frac{1}{1-(1-\frac{1}{B})^B} = \frac{1}{1-1/e} = \frac{e}{e-1} \approx 1.582$.

Upper bound for Ski-Rental

Karlsruhe Institute of Technology

Remark: The lower bound is tight (Karlin et al. 1994)

There exists a distribution $\mathcal{A}$ on $[B]$ such that $c_{\mathcal{A}} \leq \frac{e}{e-1}$.

Applications

Very basic online question:

Should I pay a small possibly recurring cost or a large one time cost?

Occurs in:

- Cache management.
- Networking.
- Scheduling.
- ...

Conclusion

Karlsruhe Institute of Technology

Algorithm Design as a Two-Player Game

- “we” choose algorithm to minimise cost
- “adversary” chooses input to maximise cost
- Nash/Loomis: It does not matter who moves first *if* mixed strategy is allowed for first player.

Yao’s Principle

Lower bound on worst-case expected cost of *any randomised algorithm* $\mathcal{A}_0$ by analying *any deterministic algorithm* on specific input distribution $\mathcal{I}_0$.

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \geq \mathcal{C} \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Can narrow down randomised complexity $\mathcal{C}$ of underlying problem from both sides.

Anhang: Mögliche Prüfungsfragen I

Spieltheorie:

- Was ist ein Zwei-Spieler-Spiel im Sinne der Spieltheorie?
- Was ist ein Nash-Equilibrium?
- Gibt es immer ein Nash-Equilibrium?
- Was ist ein Nullsummenspiel?
- Was besagt der Satz von Nash (für Zwei-Spieler Nullsummenspiele)?
- Was besagt der Satz von Loomis?
- Beweise den Satz von Loomis! (anspruchsvolle Aufgabe)

Yaos Prinzip:

- Worin besteht die Verbindung zwischen Spieltheorie und dem Entwurf von Algorithmen?
- Wie ist die randomisierte Komplexität (bzgl. einer Kostenfunktion C) normalerweise definiert? Welche andere Sichtweise ergibt sich darauf durch den Satz von Loomis?
- Formuliere Yaos Prinzip! Wofür ist es nützlich?

Anhang: Mögliche Prüfungsfragen II

Karlsruhe Institute of Technology

Anwendung auf $\bar{\wedge}$-Bäume:

- Welches Ziel haben wir uns bei der Auswertung von $\bar{\wedge}$-Bäumen gesetzt? (Anfragekomplexität minimieren)
- Welche Worst-Case Kosten lassen sich mit einem deterministischen Algorithmus erreichen?
- Können randomisierte Algorithmen das besser? Wie?
- Man kann sich recht leicht überlegen, dass die randomisierte Komplexität $\Omega(\sqrt{n})$ beträgt. Wie ging das?
- Wir haben auch eine schärfere Analyse gesehen. Welche Komponenten hatte diese? Insbesondere: Wie kommt dabei Yao Prinzip zur Anwendung?
- Was besagt der Satz von Tarsi?

Ski-Rental-Problem:

- Formuliere das Ski-Rental Problem.
- Wie nennt man diese Art von Problem? (*Online* Problem)
- Spielt das nur im Wintersport eine Rolle? (nur Stichworte)
- Wie ist der kompetitive Faktor definiert?

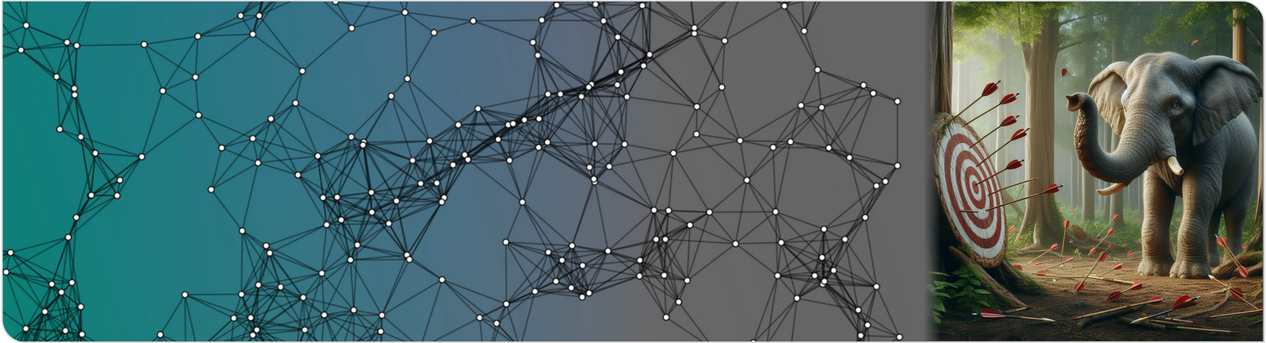
Anhang: Mögliche Prüfungsfragen III

- Was ist der beste deterministische Algorithmus? Wie kann man das einsehen?
- Gibt es einen randomisierten Algorithmus der Break-Even schlagen kann? (nur die Idee)
- Formuliere Yaos Prinzip für Online Algorithmen.
- Welche Eingabeverteilung haben wir für die untere Schranke für Ski-Rental zugrunde gelegt? Was ist die Intuition?
- Welche Kosten ergeben sich für Online und Offline Algorithmen für diese Eingabeverteilung? Was lässt sich entsprechend über den kompetitiven Faktor sagen?

Karlsruhe Institute of Technology

Probability and Computing – Approximation Algorithms

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Lecture Notes by Worsch

This lecture's content is covered in Thomas Worsch's notes from 2019.

Content

1. What is Randomised Approximation?

2. Approximately counting satisfying assignments for Boolean formulas

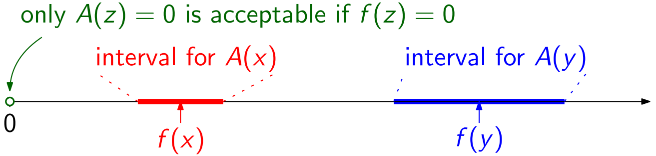
Randomised Approximate Counting

Definition

A randomised algorithm A *approximates* a quantity $f(x)$ if for any input x the output $A(x)$ satisfies:

$$\Pr[|A(x) - f(x)| \leq \varepsilon \cdot f(x)] \geq 1 - \delta.$$

The parameters are the *relative error* ε and the *failure probability* δ.



$(\varepsilon = \frac{1}{4})$

Remark: Related Complexity Classes

PRAS. Problems admitting A with running time polynomial in $|x|$, but not necessarily in $\frac{1}{\varepsilon}$ (for $\delta = 1/4$).

FPRAS. Problems admitting A with running time polynomial in $|x|$ and $\frac{1}{\varepsilon}$ (for $\delta = 1/4$).

Note: Also defined where $f(x)$ is not a *number*. For instance: Want to compute a *vertex cover* with a size close to optimal.

Counting Satisfiable Assignments of Boolean Formulas

Karlsruhe Institute of Technology

A counting problem

For Boolean formula $B(x_1, \dots, x_n)$ let $\#B$ be the number of satisfying assignments:

$$\#B = |\{(x_1, \dots, x_n) \in \{0,1\}^n \mid B(x_1, \dots, x_n) = 1\}|.$$

Example

$$B = (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_3)$$
$$\#B = |\{(0,0,0), (0,0,1), (1,0,1), (1,1,1)\}| = 4$$

Approximation algorithm for $\#B$ in general? Unlikely.

Assume A satisfies $\Pr[|A(B) - \#B| \leq \varepsilon(\#B)] \leq 1 - \delta$ for $\varepsilon = \frac{1}{2}$ and $\delta = \frac{1}{4}$. Then

$$B \text{ is UNSAT} \Leftrightarrow \#B = 0 \Rightarrow \Pr[|A(B) - 0| \leq \tfrac{1}{2} \cdot 0] \geq \tfrac{3}{4} \Rightarrow \Pr[A(B) = 0] \geq \tfrac{3}{4}$$
$$B \text{ is SAT} \Leftrightarrow \#B > 0 \Rightarrow \Pr[|A(B) - \#B| \leq \tfrac{1}{2} \cdot \#B] \geq \tfrac{3}{4} \Rightarrow \Pr[A(B) > 0] \geq \tfrac{3}{4}$$

If A is polynomial time then A is BPP algorithm for SAT.
Then SAT $\in$ BPP and NP $\subseteq$ BPP. Hard to believe...

What could be a tractable special case?

Relative error $\varepsilon < 1$ requires distinguishing:

$$\text{UNSAT} \Leftrightarrow \#B = 0 \qquad \text{from} \qquad \text{SAT} \Leftrightarrow \#B \geq 1.$$

CNF is hopeless

$$B = (x_1 \vee \bar{x}_2 \vee \bar{x}_{42}) \wedge \ldots \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_{37})$$

deciding SAT is NP-hard for clause size 3.

An asymmetry for CNF formulas

- B is called TAUTology if $\#B = 2^n$.
- "is B TAUT?" is easy to decide:
  Only empty CNF-formula is TAUT.
  (assuming x_i and $\bar{x}_i$ never in the same clause)
- Try approximating *unsatisfying* assignments?

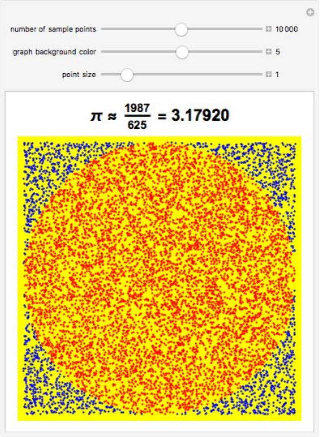
$$f(x) := 2^n - \#B.$$

Consider DNF!

$$B' = \bar{B} = (\bar{x}_1 \wedge x_2 \wedge x_{42}) \vee \ldots \vee (x_1 \wedge \bar{x}_3 \wedge x_{37})$$

- $\#B' = 2^n - \#B$.
- "B' is SAT" is easy to decide
  (only empty DNF-formula is UNSAT.)

Intuition: Approximating π



https://demonstrations.wolfram.com/ApproximatingPiByTheMonteCarloMethod/

Requirements for estimating area of disk (and hence π):

- Know formula for area of square
- Sample uniformly from square
- decide for $x, y \in [-1, 1]$ if (x, y) in disk: $x^2 + y^2 \leq 1$

Approximate $|S|$ for $S \subseteq D$ by naive sampling

Karlsruhe Institute of Technology

Algorithm approxSetSize($\mathbb{1}_{\cdot\in S}$, D)**:**

```
hits ← 0
for i = 1 to N do
    sample x ~ U(D)
    hits ← hits + 1_{x∈S}
return hits/N · |D|
```

Chernoff (→ Concentration, slide 15)

For $\varepsilon \in (0,1)$ and $X \sim Bin(N,p)$:

$$\Pr[|X - \mathbb{E}[X]| > \varepsilon\mathbb{E}[X]] < 2\exp(-\varepsilon^2\mathbb{E}[X]/3).$$

Simple Theorem

Let D be a finite set and $S \subseteq D$ such that we can efficiently

- compute $|D|$
- sample uniformly from D
- decide for given $x \in D$ whether $x \in S$

Let $p = |S|/|D|$. Then approxSetSize with $N = \frac{3\log(2/\delta)}{\varepsilon^2 p}$ approximates $|S|$ with relative error ε and failure probability δ.

$\hookrightarrow$ Special Case $\varepsilon, \delta = \Theta(1)$: Need $N = \Omega(1/p)$ samples.

Proof: Apply Chernoff to hits $\sim Bin(N,p)$.

$$\Pr[\text{fail}] = \Pr[\big|\text{result} - |S|\big| > \varepsilon|S|] = \Pr[\big|\tfrac{\text{hits}}{N}\cdot|D| - |S|\big| > \varepsilon|S|] = \Pr[\big|\text{hits} - \tfrac{|S|}{|D|}N\big| > \varepsilon\tfrac{|S|}{|D|}N]$$

$$= \Pr[\big|\text{hits} - pN\big| > \varepsilon pN] = \Pr[\big|\text{hits} - \mathbb{E}[\text{hits}]\big| > \varepsilon\mathbb{E}[\text{hits}]] \le 2\exp(-\varepsilon^2\mathbb{E}[\text{hits}]/3) = 2\exp(-\varepsilon^2 pN/3) = \delta.$$

Approximate $|S|$ for $S \subseteq D$ by naive sampling

Algorithm approxSetSize($\mathbb{1}_{\cdot \in S}$, D)**:**

hits $\leftarrow 0$
for $i = 1$ *to* N **do**
  sample $x \sim \mathcal{U}(D)$
  hits $\leftarrow$ hits $+ \mathbb{1}_{x \in S}$
return $\frac{\text{hits}}{N} \cdot |D|$

Chernoff ($\rightarrow$ Concentration, slide 15)

For $\varepsilon \in (0,1)$ and $X \sim Bin(N,p)$:

$$\Pr[|X - \mathbb{E}[X]| > \varepsilon \mathbb{E}[X]] < 2\exp(-\varepsilon^2 \mathbb{E}[X]/3).$$

Simple Theorem

Let D be a finite set and $S \subseteq D$ such that we can efficiently

- compute $|D|$
- sample uniformly from D
- decide for given $x \in D$ whether $x \in S$

Let $p = |S|/|D|$. Then approxSetSize with $N = \frac{3\log(2/\delta)}{\varepsilon^2 p}$ approximates $|S|$ with relative error ε and failure probability δ.

$\hookrightarrow$ Special Case $\varepsilon, \delta = \Theta(1)$: Need $N = \Omega(1/p)$ samples.

Application to $\#B$

- $S =$ satisfying assignments of B
- $D = \{0,1\}^n$
- $p = \frac{|S|}{|D|} = \frac{\#B}{2^n}$
- We may have $p = 1/2^n$
- $N = \Omega(2^n)$ required
- :-(

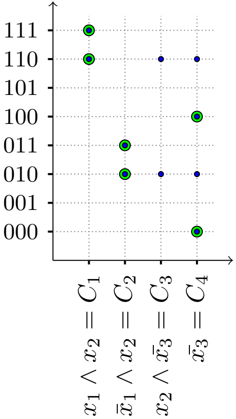
No Surprise

Of course this didn't work

Did not exploit that B is in DNF.

Approximating $\#B$ for B in DNF

Assume $B = C_1 \vee \ldots \vee C_m$
where C_i contains ℓ_i literals.



- $D_i := \{x \in \{0,1\}^n \mid C_i(x) = 1\}$ (satisfying assignments of C_i)
- $D := \{(i, x) \mid i \in [m], x \in D_i\}$ $(= D_1 \dot\cup \ldots \dot\cup D_m)$
- $S := \{(i, x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots D_{i-1}\}$

Observations

- $|S| = \#B$
- $|D_i| = 2^{n-\ell_i}$ and we can efficiently sample from $\mathcal{U}(D_i)$:
  $\hookrightarrow$ set variables appearing in C_i as required, others from $Ber(1/2)$.
- We can efficiently compute $|D| = \sum_{i=1}^{m} |D_i|$ and sample $(I, X) \sim \mathcal{U}(D)$:
  - First sample I such that $\Pr[I = i] = \frac{|D_i|}{|D|}$.
  - Then sample $X \sim \mathcal{U}(D_i)$.
  - Yields $\Pr[(I, X) = (i, x)] = \frac{|D_i|}{|D|} \cdot \frac{1}{|D_i|} = \frac{1}{|D|}$ for all $(i, x) \in D$.
- We can efficiently decide "is $(i, x) \in S$?" (in time $\mathcal{O}(mn)$)
- $p = \frac{|S|}{|D|}$ satisfies $p \geq \frac{1}{m}$.

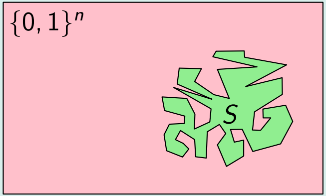
Takeaway

Theorem

If B is in DNF, then we can approximate $\#B$ in polynomial time (using $N = m \cdot \frac{3\log(2/\delta)}{\varepsilon^2}$ samples) with relative error ε and failure probability δ.

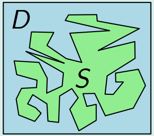
Intuition: Why did this work?

Naive strategy:



Problem: $|S|/|\{0,1\}^n|$ may be exponentially small

Improved strategy:



Advantage: $|S|/|D|$ is $\Omega(1/m)$.

Conclusion

Karlsruhe Institute of Technology

Randomised Approximation is Powerful

For B in DNF:

- Computing $\#B$ *exactly* is $\#\boldsymbol{P}$-complete.
- no *deterministic approximation* algorithm for such problems is known
- we analysed an efficient *randomised approximation* algorithm

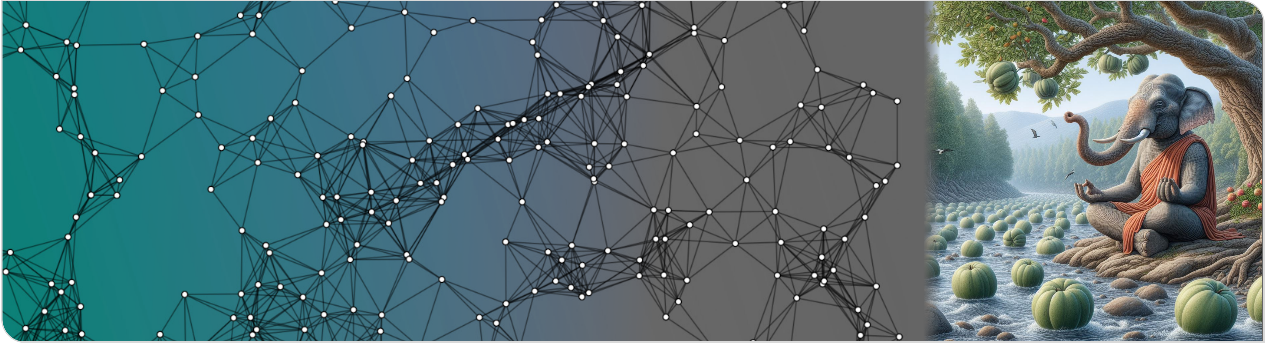
Anhang: Mögliche Prüfungsfragen

- Was ist ein randomisierter Approximationsalgorithmus (für ein Zählproblem)?
- Wir haben das Zählproblem $\#B$ für Boolesche Formeln betrachtet. Hatten wir im allgemeinen Fall Erfolg? Warum nicht?
- Welchen Spezialfall haben wir uns vorgenommen? Wieso tritt dort nicht das selbe Problem auf wie im allgemeinen Fall?
- Wir haben einen Algorithmus gesehen der für zwei Mengen $S \subseteq D$ die Größe von $|S|$ schätzt.
  - Unter welchen Annahmen ist dieser anwendbar?
  - Wie hat der Algorithmus funktioniert?
  - Wie hängt die Anzahl der nötigen samples von $|S|$ und $|D|$ ab?
- Um $\#B$ für DNF Formel B zu schätzen haben wir einen schlaueren Ansatz kennengerlernt.
  - Wie hat dieser funktioniert?
  - Wie vermeidet dieser das Problem das naiven Ansatzes?

Karlsruhe Institute of Technology

Probability and Computing – Streaming

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Content

1. **Definition: What is a Streaming Algorithm?**

2. **Morris' Algorithm for $F_1 = m$**

3. **The CVM Algorithm for $F_0 = |\{a_1 \ldots, a_m\}|$**

What is a Streaming Algorithm?

- looong input data stream $(a_1, \ldots, a_m) \in [n]^m$ can only be read *once* from left to right
- goal: approximate some value $F = F(a_1, \ldots, a_m)$ with small relative error ε and failure probability δ.
  $\hookrightarrow$ streaming algorithms are approximation algorithms
- challenge: use less *space* than exact algorithm (in particular: cannot store $(a_1, \ldots, a_m)$).
  $\hookrightarrow$ don't care about running time

Formally, a streaming algorithm is given by three algorithms init, update and result used as follows:

$Z \leftarrow \text{init}()$
for $i = 1$ **to** m **do**
  $Z \leftarrow \text{update}(Z, a_i)$
return result(Z)

Its space complexity is the space required for Z.

Today's Motivating Examples

A Router approximately counts traffic over each connection.
$\hookrightarrow$ maybe: detect anomalies related to DDoS

B Website approximately counts number of unique users visiting a resource.

Today's Formal Results

A Approximate $F_1(a_1, \ldots, a_m) = m$ in expected space $\frac{1}{\varepsilon^2\delta} \log\log m$.

B Approximate $F_0(a_1, \ldots, a_m) = |\{a_1, \ldots, a_m\}|$ in expected space $\frac{1}{\varepsilon^2} \log(n) \cdot \log(m/\delta)$.

Content

Attempt I: Naive Counting

Naive Counting

Algorithm init**:**
 $Z \leftarrow 0$
 return Z

Algorithm update(Z, a)**:**
 $Z \leftarrow Z + 1$
 return Z

Algorithm result(Z)**:**
 return Z

Observations on Naive counting

- No errors ($\varepsilon = \delta = 0$).
- Requires $\lceil \log(m + 1) \rceil$ bits of memory.
- No *deterministic* algorithm can use less space
  - Would have to "reuse" a state Z.
  - Is then trapped in an infinite loop.
  - Result arbitrarily far off if m large enough.

Attempt II: Lossy Counting

Approximate Counting

- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

Karlsruhe Institute of Technology

Lossy Counting, parameter p

Algorithm init**:**
  $Z \leftarrow 0$
  return Z

Algorithm update(Z, a)**:**
  with probability p **do**
    $Z \leftarrow Z + 1$
  return Z

Algorithm result(Z)**:**
  return Z/p

Analysis (Exercise)

For any $p \in (0, 1]$ we have

- $\mathbb{E}[\text{result}] = m$
- $\Pr[|\text{result} - m| \leq \varepsilon m] \geq 1 - 2\exp(-\varepsilon^2 pm/3)$.
- $\mathbb{E}[\text{space}] \leq \log_2(1 + mp) + 1$.

Corollary

By choosing $p = \frac{3}{\varepsilon^2 m}\log(2/\delta)$ we get

$$\Pr[\text{fail}] \leq \delta \text{ and } \mathbb{E}[\text{space}] \leq \mathcal{O}(\log(\tfrac{1}{\varepsilon}) + \log\log(1/\delta))).$$

Serious Objection

Correctly choosing p requires already knowing m.

