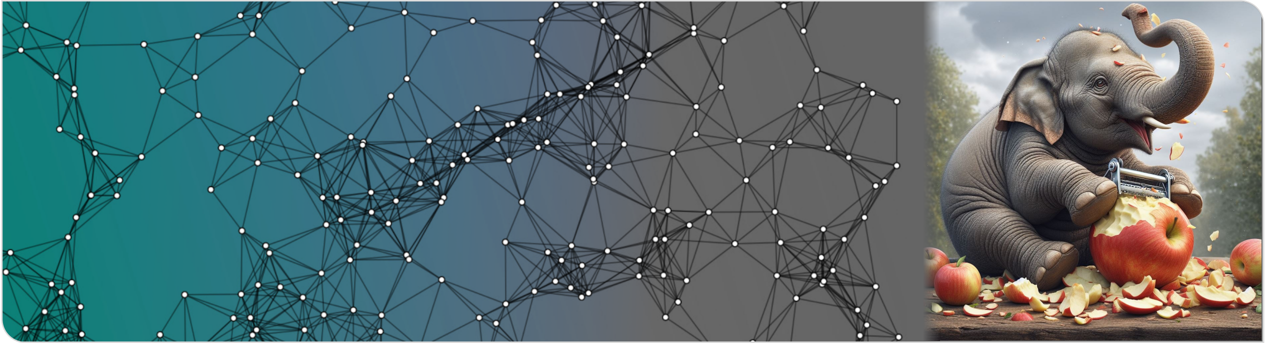
Karlsruhe Institute of Technology

Probability and Computing – The Peeling Algorithm

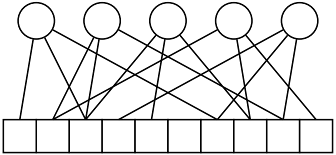
Stefan Walzer, Maximilian Katzmann | WS 2023/2024



Content

Content

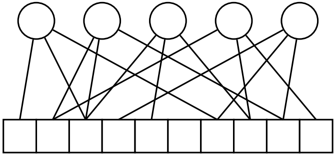
Cuckoo Hashing with one table and k hash functions



$$\begin{aligned} n &\in \mathbb{N} && \text{keys} \\ m &\in \mathbb{N} && \text{table size} \\ \alpha &= \tfrac{n}{m} && \text{load factor} \\ h_1, \ldots, h_k &\sim \mathcal{U}([m]^D) && \text{hash functions} \end{aligned}$$

$\hookrightarrow$ Could also use a separate table per hash function.

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.

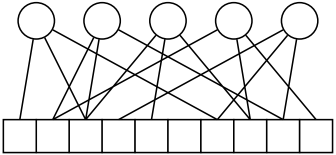
randomWalkInsert(x)

while $x \neq \perp$ **do** // TODO: limit
  sample $i \sim \mathcal{U}([k])$
  swap($x, T[h_i(x)]$)

(some improvements possible)

Cuckoo Hashing with one table and *k* hash functions

Karlsruhe Institute of Technology



$n \in \mathbb{N}$	keys
$m \in \mathbb{N}$	table size
$\alpha = \frac{n}{m}$	load factor
$h_1, \dots, h_k \sim \mathcal{U}([m]^D)$	hash functions

$\hookrightarrow$ Could also use a separate table per hash function.

randomWalkInsert(x)

while $x \neq \bot$ **do** // TODO: limit
- sample $i \sim \mathcal{U}([k])$
- swap($x, T[h_i(x)]$)

(some improvements possible)

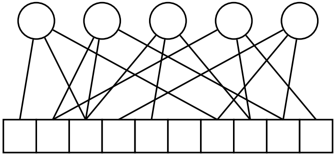
Theorem (without proof)

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 $1 - \mathcal{O}(\frac{1}{m})$.

$c_2^* = \frac{1}{2}, \quad c_3^* \approx 0.92, \quad c_4^* \approx 0.98, \dots$

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
$h_1, \dots, h_k \sim \mathcal{U}([m]^D)$	hash functions

$\hookrightarrow$ Could also use a separate table per hash function.

randomWalkInsert(x)

while $x \neq \bot$ **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 $1 - \mathcal{O}(\frac{1}{m})$.

$c_2^* = \frac{1}{2}, \quad c_3^* \approx 0.92, \quad c_4^* \approx 0.98, \dots$

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!