(or at least the order of magnitude of m)

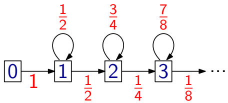
Attempt III: Morris' Algorithm

Morris' Algorithm

Algorithm init**:**
- $Z \leftarrow 0$
- **return** Z

Algorithm update(Z, a)**:**
- **with probability** 2^{-Z} **do**
  - $Z \leftarrow Z + 1$
- **return** Z

Algorithm result(Z)**:**
- **return** $2^Z - 1$



states of Z
transition probabilities

Lemma: Morris' Algorithm is an *Unbiased Estimator*

$\mathbb{E}[\text{result}] = m$.

Proof by Induction.

Claim: The state Z_i after i updates satisfies $\mathbb{E}[2^{Z_i}] = i + 1$. True for $i \in \{0, 1\}$.

$$\mathbb{E}[2^{Z_{i+1}}] \overset{\text{LTE}}{=} \sum_{j \geq 0} \Pr[Z_i = j] \cdot \mathbb{E}[2^{Z_{i+1}} \mid Z_i = j]$$

$$= \sum_{j \geq 0} \Pr[Z_i = j] \cdot (2^{j+1} \Pr[Z_{i+1} = j + 1 \mid Z_i = j] + 2^j \Pr[Z_{i+1} = j \mid Z_i = j])$$

$$= \sum_{j \geq 0} \Pr[Z_i = j] \cdot (2^{j+1} 2^{-j} + 2^j (1 - 2^{-j})) = \sum_{j \geq 0} \Pr[Z_i = j] \cdot (2 + 2^j - 1)$$

$$= \sum_{j \geq 0} \Pr[Z_i = j] + \sum_{j \geq 0} \Pr[Z_i = j] 2^j = 1 + \mathbb{E}[2^{Z_i}] \overset{\text{Ind.}}{=} 1 + (i + 1) = i + 2. \quad \square$$

Attempt III: Morris' Algorithm

Approximate Counting

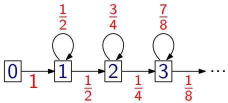
- stream $(a_1, \dots, a_m)$
- want $F_1 = m$

Morris' Algorithm

Algorithm init**:**
- $Z \leftarrow 0$
- **return** Z

Algorithm update(Z, a)**:**
- **with probability** 2^{-Z} **do**
  - $Z \leftarrow Z + 1$
- **return** Z

Algorithm result(Z)**:**
- **return** $2^Z - 1$



states of Z
transition probabilities

Lemma 1: Worryingly large Variance

$Var(2^{Z_i}) = \frac{i^2 - i}{2} = \Theta(i^2)$.

Lemma 2

$\mathbb{E}[2^{2Z_i}] = \frac{3i(i+1)}{2} + 1$.

Proof of Lemma 1 using Lemma 2.

$$Var(2^{Z_i}) = \mathbb{E}[2^{2Z_i}] - \mathbb{E}[2^{Z_i}]^2 \overset{\text{Lem. 2}}{=} \frac{3i(i+1)}{2} + 1 - (i+1)^2$$
$$= \frac{3i(i+1) + 2 - 2i^2 - 4i - 2}{2} = \frac{i^2 - i}{2}. \quad \square$$

Attempt III: Morris' Algorithm

Approximate Counting
- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

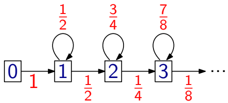
Karlsruhe Institute of Technology

Morris' Algorithm

Algorithm init**:**
- $Z \leftarrow 0$
- **return** Z

Algorithm update(Z, a)**:**
- **with probability** 2^{-Z} **do**
  - $Z \leftarrow Z + 1$
- **return** Z

Algorithm result(Z)**:**
- **return** $2^Z - 1$



states of Z

transition probabilities

Lemma 1: Worryingly large Variance

$Var(2^{Z_i}) = \frac{i^2 - i}{2} = \Theta(i^2)$.

Lemma 2

$\mathbb{E}[2^{2Z_i}] = \frac{3i(i+1)}{2} + 1$.

Proof of Lemma 2.

For $i \in \{0, 1\}$ ✓. Let now $i \geq 1$. Note $\Pr[Z_{i+1} = 0] = \Pr[Z_i = 0] = 0$.

$$\begin{aligned}
\mathbb{E}[2^{2Z_{i+1}}] &= \sum_{j \geq 1} 2^{2j} \Pr[Z_{i+1} = j] = \sum_{j \geq 1} 2^{2j} (\Pr[Z_i = j-1] \cdot 2^{-j+1} + \Pr[Z_i = j] \cdot (1 - 2^{-j})) \\
&= \sum_{j \geq 1} 2^{j+1} \Pr[Z_i = j-1] + \sum_{j \geq 1} 2^{2j} \Pr[Z_i = j] - \sum_{j \geq 1} 2^j \Pr[Z_i = j] \\
&= 4 \sum_{j \geq 0} 2^j \Pr[Z_i = j] + \sum_{j \geq 0} 2^{2j} \Pr[Z_i = j] - \sum_{j \geq 0} 2^j \Pr[Z_i = j] \\
&= 4\mathbb{E}[2^{Z_i}] + \mathbb{E}[2^{2Z_i}] - \mathbb{E}[2^{Z_i}] = 3\mathbb{E}[2^{Z_i}] + \mathbb{E}[2^{2Z_i}] = 3(i+1) + \mathbb{E}[2^{2Z_i}] \\
&\overset{\text{Ind.}}{=} 3(i+1) + \frac{3i(i+1)}{2} + 1 = \frac{3(i+2)(i+1)}{2} + 1. \quad \square
\end{aligned}$$

Space

Expected Space

$$\mathbb{E}[\text{space}] \leq \mathbb{E}[\lceil \log_2(1+Z_m)\rceil] \leq 1 + \mathbb{E}[\log_2(1+Z_m)] = 1 + \mathbb{E}[\log_2(1+\log_2(2^{Z_m}))]$$
$$\overset{(\star)}{\leq} 1 + \log_2(1+\log_2(\mathbb{E}[2^{Z_m}])) = 1 + \log_2(1+\log_2(m+1)) = \Theta(\log\log m).$$

$(\star)$ uses Jensen's inequality that you'll prove as an exercise.

Interim Conclusion: Morris is not good enough *yet*

- $\mathbb{E}[\text{result}] = m$ ✓ unbiased estimator
- $\mathbb{E}[\text{space}] = \mathcal{O}(\log\log m)$ ✓ highly space efficient
- $Var(\text{result}) = \Theta(m^2)$ ✗
  - Standarddeviation $\Theta(m)$
    $\rightsquigarrow$ right order of magnitude, but not better.

Morris$^+$: Use many copies of Morris' Algorithm

Karlsruhe Institute of Technology

Theorem

Consider a streaming algorithm that maintains a sequence $Z = (Z_1, \dots, Z_s)$ of independent Morris-counters and returns $\text{result}(Z) := \frac{\text{result}(Z_1)+\dots+\text{result}(Z_s)}{s}$. For $s = \frac{1}{\varepsilon^2\delta}$ we obtain

- $\mathbb{E}[\text{result}(Z)] = m$ and $\mathbb{E}[\text{space}] = \mathcal{O}(\frac{1}{\varepsilon^2\delta}\log\log m)$
- $\Pr[|\text{result}(Z) - m| \le \varepsilon m] = 1 - \mathcal{O}(\delta)$.

Reminder / Exercise: Variance

If X, Y are independent random variables and $s > 0$ then

- $Var(sX) = s^2\, Var(X)$
- $Var(X+Y) = Var(X) + Var(Y)$

Proof of Concentration using Chebyshev (or only Markov)

$$Var(\text{result}(Z)) = Var(\tfrac{1}{s}\sum_{i=1}^{s}\text{result}(Z_i)) = \tfrac{1}{s^2}\, Var(\sum_{i=1}^{s}\text{result}(Z_i))$$

$$= \tfrac{1}{s^2}\sum_{i=1}^{s} Var(\text{result}(Z_i)) = \tfrac{s}{s^2}\, Var(\text{result}(Z_1)) = \tfrac{1}{s}\Theta(m^2) = \Theta(m^2/s).$$

$$\Pr[\text{fail}] = \Pr[|\text{result}(Z) - m| > \varepsilon m] = \Pr[|\text{result}(Z) - \mathbb{E}[\text{result}(Z)]| > \varepsilon m]$$

$$= \Pr[(\text{result}(Z) - \mathbb{E}[\text{result}(Z)])^2 > \varepsilon^2 m^2] \overset{\text{Markov}}{\le} \frac{\mathbb{E}[(\text{result}(Z) - \mathbb{E}[\text{result}(Z)])^2]}{\varepsilon^2 m^2} = \frac{Var(\text{result}(Z))}{\varepsilon^2 m^2} = \Theta(1/(\varepsilon^2 s)) = \Theta(\delta). \quad \square$$

Markov:

$\Pr[X > c] \le \frac{\mathbb{E}[X]}{c}$.

Chebyschev:

$\Pr[X - \mathbb{E}[X] > c] \le \frac{Var(X)}{c^2}$.

Content

1. Definition: What is a Streaming Algorithm?

2. Morris' Algorithm for $F_1 = m$

3. **The CVM Algorithm for** $F_0 = |\{a_1 \dots, a_m\}|$

History

Counting Distinct Elements

- stream $(a_1, \ldots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \ldots, a_m\}|$

Karlsruhe Institute of Technology

Remark: CVM is not well-known

Popular line of algorithms for F_0 by Philippe Flajolet et al:

- ~~1984: Flajolet-Martin~~ (deprecated)
  $\hookrightarrow$ https://en.wikipedia.org/wiki/Flajolet-Martin_algorithm
- ~~2003: LogLog~~ (deprecated)
- 2007: HyperLogLog
  $\hookrightarrow$ `https://en.wikipedia.org/wiki/HyperLogLog`

The CVM-Algorithm

- 2022: European Symposium on *Simplicity* in Algorithms 2022
  $\hookrightarrow$ „Distinct Elements in Streams: An Algorithm for the (Text) Book“
- is a bit worse than HyperLogLog
- is easier to analyse than HyperLogLog

Next: We develop CVM in three steps.

Attempt I: Naively storing the set

Naive Storing

Algorithm init**:**
- $Z \leftarrow \varnothing$
- **return** Z

Algorithm update(Z, a)**:**
- $Z \leftarrow Z \cup \{a\}$
- **return** Z

Algorithm result(Z)**:**
- **return** $|Z|$

Observation

Naively storing the set requires $\Omega(F_0 \cdot \log n)$ bits.

Attempt II: Storing the set lossily

Counting Distinct Elements

- stream $(a_1, \dots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \dots, a_m\}|$

LossyStore, parameter p

Algorithm init**:**
  $Z \leftarrow \varnothing$
  return Z

Algorithm update(Z, a)**:**
  $Z \leftarrow Z \setminus \{a\}$
  with probability p **do**
    $Z \leftarrow Z \cup \{a\}$
  return Z

Algorithm result(Z)**:**
  return $|Z|$/p;

Chernoff for $X \sim Bin(n, p)$

$\Pr[|X - \mathbb{E}[X]| > \varepsilon\mathbb{E}[X]] \le 2\exp(-\varepsilon^2\mathbb{E}[X]/3)$.

Analysis

Let $Z_0, \dots, Z_m$ be the states of Z over time. Note: Each $a \in \{a_1, \dots, a_m\}$ is in Z_m independently with probability p, hence $|Z_m| \sim Bin(F_0, p)$.

- $\mathbb{E}[\text{result}] = \mathbb{E}[|Z_m|/p] = \mathbb{E}[|Z_m|]/p = F_0 p/p = F_0$.
  $\hookrightarrow$ result is *unbiased estimator* of F_0.
- $\Pr[\text{fail}] = \Pr[|\text{result} - F_0| > \varepsilon F_0] = \Pr[\big||Z_m|/p - F_0\big| > \varepsilon F_0]$
  $= \Pr[\big||Z_m| - pF_0\big| > \varepsilon p F_0] = \Pr[\big||Z_m| - \mathbb{E}[|Z_m|]\big| > \varepsilon\mathbb{E}[|Z_m|]]$
  $\overset{\text{Chern.}}{\le} 2\exp(-\varepsilon^2\mathbb{E}[|Z_m|]/3) = 2\exp(-\varepsilon^2 p F_0/3)$.
  $\hookrightarrow$ choose $p = p_\delta := \frac{3\log(2/\delta)}{\varepsilon^2 F_0}$ for $\Pr[\text{fail}] \le \delta$.
- Expected space *in the end* for $p = p_\delta$ (⚠ $\neq$ peak space consumption)
  $\mathbb{E}[|Z_m| \cdot \mathcal{O}(\log n)] = F_0 p_\delta \cdot \mathcal{O}(\log n) = \mathcal{O}(\frac{\log(1/\delta)}{\varepsilon^2}\log n)$.

Serious Objection: Need to know F_0 to choose p

- for $p \gg p_\delta$: space is wasted
- for $p \ll p_\delta$: failure becomes likely

Attempt III: Adjust lossiness dynamically

CVM, parameter T

Algorithm init:
- $Z \leftarrow \varnothing$
- $P \leftarrow 1$
- **return** (P, Z)

Algorithm update($(P, Z), a$):
- $Z \leftarrow Z \setminus \{a\}$
- **with probability** P **do**
  - $Z \leftarrow Z \cup \{a\}$
- **while** $|Z| \geq T$ **do** // shrink
  - $Z' \leftarrow \varnothing$
  - **for** $a \in Z$ **do**
    - **with probability** $1/2$ **do**
      - $Z' \leftarrow Z' \cup \{a\}$
  - $(Z, P) \leftarrow (Z', P/2)$
- **return** (P, Z)

Algorithm result((P, Z)):
- **return** $|Z|/P$

CVM behaves like LossyStore with dynamic p

Consider $A^{(p)} :=$ LossyStore(p) with states $Z_0^{(p)}, \dots, Z_m^{(p)}$ for $p \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\}$.

Let $(P_0, Z_0^{(\mathrm{CVM})}), \dots, (P_m, Z_m^{(\mathrm{CVM})})$ be the state of CVM.

$$
\begin{array}{lccccccccccc}
 & & a_1 & & a_2 & & a_3 & & a_4 & & a_m & \\
A^{(1)}: & Z_0^{(1)} & \longrightarrow & Z_1^{(1)} & \longrightarrow & Z_2^{(1)} & \longrightarrow & Z_3^{(1)} & \longrightarrow & \dots & \longrightarrow & Z_m^{(1)} \\
 & \supseteq & & \supseteq & & \supseteq & & \supseteq & & & & \supseteq \\
A^{(1/2)}: & Z_0^{(1/2)} & \textcircled{1}\longrightarrow & Z_1^{(1/2)} & \textcircled{1}\longrightarrow & Z_2^{(1/2)} & \textcircled{1}\longrightarrow & Z_3^{(1/2)} & \textcircled{1}\longrightarrow & \dots & \textcircled{1}\longrightarrow & Z_m^{(1/2)} \\
 & \supseteq & & \supseteq & & \supseteq & & \supseteq & & & & \supseteq \\
A^{(1/4)}: & Z_0^{(1/4)} & \textcircled{1}\textcircled{1}\longrightarrow & Z_1^{(1/4)} & \textcircled{1}\textcircled{1}\longrightarrow & Z_2^{(1/4)} & \textcircled{1}\textcircled{1}\longrightarrow & Z_3^{(1/4)} & \textcircled{1}\textcircled{1}\longrightarrow & \dots & \textcircled{1}\textcircled{1}\longrightarrow & Z_m^{(1/4)} \\
 & \supseteq & & \supseteq & & \supseteq & & \supseteq & & & & \supseteq \\
A^{(1/8)}: & Z_0^{(1/8)} & \textcircled{1}\textcircled{1}\textcircled{1}\longrightarrow & Z_1^{(1/8)} & \textcircled{1}\textcircled{1}\textcircled{1}\longrightarrow & Z_2^{(1/8)} & \textcircled{1}\textcircled{1}\textcircled{1}\longrightarrow & Z_3^{(1/8)} & \textcircled{1}\textcircled{1}\textcircled{1}\longrightarrow & \dots & \textcircled{1}\textcircled{1}\textcircled{1}\longrightarrow & Z_m^{(1/8)} \quad (Z_j^{\mathrm{CVM}})_{j\in[m]} \\
 & \supseteq & & \supseteq & & \supseteq & & \supseteq & & & & \supseteq \\
 & \vdots & & \vdots & & \vdots & & \vdots & & & &
\end{array}
$$

Intuition: The path of CVM:
- $(x, y) \leftarrow (0, 0)$ // top left
- **for** $i = 1$ **to** m **do** // m updates
  - $x \leftarrow x + 1$ // go right
  - **while** $|Z_x^{(2^{-y})}| \geq T$ **do**
    - $y \leftarrow y + 1$ // go down

final state is $Z_m^{(2^{-y})}$

Coupling between executions of $A^{(p)}$ and CVM:
- $A^{(p/2)}$ uses coin tosses of $A^{(p)}$ and one more.
  "$A^{(p/2)}$ keeps half of what $A^{(p)}$ keeps."
- CVM uses coin tosses of $A^{(P)}$ to process elements.
- When shrinking, CVM inspects past coin tosses done by $A^{(P/2)}$. (the next unused coin for all $a \in Z$)

Effects of the coupling:
- $Z_j^{(\mathrm{CVM})} = Z_j^{(P_j)}$ for $j \in [m]$
- $\mathrm{result}^{(\mathrm{CVM})} = \mathrm{result}^{(P_m)}$
- $\mathrm{fail}^{(\mathrm{CVM})} = \mathrm{fail}^{(P_m)}$

Attempt III: Adjust lossiness dynamically

Counting Distinct Elements

- stream $(a_1, \ldots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \ldots, a_m\}|$

Karlsruhe Institute of Technology

CVM, parameter T

Algorithm init:
  $Z \leftarrow \varnothing$
  $P \leftarrow 1$
  return (P, Z)

Algorithm update$((P, Z), a)$:
  $Z \leftarrow Z \setminus \{a\}$
  with probability P **do**
    $Z \leftarrow Z \cup \{a\}$
  while $|Z| \geq T$ **do** // shrink
    $Z' \leftarrow \varnothing$
    for $a \in Z$ **do**
      with probability $1/2$ **do**
        $Z' \leftarrow Z' \cup \{a\}$
    $(Z, P) \leftarrow (Z', P/2)$
  return (P, Z)

Algorithm result$((P, Z))$:
  return $|Z|/P$

Lemma: Failure Probability and Space

With $T = \frac{18 \log_2(2m/\delta)}{\varepsilon^2}$ we get $\Pr[\text{fail}^{\text{CVM}}] = \mathcal{O}(\delta)$ and $\text{space}^{\text{CVM}} = \mathcal{O}(\frac{\log(m/\delta)}{\varepsilon^2} \log n) + \lceil \log_2(\log_2(1/P_m)) \rceil$.

Analysis of CVM's failure probability (a bit sketchy)

- Recall: LossyStore$\left(p_\delta = \frac{3\log(2/\delta)}{\varepsilon^2 F_0}\right)$ has failure probability $\leq \delta$. Assume p_δ is power of 2.
- Then $\Pr[\text{fail}^{(p_\delta)}] \leq \delta$, $\Pr[\text{fail}^{(2p_\delta)}] \leq \delta^2$, $\Pr[\text{fail}^{(4p_\delta)}] \leq \delta^4, \ldots$.
- Therefore $\Pr[\text{fail}^{(1)}] + \ldots + \Pr[\text{fail}^{(2p_\delta)}] + \Pr[\text{fail}^{(p_\delta)}] \leq \ldots + \delta^8 + \delta^4 + \delta^2 + \delta = \mathcal{O}(\delta)$.
- $$\Pr[P_m < p_\delta] = \Pr[|Z_j^{(p_\delta)}| \geq T \text{ for some } j \in [m]] \leq m \cdot \Pr[|Z_m^{(p_\delta)}| \geq T]$$
  $$= m \cdot \Pr_{Z \sim Bin(F_0, p_\delta)}[Z \geq T] \overset{\triangle}{\leq} m \cdot 2^{-T} \leq m \cdot 2^{-\log(m/\delta)} = \delta.$$
  where Δ uses a Chernoff bound and $6\mathbb{E}[Z] = 6F_0 p_\delta = \dfrac{18\log_2(2/\delta)}{\varepsilon^2} \leq T$.
- $\text{fail}^{\text{CVM}} \Leftrightarrow \text{fail}^{(P_m)} \Rightarrow \left(P_m < p_\delta \vee \text{fail}^{(1)} \vee \text{fail}^{(2)} \vee \ldots \vee \text{fail}^{(p_\delta)}\right)$

Finally: $\Pr[\text{fail}^{\text{CVM}}] \leq \Pr[P_m < p_\delta \vee \text{fail}^{(1)} \vee \text{fail}^{(2)} \vee \ldots \vee \text{fail}^{(p_\delta)}] \overset{\text{UB}}{\leq} \delta + \mathcal{O}(\delta) = \mathcal{O}(\delta)$.

Conclusion

Streaming Algorithms

- Input read only once, from left to right.
- Goal: Use little space. (less than what is needed to store input stream)
- Motivation: Network actor wants to maintain statistic on traffic.

Morris' Algorithm for Counting the Stream Length

- approximation in space $\mathcal{O}(\frac{1}{\varepsilon^2\delta}\log\log m)$
  ($\varepsilon =$ relative error, $\delta =$ failure probability)
- deterministic algorithms need space $\lceil \log(1+m) \rceil$

CVM Algorithm for Counting *Distinct* Elements

- approximation in space $\frac{1}{\varepsilon^2}\log(n)\log(m/\delta)$

Anhang: Mögliche Prüfungsfragen I

Karlsruhe Institute of Technology

- Definition Streamingalgorithmen:
  - Was ist die Aufgabe eines Streamingalgorithmus (in Bezug auf eine Größe $F = F(a_1, \ldots, a_m)$)?
  - Was ist die spezifische Herausforderung für Streamingalgorithmen?
- Streamingalgorithmen für $F_1 = m$:
  - Was könnte ein Anwendungsfall sein, in dem man F_1 schätzen möchte?
  - Wie viel Speicher braucht man wenn man einfach nur zählt? Kann ein deterministischer Algorithmus etwas Schlaueres machen?
  - Wie funktioniert der LossyCounting Algorithmus? Warum hilft dieser uns nicht weiter?
  - Wie funktioniert Morris' Algorithmus?
  - Beweise, dass Morris' Algorithmus erwartungstreu ist.*
  - Beweise, dass der Speicherbedarf von Morris doppelt logarithmisch in m ist.
  - Welche Schwäche hatte Morris' Algorithmus noch und wie haben wir diese behoben?

Anhang: Mögliche Prüfungsfragen II

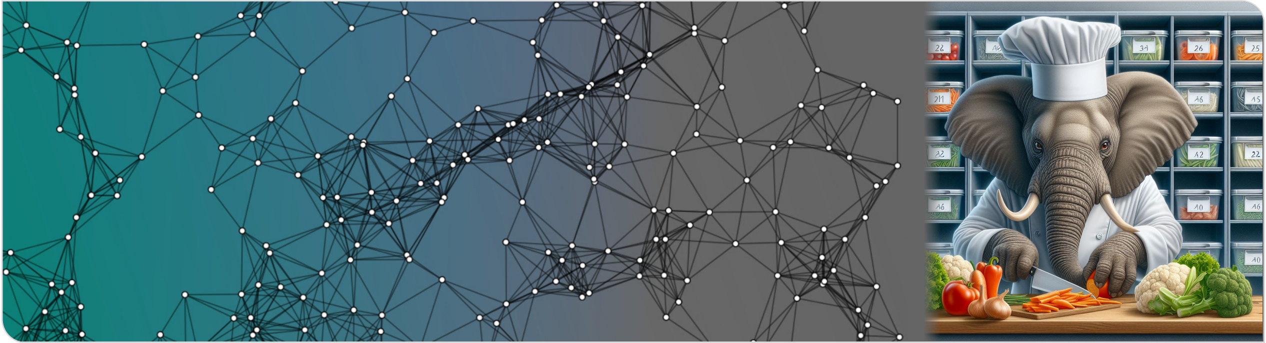
Karlsruhe Institute of Technology

- Streamingalgorithmen für $F_0 = \{a_1, \ldots, a_m\}$:
  - Was könnte ein Anwendungsfall sein, in dem man F_0 schätzen möchte?
  - Wie viel Speicher braucht der naive deterministische Algorithmus? Was können wir mit CVM erreichen?
  - Als Zwischenschritt haben wir den Algorithmus LossyStore formuliert. Wie funktioniert dieser?
  - Wie funktioniert der CVM Algorithmus? Wie steht dieser mit dem LossyStore Algorithmus in Verbindung?
  - In der Analyse der Fehlerwahrscheinlichkeit von CVM haben wir zwei Arten von Problemen unterschieden. Welche?*

Karlsruhe Institute of Technology

Probability and Computing – Classic Hash Tables

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Prüfungsanmeldung

- Am einfachsten: Hier angeben, wann ihr Zeit habt:
  `https://www.terminplaner.dfn.de/W4m8QyA9vvp1K19m`
- Alternativ: Email an Stefan und Max.
- Wir bieten euch dann einen Termin per Email an.

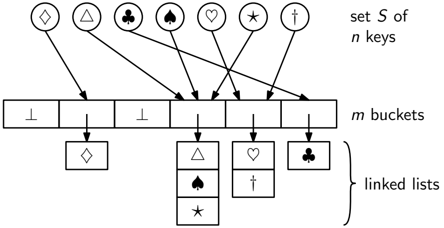
Content

Hash Table with Chaining

e.g. `std::unordered_set`, `java.util.HashMap`

Terminology

D:	*Universe* (or domain) of *keys* (strings, integers, game states in chess)
$S \subseteq D$:	set of n keys (possibly with associated data)
$h : D \to R$:	hash function, range usually $R = [m]$
$\alpha = \frac{n}{m}$:	*load factor*, $\alpha \leq \alpha_{\max} = \mathcal{O}(1)$



Goal

Operations in time t with $\mathbb{E}[t] = \mathcal{O}(1)$.
Randomness comes from the hash function.

Ideal Hash Functions

Every function from D to R is equally likely to be h.

Ideal Hash Functions are Impractical

Naive Idea

- Let R^D denote all functions from D to R. We pick $h \sim \mathcal{U}(R^D)$.
- There are $|R|$ options for the hash of each $x \in D$
- Hence: $|R^D| = |R|^{|D|}$

$x \in D$	x_1	x_2	x_3	$\dots$	$x_{\lvert D \rvert}$
$h(x) \in R$	?	?	?	$\dots$	?

Why $h \sim \mathcal{U}(R^D)$ is desirable

- $h \sim \mathcal{U}(R^D) \Leftrightarrow \forall x_1, \dots, x_n \in D : h(x_1), h(x_2), \dots, h(x_n)$ are *independent* and uniformly random in R.
  $\hookrightarrow$ independence is very useful in an analysis
- In particular: $\forall x_1, \dots, x_n \in D, \forall i_1, \dots, i_n : \Pr_{h \sim \mathcal{U}(R^D)}[h(x_1) = i_1 \wedge \dots \wedge h(x_n) = i_n] = |R|^{-n}$.

Why $h \sim \mathcal{U}(R^D)$ is unwieldy

$\log_2(|R|^{|D|}) = |D| \cdot \log_2(|R|)$ bits to store $h \sim \mathcal{U}(R^D)$ $\rightsquigarrow$ for $D = \{0,1\}^{64}$: more than 2^{64} bits.

Content

What is a Hash Function?

(it depends on who you ask)

Cryptographic Hash Function

A **collision resistant** function such as $h = \texttt{sha256sum}$

```
$ sha256sum myfile.txt
018a7eaee8a...3e79043e21ab4  myfile.txt
```

Range $R = \{0,1\}^{256}$. It is hard to find x, y with $h(x) = h(y)$.

$\hookrightarrow$ Files with equal hashes are likely the same.

Cryptographic Pseudorandom Function

A function $f : \text{Seeds} \times D \to R$ where $\log_2 |\text{Seeds}|$ is small and no efficient algorithm can distinguish

- $f(s, \cdot)$ for $s \sim \mathcal{U}(\text{Seeds})$ and
- $h(\cdot)$ for $h \sim \mathcal{U}(R^D)$,

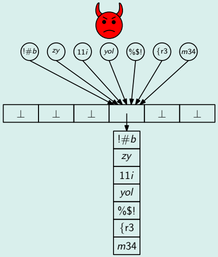
except with negligible probability.

Hash Function in Algorithm Engineering

- typically small range $|R| = \mathcal{O}(n)$
  $\hookrightarrow$ cannot be collision resistant
- should **behave like** $h \sim \mathcal{U}(R^D)$ **in my application**
- should be **fast** to evaluate
- adversarial settings rarely considered, although:

⚠ HashDoS is a thing.

However: Hash function and hash values need not be public.



Hashing in Practice

Black Magic, do not touch!

MurmurHash

Bitshifts, Magic Constants, ...

```
uint32_t murmur3_32(const uint8_t* key,
            size_t len, uint32_t seed) {
    uint32_t h = seed;
    uint32_t k;
    for (size_t i = len >> 2; i; i--) {
        memcpy(&k, key, sizeof(uint32_t));
        key += sizeof(uint32_t);
        h ^= murmur_32_scramble(k);
        h = (h << 13) | (h >> 19);
        h = h * 5 + 0xe6546b64;
    }
    [...]
    return h;
}
static inline uint32_t murmur_32_scramble(uint32_t k) {
    k *= 0xcc9e2d51;
    k = (k << 15) | (k >> 17);
    k *= 0x1b873593;
    return k;
}
```

Usage

For $R = [m]$, pick seed $\sim \mathcal{U}(\{0,1\}^{32})$ and use

$$h(x) = \texttt{murmur3\_32}(x, \text{seed}) \bmod m.$$

(should avoid modulo in practice, see `https://github.com/lemire/fastrange`)

Does *h* behave like a random function?

- **YES,** with respect to many statistical tests.
  see `https://github.com/aappleby/smhasher`
- **NO,** HashDoS attacks are known.
  see `https://en.wikipedia.org/wiki/MurmurHash#Vulnerabilities`
- **MAYBE,** for your favourite application.

Content

What should a Theorist do?

Approach 1: Ignore the Problem

Karlsruhe Institute of Technology

Simple Uniform Hashing Assumption (SUHA)

- We have access to $h \sim \mathcal{U}(R^D)$ for any R and D.
- h takes $\mathcal{O}(1)$ time to evaluate.
- h takes no space to store.

How to Analyse your Algorithm

1. *Assume* SUHA holds.
2. *Analyse* algorithm under SUHA.
3. *Hope* that algorithm still works with real hash functions.

SUHA is “wrong” but adequate

- *Modelling* assumption.
  $\hookrightarrow$ like e.g. ideal gas law in physics
- Excellent track record in non-adversarial settings.

What should a Theorist do?
Approach 2: Bring your own Hash Functions

Karlsruhe Institute of Technology

Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq R^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.
   $\hookrightarrow$ sampling and storing $h \in \mathcal{H}$ is cheap
2. Proof that algorithm with $h \sim \mathcal{U}(\mathcal{H})$ has good expected behaviour.

Remarks

- Mathematical structure of $\mathcal{H}$ must be amenable to analysis.
- *Rigorously* covers non-adversarial settings.
- Proofs often difficult.
  $\hookrightarrow$ Wider theory practice gap than with SUHA.

What should a Theorist do?

Approach 3: Let the Cryptographers do the Work

Karlsruhe Institute of Technology

How to Analyse your Algorithm using Cryptographic Assumptions

1. Analyse algorithm under *SUHA*.
2. Actually use *cryptographic pseudorandom function* f.
   - **Case 1:** Everything still works. Great! :-)
   - **Case 2:** Something fails.

     $\Rightarrow$ Your use case can tell the difference between f and true randomness.

     $\hookrightarrow$ The cryptographers said this is impossible. $\lightning$

Should we use cryptographic pseudorandom functions?

- **YES.** Algorithms become robust even in some adversarial settings.

  $\hookrightarrow$ e.g. Python, Haskell, Ruby, Rust use **SipHash** by default

  `https://en.wikipedia.org/wiki/SipHash`
- **NO.** Too slow in high-performance settings.

Hash Function	MiB / sec
SipHash	944
Murmur3F	7623
xxHash64	12109

(source: `https://github.com/rurban/smhasher`)

Content

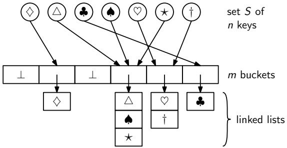
Hash Table with Chaining

Search Time under Chaining

For $n, m \in \mathbb{N}$ and a family $\mathcal{H} \subseteq [m]^D$ of hash functions the *maximum expected search time* is at most

$$T_{\text{chaining}}(n, m, \mathcal{H}) = \max_{\substack{S \subseteq D \\ |S|=n}} \max_{x \in D} \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})}\Big[1 + |\{y \in S \mid h(y) = h(x)\}|\Big]$$

⚠ Key set is *worst case*. Only $h \in \mathcal{H}$ is random. Key set is fixed *before* h is chosen.



Theorem: Hash Table with Chaining under SUHA

If $\mathcal{H} = [m]^D$ then $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + \alpha = \mathcal{O}(1)$ if $\alpha \in \mathcal{O}(1)$.

Analysis of Hash Table with Chaining under SUHA

Theorem: Hash Table with Chaining under SUHA

Let $\mathcal{H} = [m]^D$, $S \subseteq D$ with $|S| = n$ and $x \in D$ then

$$\mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + |\{y \in S \mid h(y) = h(x)\}|\Big] \leq 2 + \alpha$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + |\{y \in S \mid h(y) = h(x)\}|\Big] \\
&= \mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + \sum_{y\in S} \mathbb{1}_{\{h(y)=h(x)\}}\Big] \\
&= 1 + \sum_{y\in S} \mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[\mathbb{1}_{\{h(y)=h(x)\}}\Big]
\end{aligned}$$

$$\begin{aligned}
&= 1 + \sum_{y\in S} \Pr_{h\sim\mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
&= 1 + 1 + \sum_{y\in S\setminus\{x\}} \Pr_{h\sim\mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
&= 2 + \sum_{y\in S\setminus\{x\}} \frac{1}{m} \leq 2 + \frac{n}{m} = 2 + \alpha. \quad \square
\end{aligned}$$

Content

A Universal Hash Family

Karlsruhe Institute of Technology

Definition: c-universal hash family

A class $\mathcal{H} \subseteq [m]^D$ is called *c-universal* if: $\forall x \neq y \in D : \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x) = h(y)] \leq \frac{c}{m}$.

Note: $\mathcal{H} = [m]^D$ is 1-universal.

Reminder (?): Finite Fields

Let $\mathbb{F}_p = \{0, \ldots, p-1\}$ for a prime number p. Then $(\mathbb{F}_p, \times, \oplus)$ is a field where

$$a \times b := (a \cdot b) \bmod p \qquad \textit{and} \qquad a \oplus b := (a + b) \bmod p.$$

In particular $(\mathbb{F}_p^* := \mathbb{F}_p \setminus \{0\}, \times)$ is a group.

The class of Linear Hash Functions

Assume $D \subseteq \mathbb{F}_p$ for prime p. Then the following class is 1-universal:

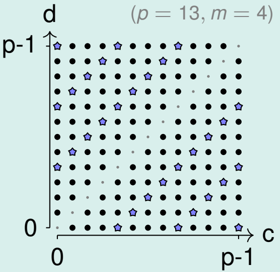
$$\mathcal{H}_{p,m}^{\text{lin}} := \{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}.$$

Proof that $\mathcal{H}^{\text{lin}}_{p,m} := \{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$ is 1-universal.

Let $x \neq y \in \mathbb{F}_p$. (To show: $\Pr_{h\sim\mathcal{H}^{\text{lin}}_{p,m}}[h(x) = h(y)] \leq 1/m$.)

- Define $\begin{aligned} c &= (a \times x) \oplus b \\ d &= (a \times y) \oplus b \end{aligned} \Leftrightarrow \begin{pmatrix} c \\ d \end{pmatrix} = \underbrace{\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix}}_{\text{regular!}} \begin{pmatrix} a \\ b \end{pmatrix}$.
- The mapping $(a, b) \mapsto (c, d)$ is a bijection (for every $x \neq y$) from $\mathbb{F}_p^{*} \times \mathbb{F}_p \to P$.
- Define *bad set* $B := \{(c, d) \in P \mid c \bmod m = d \bmod m\}$.

  $\hookrightarrow$ from picture: $\frac{|B|}{|P|} \leq \frac{1}{m}$.



$P := \mathbb{F}_p \times \mathbb{F}_p \setminus \{(b, b) \mid b \in \mathbb{F}_p\}$

$$\Pr_{a,b\sim\mathcal{U}(\mathbb{F}_p^*\times\mathbb{F}_p)}[h(x) = h(y)] = \Pr_{a,b}[((a \times x) \oplus b) \bmod m = ((a \times y) \oplus b) \bmod m]$$

$$= \Pr_{a,b}[c \bmod m = d \bmod m] = \Pr_{a,b}[(c, d) \in B] = \Pr_{c,d\sim\mathcal{U}(P)}[(c, d) \in B] = \frac{|B|}{|P|} \leq \frac{1}{m}. \quad \square$$

Analysis of Hash Table with Chaining

. . . using a Universal Hash Family

Karlsruhe Institute of Technology

Theorem

If $\mathcal{H} \subseteq [m]^D$ is a c-universal hash family then $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + c\alpha = \mathcal{O}(1)$ if $\alpha \in \mathcal{O}(1)$ and $c \in \mathcal{O}(1)$.

Proof: Mostly the same.

$$\forall S \subseteq [D], \forall x \in D : \quad \mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + |\{y \in S \mid h(y) = h(x)\}|\Big]$$

$$= \ldots = 2 + \sum_{y \in S\setminus\{x\}} \Pr_{h\sim\mathcal{U}(\mathcal{H})}[h(y) = h(x)]$$

$$= 2 + \sum_{y \in S\setminus\{x\}} \frac{c}{m} \leq 2 + \frac{cn}{m} = 2 + c\alpha. \quad \square$$

More Universal Families

Examples for Universal Hash Families

- "$((\boldsymbol{ax}+\boldsymbol{b}) \bmod \boldsymbol{p}) \bmod \boldsymbol{m}$" is 1-universal

$$\text{as discussed: } D = \mathbb{F}_p, \qquad R = [m],$$

$$\mathcal{H}^{\text{lin}}_{p,m} := \{x \mapsto ((a \times b) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$$

- "$(\boldsymbol{ax} \bmod \boldsymbol{p}) \bmod \boldsymbol{m}$" is only 2-universal:

$$D = \mathbb{F}_p, \qquad R = [m],$$

$$\mathcal{H} = \{x \mapsto (a \times b) \bmod m \mid a \in \mathbb{F}_p^*\}$$

- **Multiply-Shift** is 2-universal:

$$D = \{0, \ldots, 2^w - 1\}, \qquad R = \{0, \ldots, 2^\ell - 1\}$$

$$\mathcal{H} = \{x \mapsto \lfloor ((a \cdot x + b) \bmod 2^w)/2^{w-\ell} \rfloor \mid$$

$$\textbf{odd } a \in \{1, \ldots, 2^w - 1\}, b \in \{0, \ldots, 2^w - 1\}.\}$$

Selling point of multiply shift:

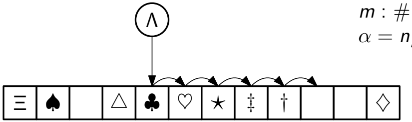
- "$x \bmod 2^w$" drops some higher order bits
- "$\lfloor x/2^{w-\ell} \rfloor$ drops some lower order bits
- No division or modulo operation needed!

For $w = 32$ (taken from Thorup 2015):

```
uint32_t hash(uint32_t x, uint32_t l, uint64_t a) {
        return (a * x + b) >> (64-l);
}
```

Content

Hash Table with Linear Probing



S : set of n keys
m : # of buckets
$\alpha = n/m$

Operations

For key x *probe* buckets $h(x), h(x)+1, h(x)+2, \dots \pmod{m}$.

Insert. Put x into first empty bucket.

Lookup. Look for x, abort when encountering empty bucket.

Delete. Lookup and remove x and ⚠ check if a key to the right wants to move into the hole.

$\hookrightarrow$ For details see `https://en.wikipedia.org/wiki/Linear_probing`

Running Times

- Lookup($x \in S$): At most x's insertion time.
- Lookup($x \notin S$): At most the time it *would take* to insert x now.
- Delete($x \in S$): At most the time it *would take* to insert $y \notin S$ with $h(y) = h(x)$.

$\hookrightarrow$ It suffices to understand insertion times!

Theorem: Linear Probing under SUHA

Let $T_{n,m}$ be the random insertion time into a linear probing hash table. If $\frac{1}{2} \le \alpha = \frac{n}{m} < \alpha_{\max}$ for some $\alpha_{\max} < 1$ then under SUHA we have

$$\mathbb{E}[T_{n,m}] = \mathcal{O}\left(\frac{1}{(1-\alpha_{\max})^2}\right) = \mathcal{O}(1). \quad \text{(not here)}$$

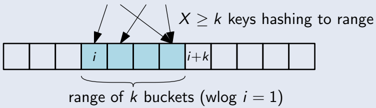
Content

Preparation: A concentration bound

Chernoff

For $X \sim Bin(n,p)$ and $\varepsilon \in [0,1]$ we have $\Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] \leq \exp(-\varepsilon^2 \mathbb{E}[X]/3)$.

Lemma: $\Pr[\geq k$ hits in segment of length $k]$



Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1,\dots,k\}\}|$.

Then $\Pr_{h \sim \mathcal{U}(R^D)}[X \geq k] \leq \exp(-(1-\alpha)^2 k/3)$.

Proof

Let $S = \{x_1,\dots,x_n\}$ and $X_i = \mathbb{1}_{\{h(x_i)\in\{1,\dots,k\}\}} \sim Ber(\frac{k}{m})$.
Then $X = \sum_{i\in[n]} X_i \sim Bin(n, \frac{k}{m})$ with $\mathbb{E}[X] = \frac{kn}{m} = \alpha k$.

$$\begin{aligned}\Pr[X \geq k] &= \Pr[X \geq \tfrac{1}{\alpha}\mathbb{E}[X]] \\ &= \Pr[X \geq (1 + \tfrac{1-\alpha}{\alpha})\mathbb{E}[X]] \\ &\leq \exp(-(\tfrac{1-\alpha}{\alpha})^2 \alpha k/3) \\ &\leq \exp(-(1-\alpha)^2 k/3). \quad (\text{using } \tfrac{1}{2} \leq \alpha \leq 1)\end{aligned}$$

Proof: Expected LP-Insertion Time under SUHA is $\mathcal{O}(1)$

$$\mathbb{E}[T] \le \mathbb{E}[B] = \sum_{k\ge1} k \cdot \Pr[B=k] = \sum_{k\ge1} k \cdot \Pr\left[\bigcup_{d=0}^{k-1} A_{h(x)-d,h(x)-d+k-1}\right]$$

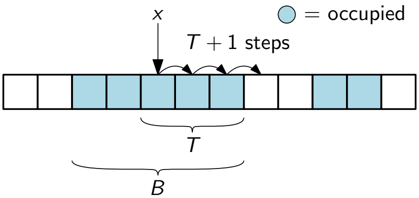
$$\stackrel{(1)}{\le} \sum_{k\ge1} k \cdot \sum_{d=0}^{k-1} \Pr\left[A_{h(x)-d,h(x)-d+k-1}\right] \stackrel{(2)}{=} \sum_{k\ge1} k \cdot k \cdot \Pr[A_{1,k}]$$

$$\stackrel{(3)}{\le} \sum_{k\ge1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1,\dots,k\}| \ge k]$$

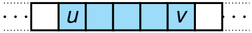
$$\stackrel{(4)}{\le} \sum_{k\ge1} k^2 \cdot \exp(-(1-\alpha)^2 k/3)$$

$$\le \sum_{k\ge1} k^2 \cdot \exp(-(1-\alpha_{\max})^2 k/3) = \mathcal{O}(1).$$

Wolfram Alpha gives: $\int_0^\infty k^2 \exp(-(1-\alpha_{\max})^2 k/3) = \frac{54}{(1-\alpha_{\max})^6}$.