Static Hash Tables

Karlsruhe Institute of Technology

Static Hash Table

`construct`(S): builds table T with key set S
`lookup`(x): checks if x is in T or not
↪ 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

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
↪ 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

Karlsruhe Institute of Technology

constructByPeeling($S \subseteq D$, $h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \dots, \bot]$ // 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

 $T[i] \leftarrow x$

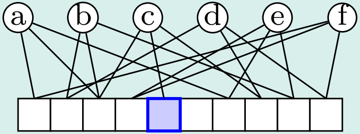
 $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

 return T

else

 return NOT-PEELABLE



The Peeling Algorithm

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \ldots, \bot]$ // 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

  $T[i] \leftarrow x$

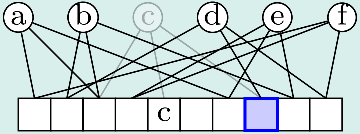
  $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

  return T

else

  return NOT-PEELABLE



The Peeling Algorithm

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \ldots, \bot]$ // 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

  $T[i] \leftarrow x$

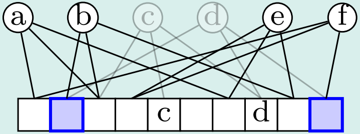
  $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

  return T

else

  return NOT-PEELABLE



The Peeling Algorithm

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \ldots, \bot]$ // 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

  $T[i] \leftarrow x$

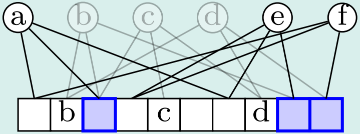
  $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

  return T

else

  return NOT-PEELABLE



The Peeling Algorithm

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \ldots, \bot]$ // 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

$T[i] \leftarrow x$

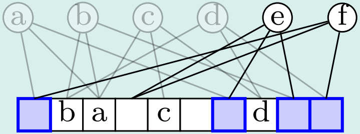
$S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

return T

else

return NOT-PEELABLE



The Peeling Algorithm

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \dots, \bot]$ // 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

  $T[i] \leftarrow x$

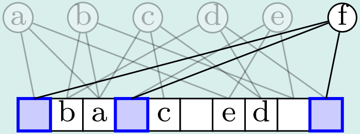
  $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

  return T

else

  return NOT-PEELABLE



The Peeling Algorithm

constructByPeeling$(S \subseteq D,\ h_1, h_2, h_3 \in [m]^D)$

$T \leftarrow [\bot, \dots, \bot]$ // 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

  $T[i] \leftarrow x$

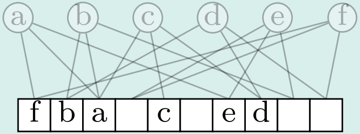
  $S \leftarrow S \setminus \{x\}$

if $S = \varnothing$ **then**

  return T

else

  return NOT-PEELABLE



The Peeling Algorithm

Karlsruhe Institute of Technology

constructByPeeling($S \subseteq D,\ h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \ldots, \bot]$ // 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

  $T[i] \leftarrow x$

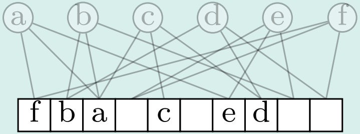
  $S \leftarrow S \setminus \{x\}$

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.

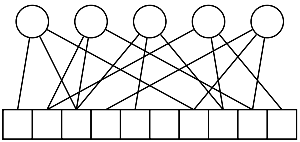
Peelability and the Cuckoo Graph

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 $m \to \infty$.



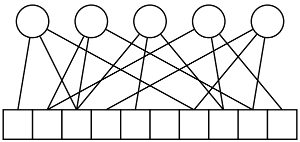
Peelability and the Cuckoo Graph

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

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 $\Pr[G_{m,\alpha m} \text{ is peelable}] = 1 - o(1)$.

Peeling Theorem

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.

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 $G_{m,\alpha m}$.

Peeling Theorem: Proof outline

Theorem (today's goal)

Let $\alpha < c_3^\Delta$. Then $\Pr[G_{m,\alpha m}$ is peelable$] = 1 - o(1)$.

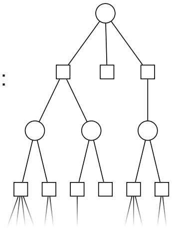
Proof Idea

The random (possibly) infinite tree T_α can be peeled for $\alpha < c_3^\Delta$ and T_α is locally like $G_{m,\alpha m}$.