$A_{u,v}$: $\{u,v\}$ is maximal occupied block:



Reasoning:

(1) Union Bound.
(2) $h(x)$ is independent of keys in the table and hash distribution is invariant under cyclic shifts.
(3) Note: Keys stored in block cannot come in from the left.
(4) Chernoff argument from previous slide.

Content

Degrees of Independence

Karlsruhe Institute of Technology

(Mutual / Collective) Independence

A family $\mathcal{E}$ of **events** is **independent** if $\forall k \in \mathbb{N}$ and distinct $E_1, \dots, E_k \in \mathcal{E}$ we have

$$\Pr\left[\bigcap_{i=1}^{k} E_i\right] = \prod_{i=1}^{k} \Pr[E_i].$$

A family $\mathcal{X}$ of discrete **random variables** is **independent** if $\forall k \in \mathbb{N}$, distinct $X_1, \dots, X_k \in \mathcal{X}$ and all $x_1, \dots, x_k \in \mathbb{R}$ we have

$$\Pr\left[\bigwedge_{i=1}^{k} X_i = x_i\right] = \prod_{i=1}^{k} \Pr[X_i = x_i].$$

Degrees of Independence

Karlsruhe Institute of Technology

Pairwise Independence

A family $\mathcal{E}$ of **events** is **pairwise independent** if for distinct $E_1, E_2 \in \mathcal{E}$ we have

$$\Pr\left[E_1 \cap E_2\right] = \Pr[E_1] \cdot \Pr[E_2].$$

A family $\mathcal{X}$ of discrete **random variables** is **pairwise independent** if for all distinct $X_1, X_2 \in \mathcal{X}$ and all $x_1, x_2 \in \mathbb{R}$ we have

$$\Pr\left[X_1 = x_1 \wedge X_2 = x_2\right] = \Pr[X_1 = x_1] \cdot \Pr[X_2 = x_2].$$

Degrees of Independence

d-wise Independence

A family $\mathcal{E}$ of **events** is d**-wise independent** if $\forall k \in \{2, \dots, d\}$ and distinct $E_1, \dots, E_k \in \mathcal{E}$ we have

$$\Pr\left[\bigcap_{i=1}^{k} E_i\right] = \prod_{i=1}^{k} \Pr[E_i].$$

A family $\mathcal{X}$ of discrete **random variables** is d**-wise independent** if $\forall k \in \{2, \dots, d\}$, distinct $X_1, \dots, X_k \in \mathcal{X}$ and all $x_1, \dots, x_k \in \mathbb{R}$ we have

$$\Pr\left[\bigwedge_{i=1}^{k} X_i = x_i\right] = \prod_{i=1}^{k} \Pr[X_i = x_i].$$

d-Independent Hash Family

Karlsruhe Institute of Technology

Definition: d-Independent Hash Family

A family $\mathcal{H} \subseteq [R]^D$ of hash functions is *d-independent* if for distinct $x_1, \ldots, x_d \in D$ and any $i_1, \ldots, i_d \in R$: (grey is implied by black)

$$\Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x_1) = i_1 \wedge \ldots \wedge h(x_d) = i_d] = \prod_{j=1}^{d} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x_j) = i_j] = |R|^{-d}.$$

Alternative Definition

$\mathcal{H}$ is d-independent if for $h \sim \mathcal{U}(\mathcal{H})$

- the family $(h(x))_{x \in D}$ of random variables is d-independent *and*
- $h(x) \sim \mathcal{U}(R)$ for each $x \in D$.

Theorem

Let $D = R = \mathbb{F}$ be a finite field. Then

$$\mathcal{H} := \{x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F}\}$$

is a d-independent family.

Note: $\mathcal{H} \subseteq \mathbb{F}^{\mathbb{F}} \rightsquigarrow$ not yet useful.

Corollary: Smaller Ranges (proof omitted)

- If m divides $|\mathbb{F}|$, then adding "mod m" gives a d-independent family $\mathcal{H}' \subseteq [m]^{\mathbb{F}}$.
- If m does not divide $|\mathbb{F}|$, then adding "mod m" gives a family $\mathcal{H}' \subseteq [m]^{\mathbb{F}}$ such that for $h \sim \mathcal{U}(\mathcal{H}')$ the family $(h(x))_{x \in \mathbb{F}}$ is d-independent but only *approximately* uniformly distributed in $[m]$.

Proof: $\mathcal{H} := \{x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \dots, a_{d-1} \in \mathbb{F}\}$ is d-independent

Let $x_1, \dots, x_d \in \mathbb{F}$ be distinct keys and $i_1, \dots i_d \in \mathbb{F}$ arbitrary.

$\hookrightarrow$ to show : $\Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j \in [d] : h(x_j) = i_j] = |\mathbb{F}|^{-d}$.

For $h \in \mathcal{H}$ (given via $a_0, \dots, a_{d-1}$) the following is equivalent:

$$\begin{matrix} h(x_1) = i_1 \\ h(x_2) = i_2 \\ \vdots \\ h(x_d) = i_d \end{matrix} \iff \begin{matrix} a_0 + a_1 x_1 + \dots + a_{d-1} x_1^{d-1} = i_1 \\ a_0 + a_1 x_2 + \dots + a_{d-1} x_2^{d-1} = i_2 \\ \vdots \\ a_0 + a_1 x_d + \dots + a_{d-1} x_d^{d-1} = i_d \end{matrix} \iff \underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ \vdots & & & \ddots & \vdots \\ 1 & x_d & x_d^2 & \dots & x_d^{d-1} \end{pmatrix}}_{\text{Vandermonde matrix } M \Rightarrow \text{ regular}} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} = \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_d \end{pmatrix}$$

Exactly one vector $\vec{a} = M^{-1} \cdot \vec{i}$ solves the equation.

$$\Rightarrow \Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j : h(x_j) = i_j] = \Pr_{a_0, \dots, a_{d-1} \sim \mathcal{U}(\mathbb{F})}[\vec{a} = M^{-1} \cdot \vec{i}\,] = \mathbb{F}^{-d}. \quad \square$$

Concentration Bound for d-Independent Variables

Karlsruhe Institute of Technology

(Tricky) Exercise

Let d be even and $X_1, \dots, X_n \sim Ber(p)$ a d-independent family of random variables with $p = \Omega(1/n)$.
Let $X = \sum_{i=1}^{n} X_i$. Then for any $\varepsilon > 0$ we have

$$\Pr[X - \mathbb{E}[X] \geq \varepsilon \mathbb{E}[X]] = \mathcal{O}(\varepsilon^{-d} \mathbb{E}[X]^{-d/2}).$$

Remark: Weaker than Chernoff, stronger than Chebyshev

Chebycheff gives $\Pr[X - \mathbb{E}[X] \geq \varepsilon \mathbb{E}[X]] \leq \frac{1-p}{\varepsilon^2 \mathbb{E}[X]}$. (requires $d = 2$)
Chernoff gave $\Pr[X - \mathbb{E}[X] \geq \varepsilon \mathbb{E}[X]] \leq \exp(-\varepsilon^2 \mathbb{E}[X]/3)$. (requires $d = n$).

Preparation: A Concentration Bound

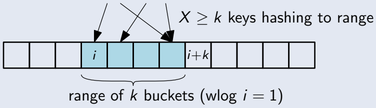
again for *d*-independence

Karlsruhe Institute of Technology

Lemma (last slide)

For d-independent $X_1, \ldots, X_n \sim Ber(p)$ and $X = \sum_{i \in [n]} X_i$ we have $\Pr[X \geq (1+\varepsilon)\mathbb{E}[X]] = \mathcal{O}(\varepsilon^{-d}\mathbb{E}[X]^{-d/2})$.

Lemma: $\geq k$ hits in segment of length k



Let $\mathcal{H}$ be a d-independent hash family and $h \sim \mathcal{U}(\mathcal{H})$.
Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1, \ldots, k\}\}|$.

Then $\Pr[X \geq k] \leq \mathcal{O}((1-\alpha)^{-d} k^{-d/2})$.

Proof

Let $S = \{x_1, \ldots, x_n\}$ and $X_i = \mathbb{1}_{\{h(x_i) \in \{1,\ldots,k\}\}} \sim Ber(\frac{k}{m})$.
Then $X = \sum_{i \in [n]} X_i$ fits the Lemma with $\mathbb{E}[X] = \frac{kn}{m} = \alpha k$.

$$\begin{aligned}
\Pr[X \geq k] &= \Pr[X \geq \tfrac{1}{\alpha}\mathbb{E}[X]] \\
&= \Pr[X \geq (1 + \tfrac{1-\alpha}{\alpha})\mathbb{E}[X]] \\
&= \mathcal{O}\Big(\Big(\frac{1-\alpha}{\alpha}\Big)^{-d} (\alpha k)^{-d/2}\Big) \\
&\leq \mathcal{O}((1-\alpha)^{-d} k^{-d/2}). \quad \text{(using } \alpha \leq 1\text{)}
\end{aligned}$$

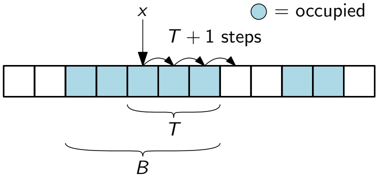
Theorem: Linear Probing with d-independence

Under the same conditions as before, except with 9-independent hash functions, the insertion time $T_{n,m}$ for linear probing satisfies:

$$\mathbb{E}[T_{n,m}] = \mathcal{O}(1)$$

Proof Sketch

$$\begin{aligned}
\mathbb{E}[T] \le \mathbb{E}[B] &\le \dots \\
&\overset{(1)}{\le} \sum_{k\ge 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1,\dots,k\}\}| \ge k] \\
&\overset{(2)}{\le} \sum_{k\ge 1} k^2 \cdot \mathcal{O}((1-\alpha)^{-8} k^{-8/2}) \\
&\le \sum_{k\ge 1} k^{-2} \cdot \mathcal{O}((1-\alpha)^{-8}) \\
&\overset{(3)}{=} \frac{\pi^2}{6}\mathcal{O}((1-\alpha)^{-8}) = \mathcal{O}(1). \quad \square
\end{aligned}$$



$A_{u,v}$: $\{u, v\}$ is a maximal occupied block:

Reasoning:

(1) Same as before, except we have to condition on $h(x)$ and may only use 8-independence in the following. (this is the hand wavy part!)

(2) Concentration bound from previous slide for $d = 8$.

(3) If interested, see 3Blue1Brown video: `https://www.youtube.com/watch?v=d-o3eB9sfls`

Final Remarks on Linear Probing + Universal Hashing

Karlsruhe Institute of Technology

Much more is known about insertion times of linear probing:

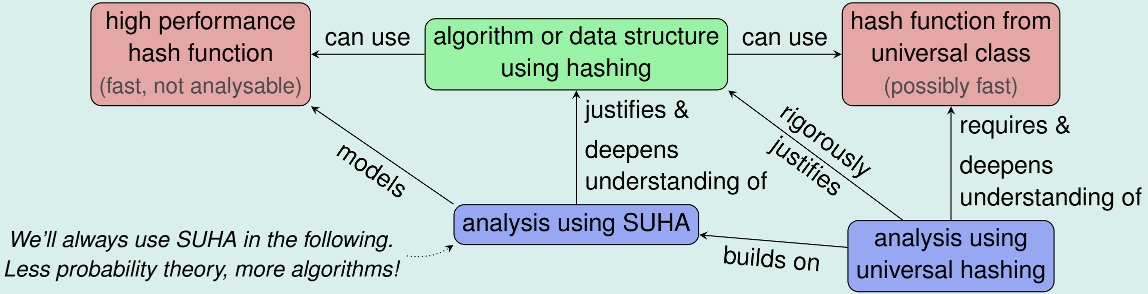
- Any 5-independent family gives $\mathcal{O}(\frac{1}{(1-\alpha)^2})$.
  $\hookrightarrow$ A. Pagh, R. Pagh, and Ruzic 2011
- An (artificially bad) 4-independent family gives $\Omega(\log n)$.
  $\hookrightarrow$ Puatracscu and Thorup 2016
- A (well-designed) 4-independent family gives $\mathcal{O}(\frac{1}{(1-\alpha)^2})$.
  $\hookrightarrow$ Puatracscu and Thorup 2013

Conclusion

Technical Takeaway: Performance of Hash Tables

For both an **ideal hash function** (SUHA) and a random hash function from a suitable **universal class**, a hash table using **linear probing** or **chaining** provably has an expected running time of $\mathcal{O}(1)$ per operation.

Non-Technical Takeaway: Approaches to analyse hashing based algorithms



Anhang: Mögliche Prüfungsfragen I

- Was könnte eine Idealvorstellung einer Hashfunktion sein? Inwiefern wäre eine ideale Hashfunktion nützlich? Was ist das Problem an dieser Vorstellung?
- Was ist die Simple Uniform Hashing Assumption (SUHA)? Was spricht dafür diese Annahme zu treffen? Welche Alternativen gibt es?
- Inwiefern ist eine pseudozufällige Funktion mit kryptographischen Ununterscheidbarkeitsgarantien nützlich für uns? Wie ist der Zusammenhang zur SUHA?*
- Universelles Hashing:
  - Wie ist c-Universalität definiert?
  - Welche c-universellen Hashklasse haben wir kennengelernt? Wie haben wir die c-Universalität bewiesen?
  - Wie ist d-Unabhängigkeit für eine Hashklasse definiert?
  - Welche d-universelle Hashklasse haben wir kennengelernt?
  - Welcher Zusammenhang besteht zwischen d-Unabhängigkeit und c-Universalität? (Übungsaufgabe)
  - Chernoff Schranken sind für Summen unabhängiger Zufallsvariablen gedacht. Was kann man machen, wenn die Zufallsvariablen nur d-unabhängig sind?*

Anhang: Mögliche Prüfungsfragen II

- Betrachten wir Hashing mit verketteten Listen:
  - Welche Schranke an die erwartete Einfügezeit haben wir bewiesen? Wie?
  - An welcher Stelle spielt die Verteilung der Hashfunktion eine Rolle?
  - Nenne eine hinreichende Eigenschaft, die eine universelle Hashklasse haben sollte, damit der Beweis funktioniert.
- Betrachten wir Hashing mit linearem Sondieren:
  - Welche Schranke an die erwartete Laufzeit haben wir bewiesen? Wie?
  - An welcher Stelle spielt die Verteilung der Hashfunktion eine Rolle?
  - Nenne eine hinreichende Eigenschaft, die eine universelle Hashklasse haben sollte, damit der Beweis funktioniert.
  - Wie wir diese Eigenschaft ausgenutzt?*

References I

[1] Anna Pagh, Rasmus Pagh, and Milan Ruzic. "Linear Probing with 5-wise Independence". In: *SIAM Rev.* 53.3 (2011), pp. 547–558. DOI: `10.1137/110827831`. URL: `https://doi.org/10.1137/110827831`.

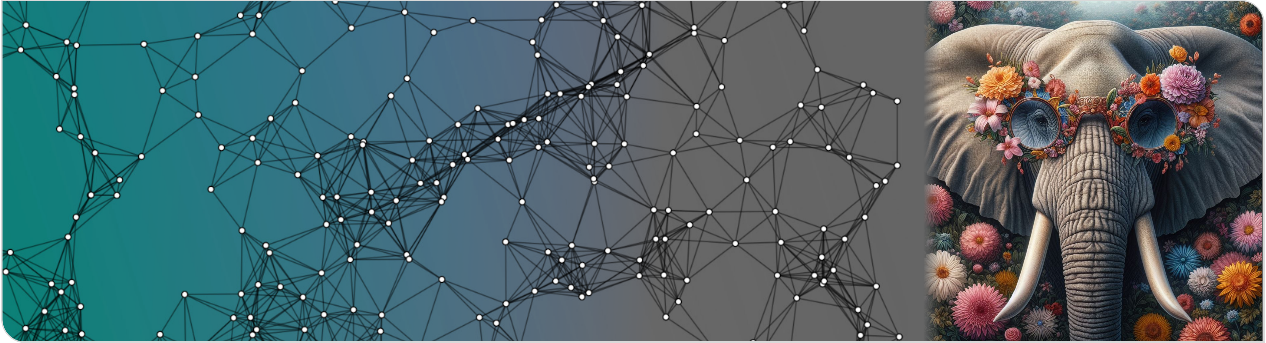
[2] Mihai Puatracscu and Mikkel Thorup. "On the *k*-Independence Required by Linear Probing and Minwise Independence". In: *ACM Trans. Algorithms* 12.1 (2016), 8:1–8:27. DOI: `10.1145/2716317`. URL: `https://doi.org/10.1145/2716317`.

[3] Mihai Puatracscu and Mikkel Thorup. "Twisted Tabulation Hashing". In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*. Ed. by Sanjeev Khanna. SIAM, 2013, pp. 209–228. DOI: `10.1137/1.9781611973105.16`. URL: `https://doi.org/10.1137/1.9781611973105.16`.

[4] Mikkel Thorup. "High Speed Hashing for Integers and Strings". In: *CoRR* abs/1504.06804 (2015). arXiv: `1504.06804`. URL: `http://arxiv.org/abs/1504.06804`.

Probability and Computing – Bloom Filters

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Reminder: SUHA

Karlsruhe Institute of Technology

Simple Uniform Hashing Assumption (SUHA)

- We have access to $h \sim \mathcal{U}(R^D)$ for any R and D.
- h takes $\mathcal{O}(1)$ time to evaluate.
- h takes no space to store.

Content

Filter = Approximate Membership Query Data Structure

Karlsruhe Institute of Technology

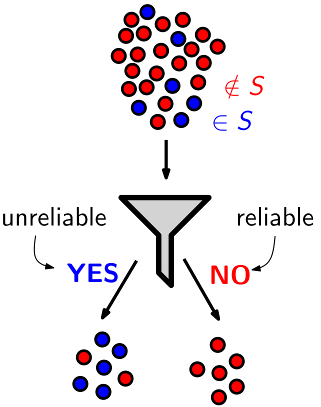
Setting

- universe D of possible keys
- a set $S \subseteq D$ of $n = |S|$
- a false positive probability ε

Want: Data structure representing S.

Space Requirement

- want $\mathcal{O}(n \log(1/\varepsilon))$ bits
- *much* smaller than $\mathcal{O}(n \log |D|)$ bits needed for hash table



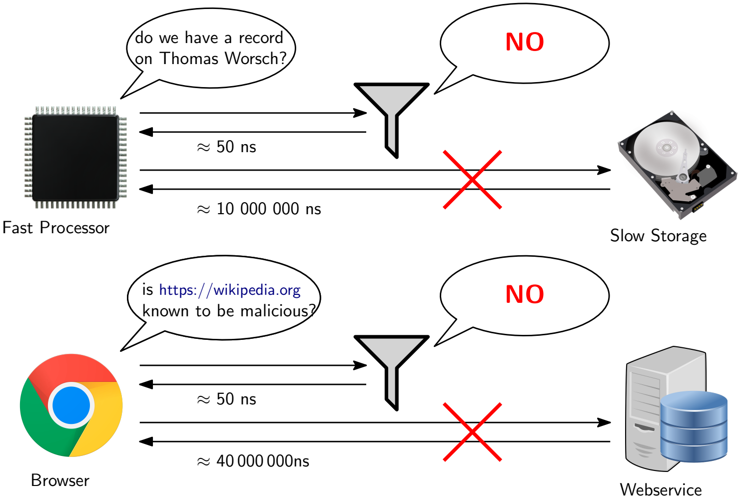
Operations

- **insert** elements to S and **delete** elements from S (optional)
- **query**: given $x \in D$ answer "is $x \in S$?" *approximately*:

$$\textbf{query}(x) = \textbf{YES} \text{ for } x \in S$$

$$\Pr[\textbf{query}(x) = \textbf{NO}] \geq 1 - \varepsilon \text{ for } x \notin S$$

Applications of Filters



General Idea

If the reliable **NO** answers are frequent, a filter access can replace a (costly) access to a reliable data structure.

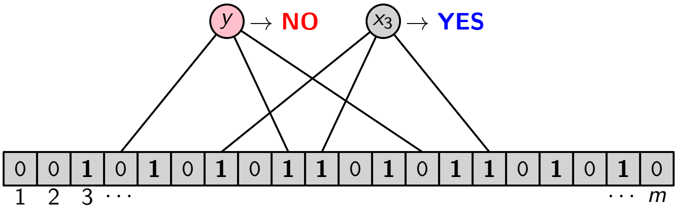
Content

The Bloom Filter Data Structure

Karlsruhe Institute of Technology

Parameters

m	length of a bit array $A[1..m]$ that we use
$k \in \mathcal{O}(1)$	number of hash functions $h_1, \dots, h_k \sim \mathcal{U}([m]^D)$
n	number of keys in $S \subseteq D$ (dynamic)
$\alpha \in \mathcal{O}(1)$	load n/m (dynamic)



insert(x):

for $i \in [k]$ **do**
  $A[h_i(x)] = 1$

query(x):

for $i \in [k]$ **do**
  if $A[h_i(x)] = 0$ **then**
    return NO
return YES

Content

1. What is a Filter or AMQ?
   - Applications of Filters

2. The Bloom Filter Data Structure

3. Analysis of Bloom Filters
   - Expected fraction of zeroes in Bloom filters
   - Optimal tuning for Bloom filters
   - Main Theorem on Bloom filters

Preparation

Karlsruhe Institute of Technology

Exercise: Some approximations of e

$$\forall n \in \mathbb{N} : (1 + \tfrac{1}{n})^n \leq e \leq (1 + \tfrac{1}{n})^{n+1}$$

$$\text{and} \quad (1 - \tfrac{1}{n})^n \leq e^{-1} \leq (1 - \tfrac{1}{n})^{n-1}.$$

Corollaries

$$\forall n \in \mathbb{N} : (1 + \tfrac{1}{n})^n = e - \mathcal{O}(1/n)$$

$$\text{and} \quad (1 - \tfrac{1}{n})^n = e^{-1} - \mathcal{O}(1/n).$$

Bloom Filter Analysis (i)

Karlsruhe Institute of Technology

Lemma

Assume $S = \{x_1, \dots, x_n\}$ is inserted into the Bloom filter. Let $(A_1, \dots, A_m) \in \{0,1\}^m$ be the random filter state and $Z := \sum_{i=1}^m (1 - A_i)$ the number of zeroes. Then

- i $\mathbb{E}[\frac{Z}{m}] = (1 - \frac{1}{m})^{m\alpha k} = e^{-\alpha k} - o(1)$
- ii For $y \notin S$: $\Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = (1 - \frac{z}{m})^k$

0	0	**1**	0	**1**	0	**1**	0	**1**	**1**	0	**1**	0	**1**	**1**	0	**1**	0	**1**	0
1	2	3 ···																···	m

Proof of (i).

$$\mathbb{E}[\tfrac{Z}{m}] = \tfrac{1}{m}\mathbb{E}[\sum_{i=1}^{m}(1 - A_i)] = \tfrac{1}{m}\sum_{i=1}^{m}\Pr[A_i = 0] = \tfrac{1}{m}\sum_{i=1}^{m}\Pr[A_1 = 0] = \Pr[A_1 = 0]$$

$$= \Pr[\forall x \in S : \forall i \in [k] : h_i(x) \neq 1] \overset{SUHA}{=} \prod_{x \in S}\prod_{i \in [k]}\Pr[h_i(x) \neq 1] \overset{SUHA}{=} \prod_{x \in S}\prod_{i \in [k]}(1 - \tfrac{1}{m})$$

$$= (1 - \tfrac{1}{m})^{nk} = (1 - \tfrac{1}{m})^{m\alpha k} = (e^{-1} - o(1))^{\alpha k} = e^{-\alpha k} - o(1).$$

Bloom Filter Analysis (i)

Lemma

Assume $S = \{x_1, \dots, x_n\}$ is inserted into the Bloom filter. Let $(A_1, \dots, A_m) \in \{0,1\}^m$ be the random filter state and $Z := \sum_{i=1}^{m}(1 - A_i)$ the number of zeroes. Then

- i $\mathbb{E}[\frac{Z}{m}] = (1 - \frac{1}{m})^{m\alpha k} = e^{-\alpha k} - o(1)$
- ii For $y \notin S$: $\Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = (1 - \frac{z}{m})^k$

0	0	**1**	0	**1**	0	**1**	0	**1**	**1**	0	**1**	0	**1**	**1**	0	**1**	0	**1**	0
1	2	3	$\cdots$															$\cdots$	m

Proof of (ii).

$$\Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = \Pr[\forall i \in [k] : A_{h_i(y)} = 1 \mid Z = z] = \prod_{i \in [k]} \left(\frac{m - z}{m}\right) = (1 - \tfrac{z}{m})^k.$$

How should a Bloom filter be configured?

Karlsruhe Institute of Technology

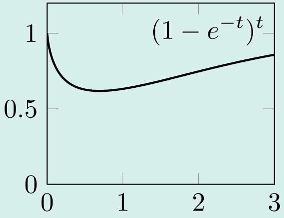
Approximate false positive rate

From the previous Lemma we get for $y \notin S$:

$$\varepsilon = \Pr[\textbf{query}(y) = \textbf{YES}] \approx \Pr[\textbf{query}(y) = \textbf{YES} \mid Z = \mathbb{E}[Z]]$$
$$\overset{\text{ii}}{=} \left(1 - \frac{\mathbb{E}[Z]}{m}\right)^k \overset{\text{i}}{=} (1 - e^{-\alpha k} + o(1))^k \approx (1 - e^{-\alpha k})^k.$$

Which k minimises ε? (when α is fixed)

$$\begin{aligned} & \arg\min_{k \in \mathbb{N}} (1 - e^{-\alpha k})^k \\ = \; & \arg\min_{k \in \mathbb{N}} (1 - e^{-\alpha k})^{\alpha k} \\ \approx \; & \frac{1}{\alpha} \arg\min_{t \in \mathbb{R}_+} (1 - e^{-t})^t \\ = \; & \frac{1}{\alpha} \arg\min_{t \in \mathbb{R}_+} t \ln(1 - e^{-t}) \end{aligned}$$



- plot $(1 - e^{-t})^t \rightsquigarrow$ one global minimum.
- deriving $t \ln(1 - e^{-t})$ gives $\ln(1 - e^{-t}) + \frac{te^{-t}}{1 - e^{-t}}$
- $t = \ln(2)$ is root of the derivative.

$\hookrightarrow$ $k = \ln(2)/\alpha$ is optimal for fixed α.
$\hookrightarrow$ choose α and k such that $\alpha k = \ln(2)$

False Positive Probability and Space

Karlsruhe Institute of Technology

Intuition for optimality of $\alpha k = \ln(2)$

- gives $\mathbb{E}[\frac{Z}{m}] \approx e^{-\alpha k} = \frac{1}{2}$
- maximises *entropy* of the filter bits

Theorem

A Bloom filter with $k \in \mathbb{N}$ hash functions and load factor $\alpha = \ln(2)/k$ has

space requirement $m = n/\alpha = \frac{kn}{\ln 2} \approx 1.44kn$ bits and

false positive probability $\varepsilon = 2^{-k} + o(1)$.

- space requirement ✓
- false positive probability: need a concentration bound first.

Concentration bound for Z

Reminder: McDiarmid's Inequality

If $X_1, \ldots, X_n$ are indepedent and f satisfies the bounded difference property with parameters $(\Delta_i)_{i \in [n]}$ then

$$\Pr[\mathbb{E}[f(X_1, \ldots, X_n)] - f(X_1, \ldots, X_n) \geq \delta] \leq \exp\left(\frac{-2\delta^2}{\sum_{i=1}^n \Delta_i^2}\right).$$

Lemma

- (i) $\Pr[Z \leq \mathbb{E}[Z] - \delta \quad] \leq \exp(-\Theta(\delta^2/m))$ for any $\delta > 0$,
- (ii) $\Pr[Z \leq \mathbb{E}[Z] - m^{2/3}] \leq \exp(-\Theta(m^{1/3}))$ by setting $\delta = m^{2/3}$.

Proof of (i) using the method of bounded differences.

- Z is a function of kn independent hash values
- each hash value can change Z by at most 1
- use method of bounded differences!

$$\Rightarrow \Pr[Z \leq \mathbb{E}[Z] - \delta] \leq \Pr[\mathbb{E}[Z] - Z \geq \delta] = \exp\left(\frac{-2\delta^2}{nk}\right) = \exp\left(\frac{-2\delta^2}{m\alpha k}\right) = \exp\left(\frac{-2\delta^2}{m\ln(2)}\right). \quad \square$$

False Positive Probability of Bloom filters

Karlsruhe Institute of Technology

Proof of the Main Theorem on Bloom filters (false positive probability).

By choice of k and α we have $\mathbb{E}[\frac{Z}{m}] = e^{-\alpha k} - o(1) = \frac{1}{2} - o(1)$.
Let $y \notin S$ and $B = \lfloor \mathbb{E}[Z] - m^{2/3} \rfloor$.

$$\Pr[\textbf{query}(y) = \textbf{YES}] \overset{\text{LTP}}{=} \sum_{z=1}^{m} \Pr[Z = z] \cdot \Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = \sum_{z=1}^{m} \Pr[Z = z] \cdot (1 - \tfrac{z}{m})^k$$

$$\leq \sum_{z=1}^{B} \Pr[Z = z] + \sum_{z=B+1}^{m} \Pr[Z = z](1 - \tfrac{B+1}{m})^k \leq \Pr[Z \leq B] + (1 - \tfrac{B+1}{m})^k$$

$$\leq \Pr[Z \leq \mathbb{E}[Z] - m^{2/3}] + \left(1 - \frac{\mathbb{E}[Z] - m^{2/3}}{m}\right)^k \overset{\text{ii}}{\leq} \exp(-\Theta(m^{1/3})) + (1 - \tfrac{1}{2} + o(1))^k = 2^{-k} + o(1). \quad \square$$

How to Configure Your Bloom Filter

Karlsruhe Institute of Technology

Theorem

A Bloom filter with $k \in \mathbb{N}$ hash functions and load factor $\alpha = \ln(2)/k$ has

space requirement $m = n/\alpha = \frac{kn}{\ln 2} \approx 1.44kn$ bits and

false positive probability $\varepsilon = 2^{-k} + o(1)$.

How to determine m and k (the parameters you actually need)

1. n: determined by input
2. ε: choose a trade-off between space usage and false positive probability
   - If utility comes from negative answers "$x \notin S$, definitely" and running time is negligible, then:
     - want to maximise utility $-$ disutility, where: ($\propto$ means "proportional to")
     - utility $\propto \frac{\text{negative answers}}{\text{second}} = \frac{\text{queries}}{\text{second}} \cdot \Pr[x \notin S] \cdot (1-\varepsilon)$
     - disutility $\propto$ space consumption $= 1.44 \log(1/\varepsilon)n$ bits of RAM or cache
3. compute $k = \lceil \log(1/\varepsilon) \rceil$ // effectively restricts ε to powers of 2
4. compute $\alpha = \ln(2)/k$ and $m = \lceil n/\alpha \rceil$

Remarks

Much, much more is known

- more functionality
  - $\hookrightarrow$ counting Bloom filters support deletions
- better space efficiency
  - $\hookrightarrow$ cuckoo filters use $n \log(1/\varepsilon) + \mathcal{O}(n)$ bits rather than $\approx 1.44 n \log(1/\varepsilon)$ bits
  - $\hookrightarrow$ static filters (no insertions or deletions) use $n \log(1/\varepsilon) + o(n)$ bits.
- better query times
  - $\hookrightarrow$ blocked Bloom filters improve cache efficiency
- ...

Conclusion

- **Approximate Membership Queries.**
  - Decide "is $x \in S$?" with *false positive probability* ε.
  - The Bloom filter is the most widespread AMQ.
- **Space Efficient.** $\approx 1.44 \log(1/\varepsilon)$ bits per element
  - often fit into cache or RAM when proper set data structure does not
- **Used to prevent costly accesses.**
  - Reliable on **NO** answers.
  - Useful if **NO** answers are frequent.

Anhang: Mögliche Prüfungsfragen I

- Approximate-Membership-Query Datenstrukturen im Allgemeinen
  - Welche Aufgabe hat eine AMQ Datenstruktur?
  - Was ist der Vorteil gegenüber einer exakten Datenstruktur?
  - Was wäre ein Anwendungsfall, in dem eine AMQ Datenstruktur nützlich ist?
- Bloomfilter
  - Wie ist ein Bloomfilter aufgebaut und welche Operationen unterstützt er?
  - Welche Parameter gibt es, und wie hängen diese zusammen?
  - Was hat unsere Analyse zur geschickten Wahl der Parameter zu sagen? Wie werden die übrigen Parameter gewählt? Welcher Speicherverbrauch ergibt sich?
  - Fragen zur Analyse
    - Welche Anzahl von Nullen bzw. Einsen erwarten wir?
    - Wie hängt die falsch-positiv Wahrscheinlichkeit mit der Anzahl Nullen bzw. Einsen zusammen?
    - Wir kann man argumentieren, dass die Anzahl Nullen bzw. Einsen im Bloomfilter nahe am Erwartungswert liegt?

Ablauf der letzten Termine

Inverted Classroom Grundidee

- Zu Hause: Videovorlesung gucken.
- Vor Ort: Übungsaufgaben mit Hilfestellung bearbeiten.
  $\hookrightarrow$ Weniger oder keine Übungen mehr zuhause.

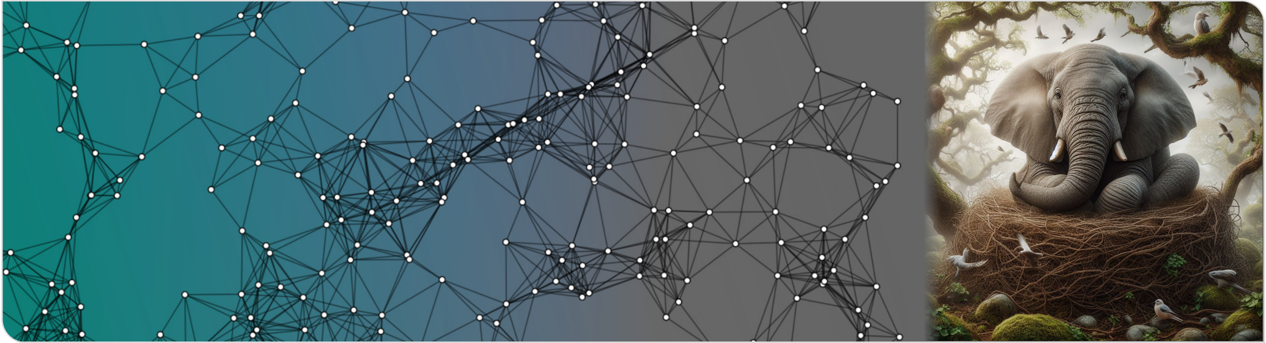
Ablauf der restlichen Termine

- **Do 24.1: reguläre Vorlesung zu klassischen Hashtabellen und Bloom Filtern** **(mit Stefan)**
- **Di 30.1: reguläre Übung zu Blättern 10 + 11** **(mit Hans-Peter)**
- Video gucken (Cuckoo Hashing, 30 min)
- Blatt 12 abgeben (Bloom Filter, nur 1 Aufgabe)
- **Do 1.2: reguläre Übung zu Blatt 12, Bearbeitung von Blatt 13 zu Cuckoo Hashing** **(mit Stefan)**
- Video gucken (Peeling)
- Blatt 13 (finalisieren und) abgeben
- **Do 8.2: Bearbeitung von Blatt 14 zu Peeling** **(mit Stefan)**
- Video gucken (Perfect Hashing)
- Blatt 14 (finalisieren und) abgeben
- **Di 13.2: Bearbeitung von Blatt 15 zu Perfect Hashing** **(mit Stefan)**
- Blatt 15 (finalisieren und) abgeben
- **Do 15.2: Termin reserviert für Fragen, Prüfungsmodalitäten usw.** **(mit allen)**

Karlsruhe Institute of Technology

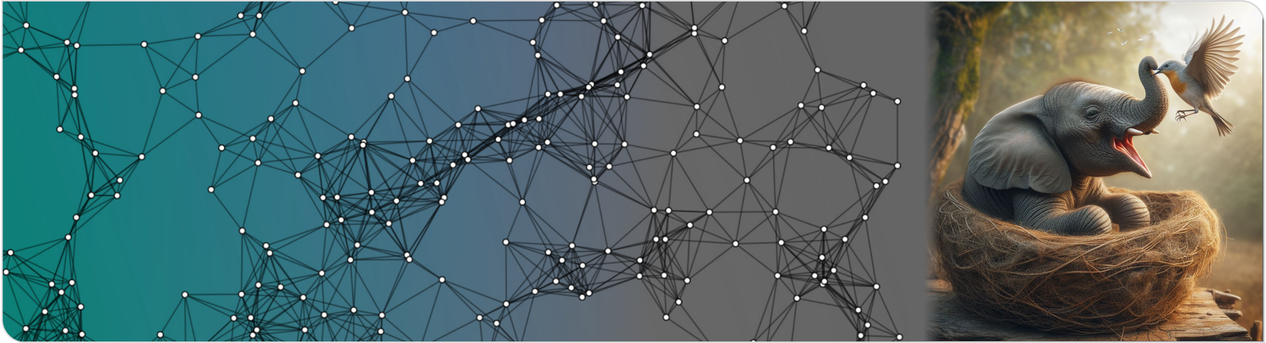
Probability and Computing – Cuckoo Hashing

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Probability and Computing – Cuckoo Hashing

Stefan Walzer, Maximilian Katzmann | WS 2023/2024

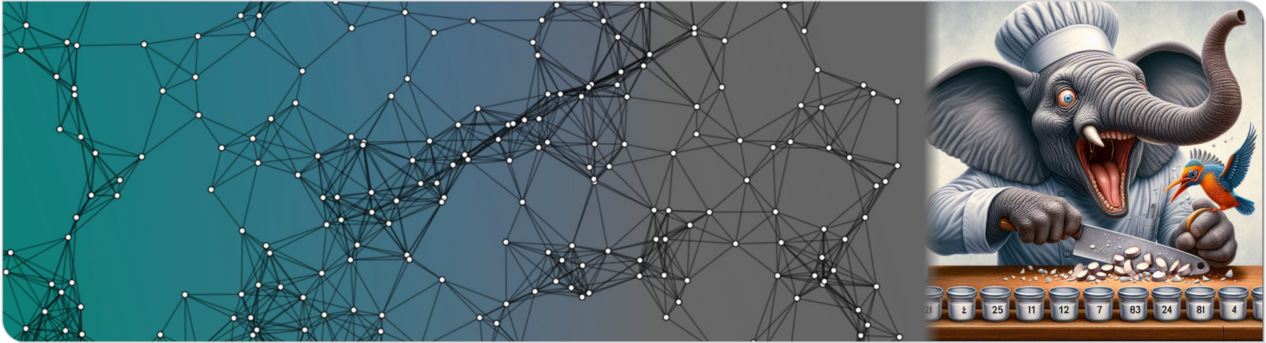


Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Cuckoo Hashing

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



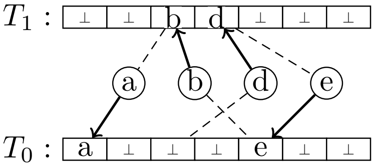
Content

1. **Cuckoo Hashing**
   - Algorithm
   - Analysis

Cuckoo Hashing

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	(⚠ load factor $\alpha = \frac{n}{2m}$)



Algorithm lookup(x)**:**
- **return** $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

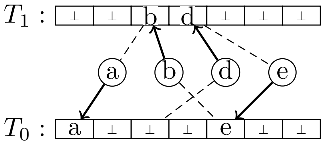
Algorithm delete(x)**:**
- **if** $T_0[h_0(x)] = x$ **then**
  - $T_0[h_0(x)] \leftarrow \perp$
- **else if** $T_1[h_1(x)] = x$ **then**
  - $T_1[h_1(x)] \leftarrow \perp$

Algorithm insert(x)**:**
- **for** $i = 0$ **to** LIMIT **do**
  - $b \leftarrow i \bmod 2$
  - swap(x, $T_b[h_b(x)]$)
  - **if** $x = \perp$ **then**
    - **return** SUCCESS
- **return** FAILURE

Cuckoo Hashing Theorem

Algorithm insert(x)**:**

- **for** $i = 0$ **to** LIMIT **do**
  - $b \leftarrow i \bmod 2$
  - swap(x, $T_b[h_b(x)]$)
  - **if** $x = \perp$ **then**
    - **return** SUCCESS
- **return** FAILURE



Theorem (Analysis with LIMIT $= \infty$)

Assume we insert all $x \in S$ and then another key y. Let E be the event that this succeeds and

$$T = \begin{cases} \text{insertion time of } y & \text{if } E \text{ occurs} \\ 0 & \textit{otherwise} \end{cases}$$

Then **i** $\Pr[E] = 1 - \mathcal{O}(1/m)$ and **ii** $\mathbb{E}[T] = \mathcal{O}(1)$.

Theorem (full analysis, not here)

If we

- set LIMIT $= \Omega(\log n)$ appropriately
- rebuild the table with fresh hash functions when LIMIT is reached

we obtain a hash table where lookup and delete take $\mathcal{O}(1)$ time and insert takes *expected* $\mathcal{O}(1)$ time.

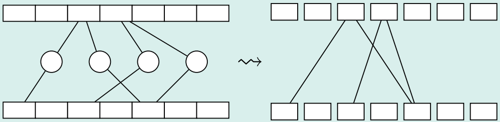
Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

The Cuckoo Graph

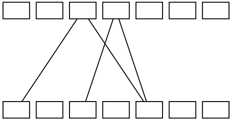
Consider the bipartite *cuckoo graph*

$$G = ([m], [m], \{(h_0(x), h_1(x)) \mid x \in S\})$$

the key x corresponds to the edge $(h_0(x), h_1(x))$ and each table position to a vertex.



Proof of i: Success probability is $1 - \mathcal{O}(1/m)$



Keys and buckets in the infinite loop

Assume $\bar{E}$ occurs, i.e. an insertion fails due to an infinite loop. Let $G^* = (V^*, E^*)$ be the subgraph of G with

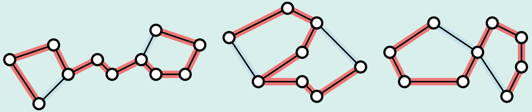
- V^*: table positions touched infinitely often
- E^*: keys touched infinitely often.

Properties of G^*:

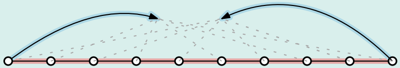
- connected
- $|E^*| = |V^*| + 1$ can you see why?
- $\deg_{E^*}(v) \geq 2$.

Possibilities for G^*

There are three options:

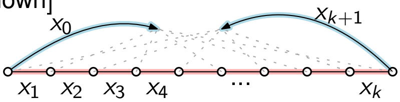


In all three cases: Simple path through $|V^*|$ and two extra edges connecting inwards:



Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

$$
\begin{aligned}
\Pr[\bar{E}] &= \Pr[\exists \text{path as shown}] \\
&= \Pr[\exists k \in \mathbb{N} : \exists x_0, \ldots, x_{k+1} \in S : x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0, \ldots, x_{k+1} \in S} \Pr[x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{a} \cdot \underbrace{2}_{b} \cdot \underbrace{\tfrac{1}{m^{k+1}}}_{c} \cdot \underbrace{\left(\tfrac{k+1}{2m}\right)^2}_{d} \\
&\leq \tfrac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2} (1-\beta)^{k+2} (k+1)^2 \\
&\leq \tfrac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2} (k+1)^2 = \tfrac{1}{m} \cdot \mathcal{O}\left(\tfrac{1}{\beta^3}\right) = \tfrac{1}{m} \cdot \mathcal{O}(1) \quad \square
\end{aligned}
$$

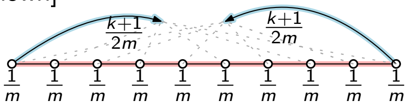


- a Choose sequence of $k+2$ keys.
- b Choose to start in left or right table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

Karlsruhe Institute of Technology

$$\Pr[\bar{E}] = \Pr[\exists \text{path as shown}]$$
$$= \Pr[\exists k \in \mathbb{N} : \exists x_0, \ldots, x_{k+1} \in S : x_0, \ldots, x_{k+1} \text{ form a path as shown}]$$
$$\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0,\ldots,x_{k+1} \in S} \Pr[x_0, \ldots, x_{k+1} \text{ form a path as shown}]$$
$$\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{a} \cdot \underbrace{2}_{b} \cdot \underbrace{\tfrac{1}{m^{k+1}}}_{c} \cdot \underbrace{\left(\tfrac{k+1}{2m}\right)^2}_{d}$$
$$\leq \tfrac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2} (1-\beta)^{k+2} (k+1)^2$$
$$\leq \tfrac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2} (k+1)^2 = \tfrac{1}{m} \cdot \mathcal{O}\left(\tfrac{1}{\beta^3}\right) = \tfrac{1}{m} \cdot \mathcal{O}(1) \qquad \square$$



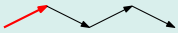
- a Choose sequence of $k+2$ keys.
- b Choose to start in left or right table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of $\boxed{\text{ii}}$: Expected insertion time is $\mathcal{O}(1)$

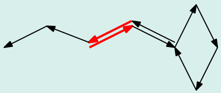
Lemma

If the insertion of y *takes* $t \in \mathbb{N}$ *steps then the cuckoo graph* G *contained (previously) a path of length* $\lceil (t-2)/3 \rceil$ *starting from* $h_0(y)$ *or from* $h_1(y)$.

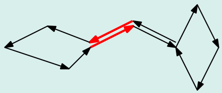
Proof.



no turning back
$\rightsquigarrow$ path of length $t-1$
starting from $h_0(y)$



turn back once
$\rightsquigarrow$ path of length $\lceil (t-2)/3 \rceil$
starting from $h_0(y)$ or $h_1(y)$



turn back twice
impossible: insertion would fail

$\square$

Proof of ii: Expected insertion time is $\mathcal{O}(1)$ (continued)

$$\mathbb{E}[T] = \sum_{t\geq 1} \Pr[T \geq t] \qquad \text{(see complexity classes slide 10)}$$

$$\leq \sum_{t\geq 1} \Pr[\exists \text{path of length } \lceil (t-2)/3 \rceil \text{ starting from } h_0(y) \text{ or } h_1(y)] \qquad \text{by Lemma}$$

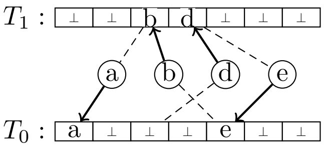
$$\leq 2 \cdot \sum_{t\geq 1} \Pr[\exists \text{path of length } \lceil (t-2)/3 \rceil \text{ starting from } h_0(y)] \qquad \text{union bound + symmetry}$$

$$\leq 2\left(2 + 3 \cdot \sum_{t\geq 1} \Pr[\exists \text{path of length } t \text{ starting from } h_0(y)]\right) \qquad \sum_{i\geq 1} f(\lceil t/3 \rceil) = 3 \cdot f(1) + 3 \cdot f(2) + \ldots$$

$$\leq 4 + 6 \cdot \sum_{t\geq 1} \sum_{x_1,\ldots,x_t \in S} \Pr[x_1,\ldots,x_t \text{ form path starting from } h_0(y)] \qquad \text{union bound}$$

$$\leq 4 + 6 \cdot \sum_{t\geq 1} n^t m^{-t} = 6 \sum_{t\geq 0} (1-\beta)^t = \mathcal{O}(1/\beta) = \mathcal{O}(1). \quad \square$$

Conclusion



Cuckoo Hashing

- hash table with *worst case* constant access times
- analysis considers path in graphs similar to the Erdős-Renyi model
- many variations and spin-offs (not discussed here)

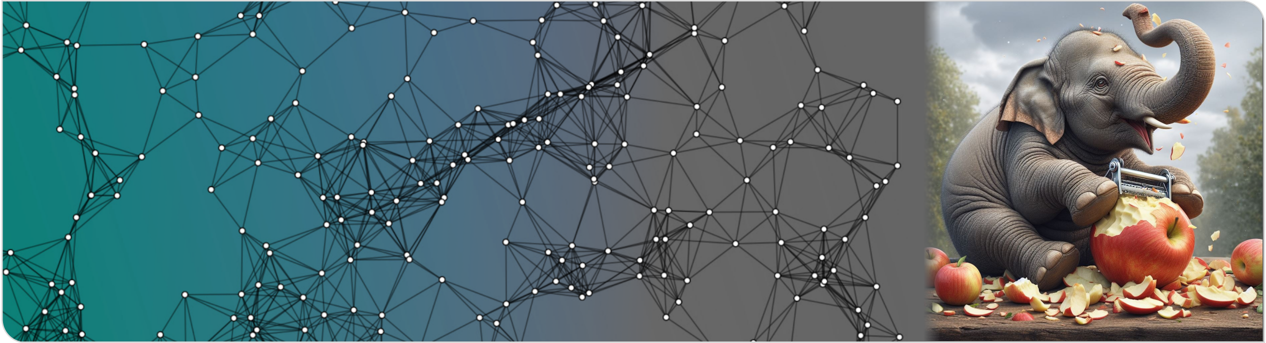
Anhang: Mögliche Prüfungsfragen I

Karlsruhe Institute of Technology

- Was ist und was kann Cuckoo Hashing?
  - Was ist die Grundidee? Wie funktionieren die Operationen?
  - Worauf ist bei der Wahl der Tabellengröße / beim Load Factor zu achten?
  - Was kann man über die Laufzeit der Operationen sagen?
  - Welche Vorteile und Nachteile ergeben sich im Vergleich zu anderen Techniken wie linearem Sondieren?
- Analyse:
  - Eine Einfügung, die fehlschlägt, entspricht gewissen Strukturen im Cuckoo-Graphen. Welchen?
  - Wie haben wir gezeigt, dass solche Strukturen unwahrscheinlich sind?
  - Wie haben wir die erwartete Einfügezeit abgeschätzt?

Probability and Computing – The Peeling Algorithm

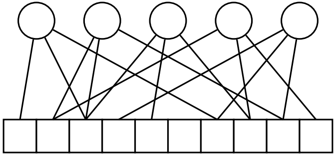
Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Content

Content

Cuckoo Hashing with one table and *k* hash functions



$n \in \mathbb{N}$ keys
$m \in \mathbb{N}$ table size
$\alpha = \frac{n}{m}$ load factor
$h_1, \ldots, h_k \sim \mathcal{U}([m]^D)$ hash functions
$\hookrightarrow$ Could also use a separate table per hash function.

randomWalkInsert(x)

while $x \neq \perp$ **do** // TODO: limit
  sample $i \sim \mathcal{U}([k])$
  swap($x, T[h_i(x)]$)

(some improvements possible)

Theorem (without proof)

For each $k \in \mathbb{N}$ there is a **threshold** c_k^* such that:

- if $\alpha < c_k^*$ all keys can be placed with probability $1 - \mathcal{O}(\frac{1}{m})$.
- if $\alpha > c_k^*$ **not** all keys can be placed with probability $1 - \mathcal{O}(\frac{1}{m})$.

$c_2^* = \frac{1}{2}, \quad c_3^* \approx 0.92, \quad c_4^* \approx 0.98, \ldots$

Conjecture

If $\alpha < c_k^*$ then the expected number of steps of successful insertions is $\mathcal{O}(1)$.
$\hookrightarrow$ several proof attempts for random walk and other algorithms exist, with *partial* success

Static Hash Tables

Static Hash Table

`construct`(S): builds table T with key set S
`lookup`(x): checks if x is in T or not
$\hookrightarrow$ no insertions or deletions after construction!

Constructing cuckoo hash tables:

- solved by Khosla 2013: “Balls into Bins Made Faster”
- matching algorithm resembling preflow push
- expected running time $\mathcal{O}(n)$, finds placement whenever one exists
- not in this lecture

Greedily constructing cuckoo hash tables

- Peeling algorithm: simple but sophisticated analysis
- interesting applications beyond hash tables (see “retrieval” in next lecture)

Content

The Peeling Algorithm

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\perp, \dots, \perp]$ // empty table of size m

while $\exists i \in [m] : \exists$ *exactly one* $x \in S : i \in \{h_1(x), h_2(x), h_3(x)\}$ **do**

   // x is only unplaced key that may be placed in i

   $T[i] \leftarrow x$

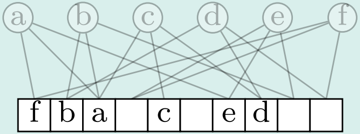
   $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

   return T

else

   return NOT-PEELABLE



Exercise

- Success of constructByPeeling does not depend on choices for i made by while.
- constructByPeeling can be implemented in linear time.

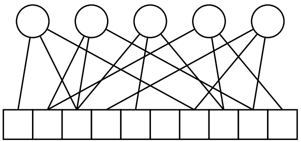
Peelability and the Cuckoo Graph

Cuckoo Graph and Peelability

- The **Cuckoo Graph** is the bipartite graph

$$G_{S,h_1,h_2,h_3} = (S, [m], \{(x, h_i(x)) \mid x \in S, i \in [3]\})$$

- Call G_{S,h_1,h_2,h_3} **peelable** if constructByPeeling(S, h_1, h_2, h_3) succeeds.
- If $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ then the distribution of G_{S,h_1,h_2,h_3} does not depend on S. We then simply write $G_{m,\alpha m}$.
  - m □-nodes and $\lfloor \alpha m \rfloor$-○-nodes
  - think: α is constant and $m \to \infty$.



Peeling simplified (not computing placement)

while $\exists$ □*-node of degree* 1 **do**
  remove it and its incident ○

G is peelable if and only if this algorithm removes all ○-nodes.

Content

Peeling Theorem

Karlsruhe Institute of Technology

Peeling Threshold

Let $c_3^\Delta = \min_{y\in[0,1]} \frac{y}{3(1-e^{-y})^2} \approx 0.81$.

Theorem (today's goal)

Let $\alpha < c_3^\Delta$. Then $\Pr[G_{m,\alpha m}$ is peelable$] = 1 - o(1)$.

Remark: More is known.

- For "$\alpha < c_3^\Delta$" we get "peelable" with probability $1 - \mathcal{O}(1/m)$.
- For "$\alpha > c_3^\Delta$" we get "not peelable" with probability $1 - \mathcal{O}(1/m)$.
- Corresponding thresholds c_k^Δ for $k \geq 3$ hash functions are also known.

Exercise: What about $k = 2$?

Peeling does not reliably work for $k = 2$ for any $\alpha > 0$.

Peeling Theorem: Proof outline

Theorem (today's goal)

Let $\alpha < c_3^\Delta$. Then $\Pr[G_{m,\alpha m}$ is peelable$] = 1 - o(1)$.

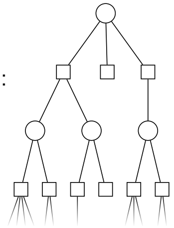
Proof Idea

The random (possibly) infinite tree T_α can be peeled for $\alpha < c_3^\Delta$ and T_α is locally like $G_{m,\alpha m}$.

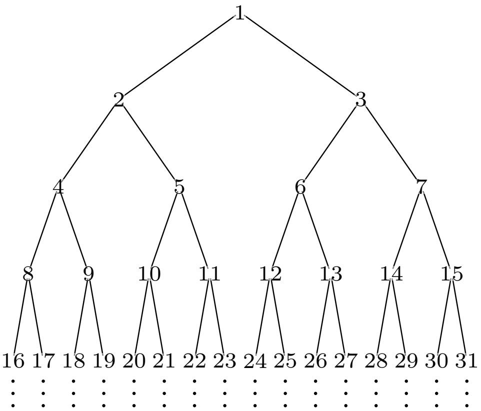
Steps

- I What is an infinite tree in general?
- II What is T_α in particular?
- III What does peeling mean in this setting?
- IV What role does c_3^Δ play?
- V What does it mean for T_α to be locally like $G_{m,\alpha m}$?
- VI What is the probability that a fixed key of $G_{m,\alpha m}$ is peeled?
- VII What is the probability that *all* keys of $G_{m,\alpha m}$ are peeled?

T_α :



What is an infinite tree in general?



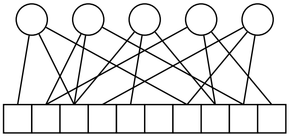
$V = \mathbb{N}$

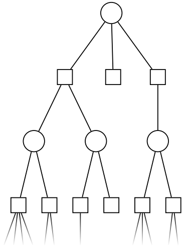
$E = \{\{n, 2n\} \mid n \in \mathbb{N}\} \cup \{\{n, 2n+1\} \mid n \in \mathbb{N}\}.$

Tree Definitions

- connected and acyclic ✓
  sensible and satisfied
- connected and $|E| = |V| - 1$ ✗
  not sensible

What is T_α in particular?





Observations for the finite Graph $G_{m,\alpha m}$

- each ◯ has 3 □ as neighbours (rare exception: $h_1(x), h_2(x), h_3(x)$ not distinct)
- each □ has random number X of ◯ as neighbours with $X \sim Bin(3n, \frac{1}{m}) = Bin(3\lfloor \alpha m \rfloor, \frac{1}{m})$. In an exercise you'll show

$$\Pr[X = i] \xrightarrow{m \to \infty} \Pr_{Y \sim Pois(3\alpha)}[Y = i].$$

Definition of the (possibly) infinite random tree T_α

- root is ◯ and has three □ as children
- each □ has random number of ◯ children, sampled $Pois(3\alpha)$ (independently for each □).
- each non-root ◯ has two □ as children.

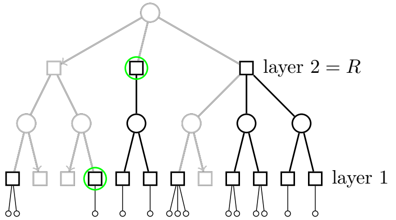
Remark: T_α is finite with positive probability > 0, e.g. when the first three $Pois(3\alpha)$ random variables come out as 0. But T_α is also infinite with positive probability.

iii What does peeling mean in this setting?

Peeling Algorithm

while $\exists$ *childless* $\square$*-node* **do**
remove it and its incident $\bigcirc$

$\hookrightarrow$ not well defined outcome on T_α!
$\hookrightarrow$ but well defined on T_α^R!



Peel only the first $R \in \mathbb{N}$ layers

- Let T_α^R be the first $2R+1$ levels of T_α.
- *R layers* of $\square$-nodes, labeled bottom to top.
- Run peeling on T_α^R (later $R \to \infty$).

$\hookrightarrow$ Why not consider the first $2R$ levels? (without $+1$)

Only care whether root is removed (root represents arbitrary node in $G_{m,\alpha m}$)

We may then simplify the peeling algorithm.

- replace "$\square$-node of degree 1" condition with stronger "childless $\square$-node".
  - prevents peeling of $\square$-nodes with one child and no parent
  - no matter: such nodes are disconnected from the root anyway
- whether node is peeled only depends on subtree
  $\hookrightarrow$ one bottom up pass suffices for peeling

What does peeling mean in this setting? (2)

Observation

Let $q_R = \Pr[\text{root survives when peeling } T_\alpha^R]$.
The values q_R are decreasing in R.

Peeling Algorithm

while $\exists$ *childless* $\square$*-node* **do**
  remove it and its incident $\bigcirc$

Proof.

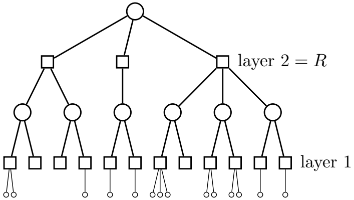
Assume when peeling T_α^R the sequence $\vec{x} = (x_1, \ldots, x_k)$ is a valid sequence of $\square$-node choices. Then $\vec{x}$ is also valid when peeling T_α^{R+1}.

$$\text{peeling } T_\alpha^R \text{ removes the root} \Rightarrow \text{peeling } T_\alpha^{R+1} \text{ removes the root}$$

$$\text{root survives when peeling } T_\alpha^{R+1} \Rightarrow \text{peeling } T_\alpha^R \text{ removes the root}$$

$$q_{R+1} \le q_R$$

$\square$

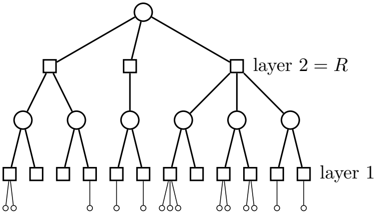


iii What does peeling mean in this setting? (3)

Karlsruhe Institute of Technology

Peeling T_α^R bottom up

for $i = 1$ **to** R **do** // $\square$-layers bottom to top
  for each *$\square$-node v in layer i* **do**
    if *v has no children* **then**
      remove v and its parent $\bigcirc$



Survival probabilities $p_i :=$ Pr[$\square$-node in layer i is *not* peeled]

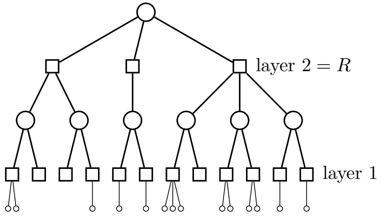
$$p_1 = \Pr[\square\text{-node has} \geq 1 \text{ child}]$$
$$= \Pr_{Y \sim Pois(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$
$$p_i = \Pr[\text{layer } i\ \square\text{-node } v \text{ has} \geq 1 \textit{ surviving} \text{ child}]$$
$$= \Pr_{X \sim Pois(3\alpha p_{i-1}^2)}[X > 0] = 1 - e^{-3\alpha p_{i-1}^2}.$$

$Y :=$ number of (initial) children of v
$X :=$ number of surviving children of v
each child $\bigcirc$-node survives if both its $\square$-children from layer $i-1$ survive $\rightsquigarrow$ probability p_{i-1}^2.
$\Rightarrow Y \sim Pois(3\alpha)$ and $X \sim Bin(Y, p_{i-1}^2)$.
$\Rightarrow X \sim Pois(3\alpha p_{i-1}^2)$. $\rightsquigarrow$ exercise!

What does peeling mean in this setting? (3)

Peeling T_α^R bottom up

for $i = 1$ **to** R **do** // $\square$-layers bottom to top
  for *each* $\square$*-node* v *in layer* i **do**
    if v *has no children* **then**
      remove v and its parent $\bigcirc$



Survival probabilities $p_i := \Pr[\blacksquare\text{-node in layer } i \text{ is } \textit{not} \text{ peeled}]$

$$p_1 = \Pr[\square\text{-node has} \geq 1 \text{ child}]$$
$$= \Pr_{Y \sim Pois(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$
$$p_i = \Pr[\text{layer } i\ \square\text{-node } v \text{ has} \geq 1 \textit{ surviving} \text{ child}]$$
$$= \Pr_{X \sim Pois(3\alpha p_{i-1}^2)}[X > 0] = 1 - e^{-3\alpha p_{i-1}^2}.$$

$\square$-survival probabilities. With $p_0 := 1$ we have

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

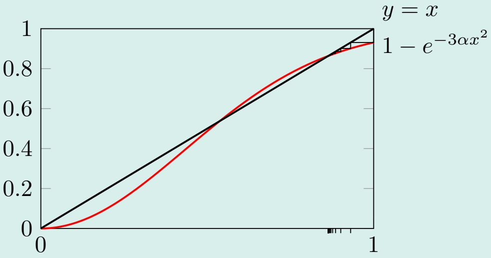
Moreover: $q_R := \Pr[\text{root survives}] = p_R^3$.

What role does $c_3^\Delta \approx 0.81$ play?

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

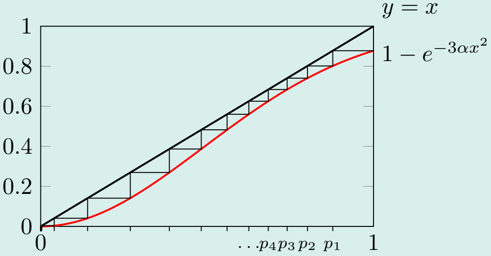
$\hookrightarrow$ consider $f(x) = 1 - e^{-3\alpha x^2}$

Case 1: $\exists x > 0 : f(x) = x$.



$$\Rightarrow \lim_{i \to \infty} p_i = x^* = \max\{x \in [0,1] \mid f(x) = x\}.$$

Case 2: $\forall x \in (0,1] : f(x) < x$



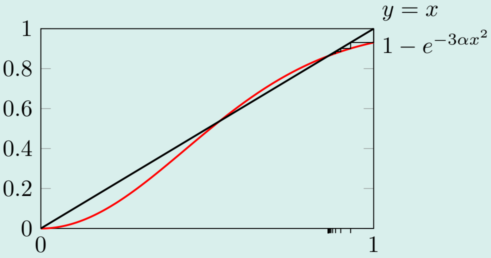
$$\Rightarrow \lim_{i \to \infty} p_i = 0.$$

iv What role does $c_3^\Delta \approx 0.81$ play?

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \dots \end{cases}$$

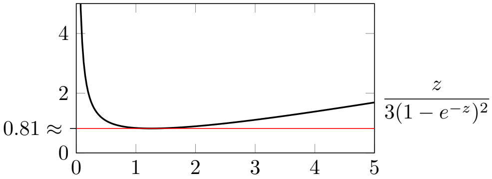
$\hookrightarrow$ consider $f(x) = 1 - e^{-3\alpha x^2}$

Case 1: $\exists x > 0 : f(x) = x$.



$$\Rightarrow \lim_{i\to\infty} p_i = x^* = \max\{x \in [0,1] \mid f(x) = x\}.$$

$$\begin{aligned} \text{Case 1} &\Leftrightarrow \exists x > 0 : x = 1 - e^{-3\alpha x^2} \\ &\Leftrightarrow \exists x > 0 : x^2 = (1 - e^{-3\alpha x^2})^2 \\ &\Leftrightarrow \exists z > 0 : \frac{z}{3\alpha} = (1 - e^{-z})^2 \quad // \; z = 3\alpha x^2 \\ &\Leftrightarrow \exists z > 0 : \alpha = \frac{z}{3(1 - e^{-z})^2} \\ &\Leftrightarrow \alpha \geq \min_{z>0} \frac{z}{3(1 - e^{-z})^2} =: c_3^\Delta \approx 0.81 \end{aligned}$$



iv Interim Conclusion: What we learned about peeling T_α

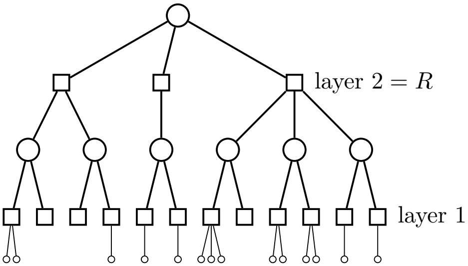
Karlsruhe Institute of Technology

Lemma

For $\alpha < c_3^\Delta \approx 0.81$ we have

- $\lim_{i\to\infty} p_i = 0.$
- $\lim_{R\to\infty} q_R = \lim_{R\to\infty} p_R^3 = 0.$

"Root rarely survives for large R."



What does it mean for T_α to be locally like $G_{m,\alpha m}$?

Neighbourhoods in T_α and G

Let $R \in \mathbb{N}$. We consider

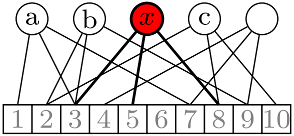
- T_α^R as before and
- for any fixed $x \in S$ the subgraph $G_{m,\alpha m}^{x,R}$ of $G_{m,\alpha m}$ induced by all nodes with distance at most $2R$ from x.

Lemma

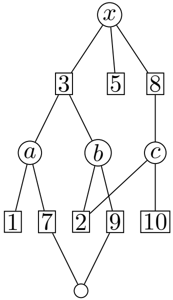
For any $R \in \mathbb{N}$, the **distribution** of $G_{m,\alpha m}^{x,R}$ converges the distribution of T_α^R, i.e.

$$\forall T : \lim_{m \to \infty} \Pr[G_{m,\alpha m}^{x,R} = T] = \Pr[T_\alpha^R = T].$$

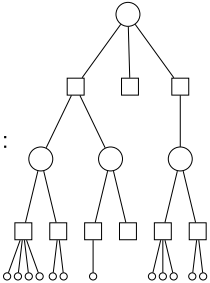
$G_{m,\alpha m}$:



$G_{m,\alpha m}^{x,R}$:



T_α^R :



Distribution of T_α^R

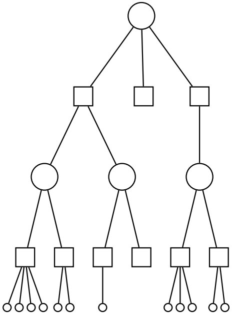
e.g. for $y = (2, 0, 1, 4, 2, 1, 0, 3, 2)$:

Lemma

Let T_y be a possible outcome of T_α^R given by a finite sequence $y = (y_1, \ldots, y_k) \in \mathbb{N}_0^k$ specifying the number of children of $\square$-nodes in level order. Then

$$\Pr[T_\alpha^R = T_y] = \prod_{i=1}^{k} \Pr_{Y \sim Pois(3\alpha)}[Y = y_i].$$

$T_y =$



No cycles in $G^{x,R}_{m,\alpha m}$

Karlsruhe Institute of Technology

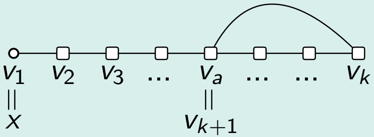
Lemma

Assume $R = \mathcal{O}(1)$. The probability that $G^{x,R}_{m,\alpha m}$ contains a cycle is $\mathcal{O}(1/m)$.

Proof.

If $G^{x,R}_{m,\alpha m}$ contains a cycle then we have

- a sequence $(v_1 = x, v_2, \dots, v_k, v_{k+1} = v_a)$ of nodes with $a \in [k]$
- of length $k \le 4R$ (consider BFS tree for x and additional edge in it)
- for each $i \in \{1, \dots, k\}$ an index $j_i \in \{1, 2, 3\}$ of the hash function connecting v_i and v_{i+1}. (If $a = k - 1$ then $j_k \neq j_{k-1}$.)



$\Pr[\exists \text{cycle in } G^{x,R}_{m,\alpha m}] \le \Pr[\exists 2 \le k \le 4R : \exists v_2, \dots, v_k : \exists a \in [k] : \exists j_1, \dots, j_k \in [3] : \forall i \in [k] : h_{j_i} \text{ connects } v_i \text{ to } v_{i+1}]$

$$\le \sum_{k=2}^{4R} \sum_{v_2,\dots,v_k} \sum_{a=1}^{k} \sum_{j_1,\dots,j_k} \prod_{i=1}^{k} \Pr[h_{j_i} \text{ connects } v_i \text{ to } v_{i+1}] \le \sum_{k=2}^{4R} (\max\{m, n\})^{k-1} \cdot k \cdot 3^k \left(\tfrac{1}{m}\right)^k = \tfrac{1}{m} \sum_{k=2}^{4R} k \cdot 3^k = \mathcal{O}(1/m). \qquad \square$$

v Distribution of $G^{x,R}_{m,\alpha m}$

Lemma

Let T_y be a possible outcome of T^R_α as before. Then

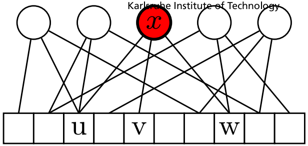
$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

"Proof by example", using T_y shown on the right.

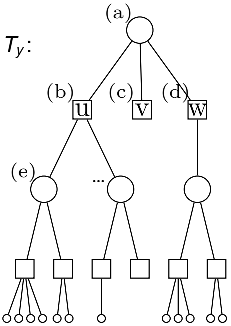
The following things have to "go right" for $G^{x,R}_{m,\alpha m} = T_y$.

- a $h_1(x), h_2(x), h_3(x)$ pairwise distinct: probability $\xrightarrow{m\to\infty} 1$
  $\hookrightarrow$ non-distinct would give cycle of length 2. Unlikely by lemma.

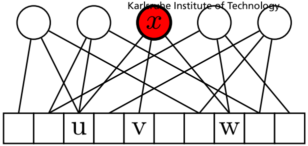
Note: $3\lfloor \alpha m \rfloor - 3$ remaining hash values $\sim \mathcal{U}([m])$.



T_y:



v Distribution of $G^{x,R}_{m,\alpha m}$



Lemma

Let T_y be a possible outcome of T^R_α as before. Then

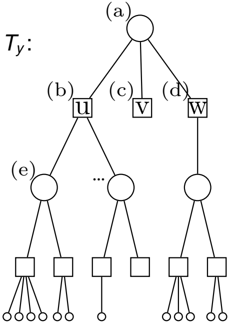
$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

“Proof by example”, using T_y shown on the right.

- b Exactly $y_1 = 2$ of the remaining hash values are u.
  $\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-3,\frac{1}{m})}[Y = 2] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y = 2]$. $\to$ exercise
  Moreover: The two hash values must belong to 2 distinct keys. Probability $\xrightarrow{m\to\infty} 1$.
  $\hookrightarrow$ non-distinct would give cycle of length 2.

Note: The $3\lfloor\alpha m\rfloor - 5$ remaining hash values are $\sim \mathcal{U}([m]\setminus\{u\})$. $\to$ exercise

T_y:



v Distribution of $G_{m,\alpha m}^{x,R}$

Lemma

Let T_y be a possible outcome of T_α^R as before. Then

$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G_{m,\alpha m}^{x,R} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

"Proof by example", using T_y shown on the right.

- c None of the remaining hash values are v.

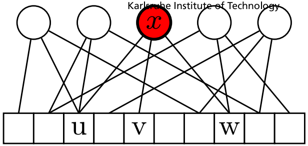
  $\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-5,\frac{1}{m-1})}[Y=0] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y=0]$.

Note: The $3\lfloor\alpha m\rfloor - 5$ remaining hash values are $\sim\mathcal{U}([m]\setminus\{u,v\})$.

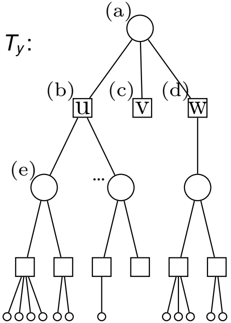
- d One of the remaining hash values is w.

  $\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-5,\frac{1}{m-2})}[Y=1] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y=1]$.

...



T_y:

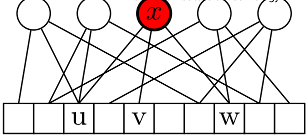


Distribution of $G^{x,R}_{m,\alpha m}$

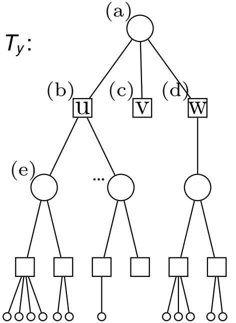
Lemma

Let T_y be a possible outcome of T^R_α as before. Then

$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$



T_y:



Proof sketch in general (some details ommitted)

- General case at i-th $\square$-node. Want: probability that $G^{x,R}_{m,\alpha m}$ continues to match T_y. Note: T_y is fixed, so i and the number c_i of previously revealed hash values is bounded.

  $$\Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor - c_i, \frac{1}{m-i+1})}[Y = y_i] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

  Moreover, those y_i hash values must belong to distinct fresh keys. Probability $\xrightarrow{m\to\infty}$ 1
  $\hookrightarrow$ otherwise we'd have a cycle.

- General case for $\bigcirc$-node. The two children must be fresh: probability $\xrightarrow{m\to\infty}$ 1
  $\hookrightarrow$ otherwise there would be a cycle.

vi Probability that a specific key survives peeling

Karlsruhe Institute of Technology

Lemma

Let $\alpha < c_3^\Delta$. *Let* x *be any* $\bigcirc$*-node in* $G_{m,\alpha m}$ *as before (chosen before sampling the hash functions). Let*

$$\mu_m := \Pr_{h_1,h_2,h_3 \sim \mathcal{U}([m]^D)}[x \text{ is removed when peeling } G_{m,\alpha m}].$$

Then $\lim_{m\to\infty} \mu_m = 1$.

vi $\mu_m := \Pr[x \text{ is removed when peeling } G_{m,\alpha m}] \xrightarrow{m\to\infty} 1$

Karlsruhe Institute of Technology

Let $\delta > 0$ be arbitrary. We will show $\lim_{m\to\infty} \mu_m \geq 1 - 2\delta$.

Let $R \in \mathbb{N}$ be such that $q_R < \delta$. — possible because $\lim_{R\to\infty} q_R = 0$

$\mathcal{Y}^R := \{\text{all possibilities for } T_\alpha^R\}$

$\mathcal{Y}^R_{\text{peel}} := \{T \in \mathcal{Y}^R \mid \text{ peeling } T \text{ removes the root}\}$ — note: $\Pr[T_\alpha^R \notin \mathcal{Y}^R_{\text{peel}}] = q_R \leq \delta$.

Let $\mathcal{Y}^R_{\text{fin}} \subseteq \mathcal{Y}^R$ be a *finite* set such that $\Pr[T_\alpha^R \notin \mathcal{Y}^R_{\text{fin}}] \leq \delta$ — uses that $\mathcal{Y}^R$ is countable and $\sum_{T\in\mathcal{Y}^R} \Pr[T_\alpha^R = T] = 1$.

$$\lim_{m\to\infty} \mu_m \geq \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} \in \mathcal{Y}^R_{\text{peel}}]$$
peeling only in R-neighbourhood of x is "weaker"

$$\geq \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} \in \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}]$$

$$= \lim_{m\to\infty} \textstyle\sum_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \Pr[G^{x,R}_{m,\alpha m} = T]$$

$$= \textstyle\sum_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} = T]$$
finite sums commute with limit

$$= \textstyle\sum_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \Pr[T_\alpha^R = T]$$
previous lemmas

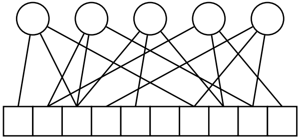
$$= \Pr[T_\alpha^R \in \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}] = 1 - \Pr[T_\alpha^R \notin \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}]$$

$$= 1 - \Pr[T_\alpha^R \notin \mathcal{Y}^R_{\text{peel}} \vee T_\alpha^R \notin \mathcal{Y}^R_{\text{fin}}]$$
De Morgan's laws: $\overline{A \cap B} = \overline{A} \cup \overline{B}$

$$\geq 1 - \Pr[T_\alpha^R \notin \mathcal{Y}^R_{\text{peel}}] - \Pr[T_\alpha^R \notin \mathcal{Y}^R_{\text{fin}}] \geq 1 - 2\delta.$$
union bound: $\Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_2]$ $\square$

Proof of the Peeling Theorem

Karlsruhe Institute of Technology



Theorem

Let $\alpha < c_3^\Delta$. Then

$$\Pr[G_{m,\alpha m} \text{ is peelable}] = 1 - o(1).$$

Proof

Let $n = \lfloor \alpha m \rfloor$ and $0 \le s \le n$ the number of ◯ nodes surviving peeling.

last lemma:	each ◯ survives with probability $o(1)$.
linearity of expectation	$\mathbb{E}[s] = n \cdot o(1) = o(n)$.
Exercise:	$\Pr[s \in \{1, \dots, \delta n\}] = \mathcal{O}(1/m)$ if $\delta > 0$ is a small enough constant.
Markov:	$\Pr[s > \delta n] \le \frac{\mathbb{E}[s]}{\delta n} = \frac{o(n)}{\delta n} = o(1)$.
finally:	$\Pr[s > 0] = \Pr[s \in \{1, \dots, \delta n\}] + \Pr[s > \delta n] = \mathcal{O}(1/m) + o(1) = o(1)$. □

Conclusion

Peeling Process

- greedy algorithm for placing keys in cuckoo table
- works up to a load factor of $c_3^\Delta \approx 0.81$

We saw glimpses of important techniques

- *Local interactions in large graphs.* Also used in statistical physics.
- *Galton-Watson Processes / Trees*. Random processes related to T_α.
- *Local weak convergence*. How the finite graph $G_{m,\alpha m}$ is locally like T_α.

But wait, there's more!

- Further applications of peeling
  - retrieval data structures (next lecture)
  - perfect hash functions (next lecture)
  - set sketches
  - linear error correcting codes

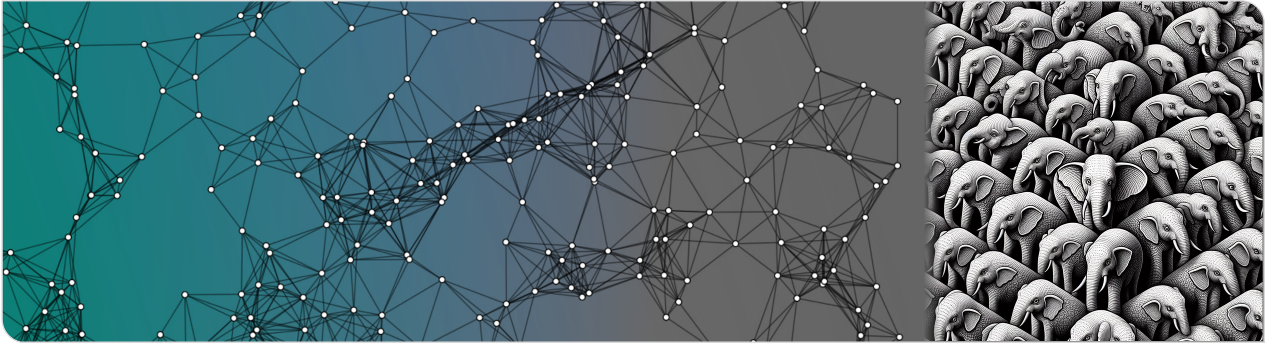
Anhang: Mögliche Prüfungsfragen I

- Cuckoo Hashing und der Schälalgorithmus
  - (Wie) kann man Cuckoo Hashing mit mehr als 2 Hashfunktionen aufziehen?
  - Welcher Vorteil ergibt sich im Vergleich zu 2 Hashfunktionen?
  - Wie funktioniert der Schälalgorithmus zur Platzierung von Schlüsseln in einer Cuckoo Hashtabelle?
  - Schälen lässt sich als einfacher Prozess auf Graphen auffassen. Wie?
  - Was besagt das Hauptresultat, das wir zum Schälprozess bewiesen haben?
- Beweis des Schälsatzes. *Mir ist klar, dass der Beweis äußerst kompliziert ist.*
  - Im Beweis haben zwei Graphen eine Rolle gespielt ein endlicher und ein (potentiell) unendlicher. Wie waren diese Graphen definiert?
  - Welcher Zusammenhang besteht zwischen der Verteilung der Knotengrade in T_α und $G_{m,\alpha m}$?

Karlsruhe Institute of Technology

Probability and Computing – Retrieval and Perfect Hashing

Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Content

1. **Retrieval Data Structures**
   - The Retrieval Problem
   - Motivation
   - Construction Using Peeling

2. **(Minimal-) Perfect Hashing**
   - The Perfect Hashing Problem
   - Motivation: Updatable Retrieval
   - Construction using Trial and Error
   - Construction using Cuckoo Hashing and Retrieval

Notational heads-up

- in other chapters $[k] := \{1, \ldots, k\}$
- in this chapter *sometimes* $[k] := \{0, \ldots, k-1\}$
- you'll figure it out. . .

The Retrieval Problem

Karlsruhe Institute of Technology

The retrieval data type (for universe D, range $[k]$)

construct(f):

input: function $f : S \to [k]$ $//f \subseteq D \times [k]$
where $S \subseteq D$ has size $n = |S|$

output: data structure R.

eval(R, x):

input: $R = \textbf{construct}(f : S \to [k])$, $x \in D$

output: some value in $[k]$

requirement: $\textbf{eval}(R, x) = f(x)$ for all $x \in S$

The price to pay

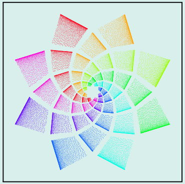
- R cannot be used to decide “is $x \in S$?”
- **eval**(R, x) is *unspecified* if $x \notin S$.

Goals

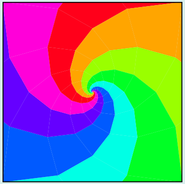
- space requirement of R is $\mathcal{O}(n \log k)$ bits
  - possibly even $n\lceil \log_2(k) \rceil + o(n)$
  - ⚠ naively storing f needs $\Omega(n(\log(k) + \log(|D|)))$
- ideally running time of **eval** is $\mathcal{O}(1)$
- ideally running time of **construct** is $\mathcal{O}(n)$

Intuition

f :



R :

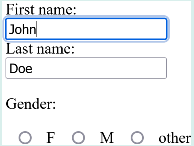


- R is a *continuation* of f
- information about the domain S is lost.

Motivation for Retrieval

Karlsruhe Institute of Technology

Task: Predict gender based on first name



- want $\geq 90\%$ accuracy
- client side only
- lightweight

Have large data base:

Annotated list of 10000 most common first names.

$$f : \{\text{Dave} \mapsto M, \text{Joanna} \mapsto F, \text{Christina} \mapsto F, \ldots\}$$

$\approx$ 10 bytes per name, too large to send to client.

Solution using retrieval

- send $R = \textbf{construct}(f)$ to client
  $\hookrightarrow \approx$ 1 bit per name
- prefill gender with $\textbf{eval}(R, \text{firstName})$

Weaknesses:

May guess incorrectly if

- name is ambiguous (“Kim”, “Chris”)
- user is non-binary / prefers not to say
- name not listed in f (e.g. “Crhistina”, “Inghean”)
  $\hookrightarrow$ would be better to *refrain from guessing*

Content

1. **Retrieval Data Structures**
   - The Retrieval Problem
   - Motivation
   - Construction Using Peeling

2. **(Minimal-) Perfect Hashing**
   - The Perfect Hashing Problem
   - Motivation: Updatable Retrieval
   - Construction using Trial and Error
   - Construction using Cuckoo Hashing and Retrieval

Peeling $\to$ Cuckoo-Style Retrieval

Retrieval Data Structure $R = (h_1, h_2, h_3, A)$

- $m = \frac{n}{0.81} = 1.23n$ //0.81 is peeling threshold $c_3^{\triangle}$
- $A \in [k]^m$ is array of cleverly chosen values
- $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ //SUHA
- **eval**$(R, x) := (A[h_1(x)] + A[h_2(x)] + A[h_3(x)]) \bmod k$

Performance

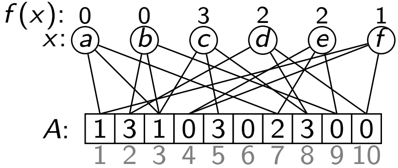
- space $1.23n\lceil \log_2(k) \rceil$ bits
- construct in $\mathcal{O}(n)$
- eval in $\mathcal{O}(1)$

How does **construct**(f) choose A?

If $A[j]$ is only used by x_i then setting $A[j]$ *in the end* takes care of x_i without affecting other keys.

$\hookrightarrow$ can forget about x_i “for now” and focus on the rest

$\hookrightarrow$ if configuration is peelable, this takes care of all keys



Equations (mod k for $k = 4$)

c: $A[5] := 3 - A[3] - A[8]$

d: $A[8] := 2 - A[2] - A[10]$

b: $A[2] := 0 - A[3] - A[9]$

a: $A[3] := 0 - A[1] - A[7]$

e: $A[7] := 2 - A[4] - A[9]$

f: $A[1] := 1 - A[4] - A[10]$

Content

The Perfect Hashing Problem

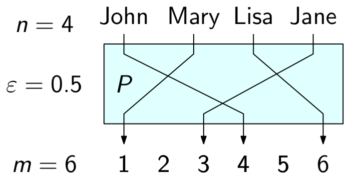
Perfect hashing data type (for universe D, $\varepsilon \geq 0$)

construct(S):
- input: $S \subseteq D$ of size $n = |S|$
- output: data structure P.

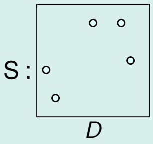
eval(P, x):
- input: $P =$ **construct**(S) and $x \in D$
- output: a number in $[m]$ where $m = (1 + \varepsilon)n$
- requirement: $x \mapsto$ **eval**(P, x) is injective on S

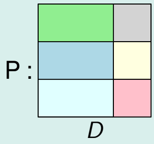
Goals

- ε is small // $\varepsilon = 0$: *Minimal* perfect hashing
- space requirement of P is $\mathcal{O}(n)$ bits
  - $\approx 1.44n$ bits is necessary and sufficient for $\varepsilon = 0$
  - note: storing S might need $\Omega(n \log(|D|))$ bits.
- ideally: running time of **eval** is $\mathcal{O}(1)$
- ideally: running time of **construct** is $\mathcal{O}(n)$



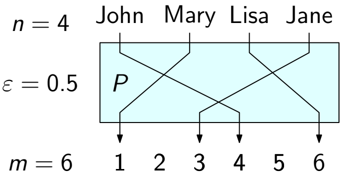
Intuition





- P is partition of D that separates S
- details about S are lost.
- note: P is “perfect hash function” but need not be random

Motivation for (Minimum-) Perfect Hashing



Short IDs

Replace keys with short unique identifies

$$\textbf{eval}(P, \text{“}\texttt{CreativeUserName\_WithSugarOnTop}\text{”}) = 10241.$$

Updatable Retrieval: A hash table without keys

- assume we have MPHF P for S
- can store additional data $f(x) \in [k]$ on $x \in S$ in array of length m in position $\textbf{eval}(P, x)$.
  $\hookrightarrow$ array takes $m\lceil \log_2(k) \rceil$ bits

⚠ Weaker than a normal hash table:

- S is static (values updateable)
- trying to access $f(x)$ for $x \notin S$ gives undefined result
- trying to update $f(x)$ for $x \notin S$ destroys information

Content

Brute Force: Perfect Hashing Using Naive Trial and Error

Karlsruhe Institute of Technology

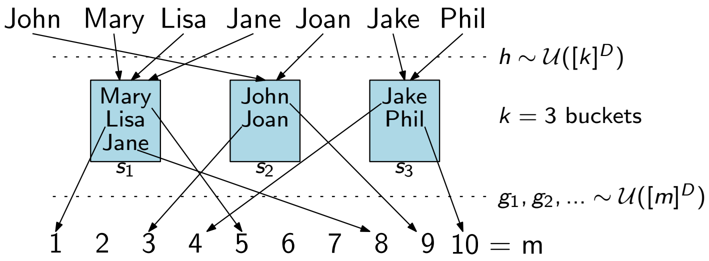
Exercise: What if we played the lottery until we win?

For any $S \subseteq D$ of size n we have $\Pr_{h \sim \mathcal{U}([n]^D)}[h \text{ is injective on } S] = \frac{n!}{n^n}$.

$\hookrightarrow$ Success after trying $\approx \frac{n^n}{n!}$ random hash functions.

$\hookrightarrow$ Need to store seed of $\log_2\left(\frac{n^n}{n!}\right) \approx \log_2(e^n) \approx 1.44n$ bits.

PTHash: Perfect Hashing Using Refined Trial and Error



Perfect Hash Function $P = (k, h, (g_i)_{i \in \mathbb{N}}, (s_1, \ldots, s_k))$

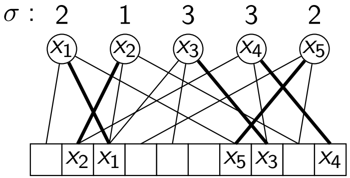
- **eval**$(P, x) := g_{s_{h(x)}}(x)$
- $s_1, \ldots, s_k$ are found using trial and error
- *huge design space*

Cuckoo Hashing + Retrieval $\rightarrow$ Perfect Hashing

Karlsruhe Institute of Technology

Cuckoo Hashing (abstract reminder)

Let $S \subseteq D$ of size $n = |S|$ and $h_1, \ldots, h_k \sim \mathcal{U}([m]^D)$ where $\frac{n}{m} < c_k^*$ for some *threshold* c_k^*.
With high probability there exists $\sigma(x) \in [k]$ for each $x \in S$ such that $x \mapsto h_{\sigma(x)}(x)$ is injective on S.



(picture assumes $h_1(x) < h_2(x) < h_3(x)$)

Perfect Hash Function from Retrieval

- Store $\sigma : S \rightarrow [k]$ as retrieval data structure R
- (non-minimal) PHF $P = (R, h_1, \ldots, h_k)$ with

$$\textbf{eval}(P, x) := h_{\textbf{eval}(R,x)}(x).$$

Example with $k = 4$

- need $\frac{n}{m} < c_4^* \approx 0.9768 \rightsquigarrow \varepsilon \approx 0.0238$
- space needed for P is the space for R: $\approx 1.23n \log_2(k) = 2.26n$ bits with the approach from slide 7

Conclusion

Karlsruhe Institute of Technology

Space efficient data structures for special purposes

- (M)PHF for $S \subseteq D$ realises injective function on S, without storing S.
- retrieval data structure for $f : S \to [k]$ can reproduce $f(x)$ for each $x \in S$, without storing S.

Relationships we saw:

Peeling $\to$ Retrieval using $1.23n\lceil\log_2(k)\rceil$ bits

4-ary Cuckoo Hashing + Retrieval $\to$ Perfect Hashing using $\approx$ 2.26 bits $(\varepsilon = 0.0238)$

Perfect Hashing $\to$ Updatable Retrieval (“hash table without keys”)

Remark: There is more...

- Best constructions are more complicated. Not here...
- Active research @ITI Sanders.

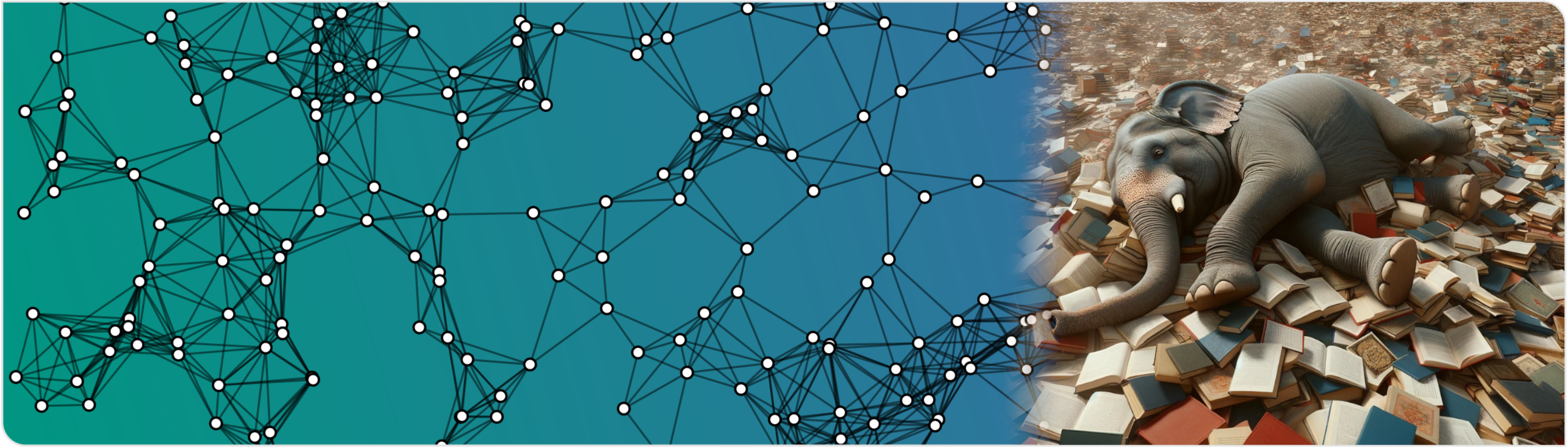
Anhang: Mögliche Prüfungsfragen I

- Retrieval Datenstrukturen
  - Was ist der Funktionsumfang einer Retrieval Datenstruktur?
  - Was sind die Vorteile und Nachteile im Vergleich zu einer normalen Hashtabelle?
  - Welche Anwendungen für Retrieval Datenstrukturen haben wir kennengerlernt?
  - Wie lässt sich eine Retrieval Datenstruktur mithilfe des Schälalgorithmus konstruieren? Was sind Konstruktions- und Zugriffszeiten? Was der Speicherverbrauch?
- Perfekte Hashfunktionen
  - Was zeichnet eine gute Perfekte Hashfunktion aus?
  - Wir haben Hashtabellen ohne Schlüssel kennengelernt. Was hat es damit auf sich?
  - Wie kann man perfekte Hashfunktionen mit Trial und Error konstruieren?
  - Wie kann man perfekte Hashfunktionen mittels Cuckoo Hashing und Retrieval konstruieren? Was ist dabei der Speicherverbrauch?

Karlsruhe Institute of Technology

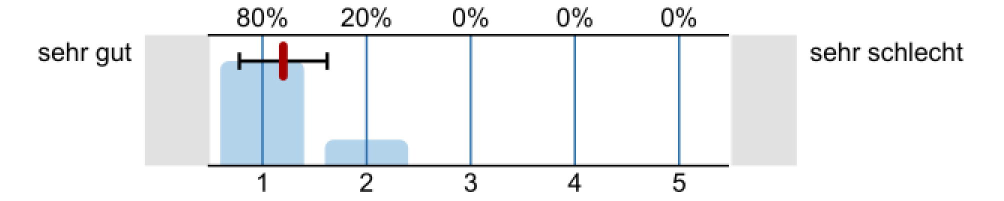
Probability & Computing

Conclusion

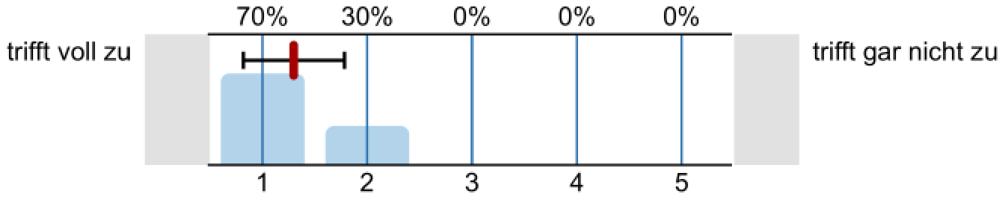


Eval

Bitte benoten Sie die Lehrveranstaltung insgesamt



In dieser Lehrveranstaltung lerne ich viel.



Things to keep

Die Folien sind extraklasse

Elephantenbilder

hinweisen auf aktuelle Forschung

spannende Themen

Things to improve

Ich finde, dass max tendentiell zu schnell die Folien bespricht.

Vor allem bei Max war die Zeit manchmal zu knapp für die Inhalte

sehr viele Mathematische Umformungen

Wall-of-Formeln

Manchmal fehlen den Folien ohne clicks viel Stoff ✓ ?

Übungsblätter jede Woche sind einfach anstrengend ✓ ?

Inverted Classroom Thoughts? (active sessions, videos, etc.)

Exercises Thoughts?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
  - What are Monte Carlo algorithms?
  - What kinds of biases can they have?
  - What is probability amplification?
  - How does probability amplification affect the running time and success probability of an algorithm?
  - How do we deal with the biases during probability amplification?
  - Can you derive the trade-off for different running times and success probabilities?
  - For the problem of minimum cuts we saw Karger's algorithm
    - How does the algorithm work? How was this approach motivated?
    - How did the Karger-Stein approach improve over it? How was this adjustment motivated?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms

- What is a randomized approximation algorithm (for a counting problem)?
- We considered the counting problem $\#B$ for Boolean formulas. Did we generally succeed? Why not?
- Which special case did we consider then? Why did we not run into the same issue?
- We saw an algorithm that, given sets $S \subseteq D$ approximates the size $|S|$
  - Under which assumptions is it applicable?
  - How does the algorithm work?
  - How does the number of required samples depend on $|S|$ and $|D|$?
- To approximate $\#B$ for a DNF B we considered a more sophisticated approach
  - How does it work?
  - How does it avoid the problem of the naive approach?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms

- Definition of streaming algorithms
  - What is the task of a streaming algorithm (with respect to a value $F = F(a_1, \ldots, a_m)$)?
  - What is the specific challenge that streaming algoithms face?

- Streaming algorithms for $F_1 = m$
  - For what application may we need an estimate of F_1?
  - How much memory is needed when simply counting? Can a deterministic method do something better?
  - How does the LossyCounting algorithm work? Why is that not useful?
  - How does Morris' algorithm work?
    - Can you prove that it is an unbiased estimator?*
    - Can you prove that the required space is doubly-logarithmic in m?
    - What is its weakness and how did we fix it?

- Streaming algorithms for $F_0 = \{a_1, \ldots, a_m\}$
  - For what application may we need an estimate of F_0?
  - How much memory does the naive deterministic algorithm need? What can we reach with CVM?
  - In an intermediate step we considered the LossyStore algorithm/ How does it work?
  - How does the CVM algorithm work? What the connection to the LossyStore algorithm?
  - During the analysis of the failure probability of CVM we distinguished between two kinds of problems. Which ones?*

What have we learned?

Karlsruhe Institute of Technology

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
  - What would an ideal hash function be like? How would that be useful? What is the problem with this ideal version?
  - What is the Simple Uniform Hashing Assumption (SUHA)? Why should we use it? What are alternatives?
  - How is a cryptographic pseudorandom function with certain guarantees with respect to being indistinguishable from actuall random functions useful for us? How is that connected to SUHA?*
  - Universal hashing
    - What is a c-universal hash family?
    - Which classes of c-universal hash functions did we encounter? How did we prove them to be c-universal?
    - How is d-independence defined for classes o hash functions?
    - Which classes of d-independent hash functions did we encounter?
    - What is the connection between c-universality and d-independence? (exercise)
    - Chernoff bounds are well suited for sums of independent random variables. What can we do if the random variables are only d-independent?*
  - Hash tables with chaining
    - What bound on expected insertion time did we prove? How?
    - Where does the distribution of the hash function come into play?
    - Can you name a property of a class of universal hash functions that is sufficient for the proof to work?
  - Hash tables with linear probing
    - What bound on expected insertion time did we prove? How?
    - Where does the distribution of the hash function come into play?
    - Can you name a property of a class of universal hash functions that is sufficient for the proof to work?
    - How did we use that property?*

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter

- Approximate membership query data structures
  - What is their task?
  - What is the advantage over exact data structures?
  - For what applications may we need one?
- Bloom filter
  - What does it consist of and which operations does it support?
  - How is it parameterized and how are the parameters related?
  - What did our analysis reveal about a good parameter choice? How do we choose the remaining parameters? What is the resulting memory requirement?
- About that analsysis
  - What are the expected numbers of zeroes and ones?
  - How is the false-positive probability related to the number of zeroes and ones?
  - How can we argue that the numbers of zeroes and ones in the bloom filter are close to their expectation?

What have we learned?

Karlsruhe Institute of Technology

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing

- What is cuckoo hashing and what can it do?
  - What is the basic idea? How do operations work?
  - What do we need to consider about table size and load factor?
  - What do we know about the running time of the operations?
  - What are advantages and disadvantages with respect to other techniques like linear probing?
- Analysis
  - A failed insertion corresponds to certain structures in the cuckoo graph. Which ones?
  - How did we show that such structures are unlikely to exist?
  - How did we bound the expected insertion time?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling

- Cuckoo hashing and the peeling algorithm
  - (How) can you extend cuckoo hashing to use more than 2 hash functions?
  - What is the advantage over using 2 functions?
  - How does the peeling algorithm for placing keys in a cuckoo hash table work?
  - Peeling can be seen as a simple process on graphs. How?
  - What does the main result, that we proved about the peeling process, state?
- Proof of the peeling theorem (yes, that's a tough one)
  - In the proof we defined two graphs: a finite one and a (potentially) infinite one. How were they defined?
  - How are the degrees of the nodes in T_α and $G_{m,\alpha m}$ related?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

- Retrieval data structures
  - What operations does a retrieval data structure support?
  - What are advantages and disadvantages compared to a normal hash table?
  - What can retrieval data structures be used for?
  - How can we construct a retrieval data structure using a peeling algorithm? What are construction and retrieval times? What are the memory requirements?
- Perfect hash functions
  - What properties does a good perfect hash function have?
  - We learned about hash tables without keys. What is that about?
  - How can we construct perfect hash functions using trial and error?
  - How can we construct a perfect hash functions using cuckoo hashing and retrieval? What are the memory requirements?

What have we learned?

Karlsruhe Institute of Technology

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes

- Define: What is a PTM? What is the difference to an NTM?
- Define the complexity classes **RP**, co-**RP**, **BPP**, **PP**, **ZPP**
- What is the relevance of the constants $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$ in the definitions? With respect to what are they irrelevant?
- What is the relation between **ZPP** and Las Vegas algorithms? How do the two implications work?
- Which containment relationships are known for the complexity classes?
- For each contianment, can you explain why it is true?
- Are there relationships that we know to be strict? Are there classes that experts believe to be identical?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes
- Random graphs

- What is a random graph model?
  - What do we use them for?
  - What are desirable properties?
- How are Erdős-Rényi random graphs and Gilbert's model defined?
  - How are they related? How do they differ?
  - How do they differ?
- What properties do sparse $G(n, p)$ graphs have? (Degree distribution? Locality?)
- How did we show that the degree of a single vertex in a $G(n, p)$ is approximately Poisson-distributed?
- What are random geometric graphs? What are the degrees of freedom we have when defining them?
- What choices did we make for these degrees of freedom when defining *simple* random geometric graphs?
- How can we compute the expected degree of a node in a simple random geometric graph?
- What are geometric inhomogeneous random graphs? Why are they interesting? How do they compare to (simple) random geometric graphs?
- What do we have to do in order to compute the expected degree of a vertex with a given weight in a GIRG?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes
- Random graphs

Tools

- Coupling

- What is a coupling? What is it used for?
- Can you develop the simpler couplings we considered in the lecture?
- How is the total variation distance defined for the distributions of two random variables?
- What is the coupling inequality?
- What does the Binomial-Poisson approximation state?
  - How was a coupling used in the proof?
- How did we use the Binomial-Poisson approximation to approximate the distribution of a vertex degree in a $G(n, p)$?
  - What random variables did we consider?
  - How were they coupled?
  - What property of the total variation distance did we use there?

What have we learned?

Karlsruhe Institute of Technology

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes
- Random graphs

Tools

- Coupling
- Concentration

- What is concentration? Why are we interested in it?
- What is Markov's inequality? When can it be applied? Can you prove its correctness? In what sense is it tight?
- What is Chebychev's inequality? When can it be applied? Can you prove its correctness?
- What is a (raw/centered) moment?
- In what sense can moments be used to characterize the shape of a distribution?
- What is a moment generating function? What does it have to do with moments?
- What are Chernoff bounds? Can you prove their correctness? How can we use them for specific probability distributions?
- What is the method of bounded differences? When can it be applied / yield useful bounds?
- What is the method of typical bounded differences?*
- How do the different concentration inequalities compare? Are some stronger than othres?
- Given a random variable, can you decide which concentration inequalities can be applied and which cannot?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes
- Random graphs

Tools

- Coupling
- Concentration
- Probabilistic method

- What is the probabilistic method? What is the basic idea?
  - What is the basic idea?
  - What are the two steps that we typically followed when applying the probabilistic method?
- What is the expectation argument?
  - Can you prove its correctness?
- Can you develop the simpler applications of the probabilistic method from the lecture?
- What is the Lovász Local Lemma?
  - About what kind of dependencies does it make a statement?
  - How does it relate to the probabilistic method?

What have we learned?

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes
- Random graphs

Tools

- Coupling
- Concentration
- Probabilistic method
- Continuous probability spaces

- How are probabilties measured in continuous spaces?
- What is a probability density function?
- How does working in continuous probability spaces differ from the discrete case?
- What is the memorylessness of the exponential distribution?
  - Can you derive it formally?
- What is a Poisson process?
  - What is its connection to the uniform/exponential distribution?
- How is independence defined for continuous random variables?
  - In that regard, what are joint / marginal / cumulative density functions?
- What is the Pareto distribution?
  - How can we determine for which parameter choices it has (in)finite expectation and variance?

What have we learned?

Karlsruhe Institute of Technology

Randomized Algorithms & Data Structures

- Probability amplification
- Approximation algorithms
- Streaming algorithms
- Classic hash tables
- Bloom filter
- Cuckoo hashing
- Peeling
- Retrieval and perfect hashing

Things to Analyze

- Complexity classes
- Random graphs

Tools

- Coupling
- Concentration
- Probabilistic method
- Continuous probability spaces
- Yao's principle

- Application to $\bar{\wedge}$-trees?
  - What was our goal when evaluating $\bar{\wedge}$-trees? (minimize query complexity)
  - What worst-case costs can we achieve with a deterministic approach?
  - Can a randomized algorithm do better? How?
  - One can relatively easily see that the randomized complexity is $\Omega(\sqrt{n})$. How?
  - We have also seen a tighter analysis. What components was it made of? In particular: How is Yao's principle applied there?
  - What does Tarsi's theorem state?
- Ski-rental problem
  - Define the problem
  - How do you call this kind of problem? (*online* problem)
  - Is this only relevant to winter sports? (Only key points)
  - What is the competitive ratio?
  - What is the best deterministic algorithm? Why?
  - Is there a randomized algorithm that can beat Break Even? (idea only)
  - Define Yao's principle for online algorithms
  - Which input distribution did we use for the lower bound in Ski-rental? What is the intuition?
  - What are the costs for the online and offline algorithms when using this input distribution? What can we say about the corresponding competitive ratio?

Where to go from here?

Exam

- Get your appointment by mailing Isabelle
- Prepare for the exam, reach out via Discord or mail if you have questions
- Max, Stefan, and Thomas will be in the room with you

Other courses

- Fortgeschrittenes algorithmisches Programmieren
- Algorithm Engineering
- Fortgeschrittene Datenstrukturen
- Parallele Algorithmen
- Text-Indexierung
- Modelle der Parallelverarbeitung
- Algorithmische Geometrie
- Parametrisierte Algorithmen
- Algorithmen für Routenplanung
- Algorithmische Graphentheorie
- Algorithmen zur Visualisierung von Graphen

Master thesis

- Reach out!

Contents