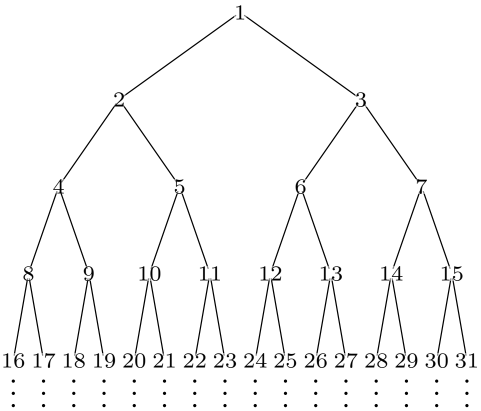
Steps

- 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 :

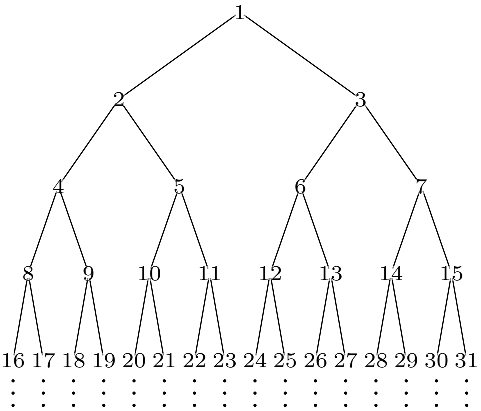


What is an infinite tree in general?



$$V = \mathbb{N}$$
$$E = \{\{n, 2n\} \mid n \in \mathbb{N}\} \cup \{\{n, 2n+1\} \mid n \in \mathbb{N}\}.$$

What is an infinite tree in general?



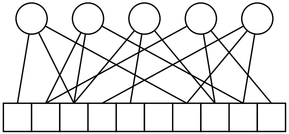
$V = \mathbb{N}$

$E = \{\{n, 2n\} \mid n \in \mathbb{N}\} \cup \{\{n, 2n+1\} \mid n \in \mathbb{N}\}.$

Tree Definitions

- connected and acyclic ✓
  sensible and satisfied
- connected and $|E| = |V| - 1$ ✗
  not sensible

What is T_α in particular?



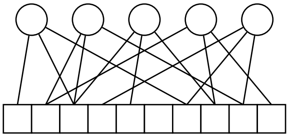
Observations for the finite Graph $G_{m,\alpha m}$

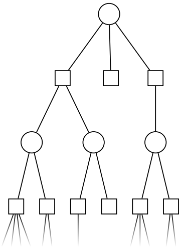
- each ◯ has 3 □ as neighbours (rare exception: $h_1(x), h_2(x), h_3(x)$ not distinct)
- each □ has random number X of ◯ as neighbours with $X \sim Bin(3n, \frac{1}{m}) = Bin(3\lfloor \alpha m \rfloor, \frac{1}{m})$. In an exercise you'll show

$$\Pr[X = i] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y = i].$$

What is T_α in particular?

Karlsruhe Institute of Technology





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.

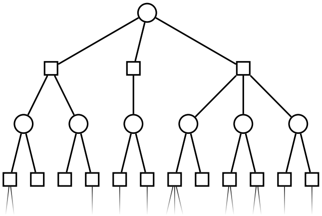
iii What does peeling mean in this setting?

Karlsruhe Institute of Technology

Peeling Algorithm

while $\exists$ □-*node of degree* 1 **do**
  remove it and its incident ○

$\hookrightarrow$ not well defined outcome on T_α!

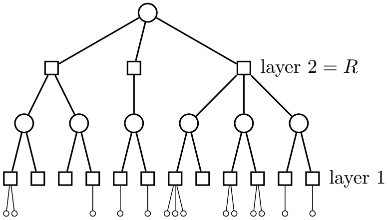


What does peeling mean in this setting?

Peeling Algorithm

while $\exists$ □*-node of degree* 1 **do**
  remove it and its incident ○

$\hookrightarrow$ not well defined outcome on T_α!



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 □-nodes, labeled bottom to top.
- Run peeling on T_α^R (later $R \to \infty$).

$\hookrightarrow$ Why not consider the first $2R$ levels? (without $+1$)

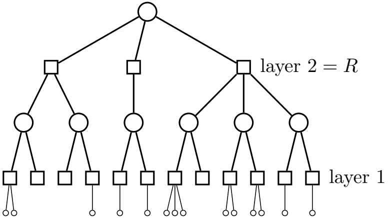
iii What does peeling mean in this setting?

Karlsruhe Institute of Technology

Peeling Algorithm

while $\exists$ □-*node of degree* 1 **do**
  remove it and its incident ○

$\hookrightarrow$ 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 □-nodes, labeled bottom to top.
- Run peeling on T_α^R (later $R \to \infty$).

$\hookrightarrow$ Why not consider the first $2R$ levels? (without $+1$)

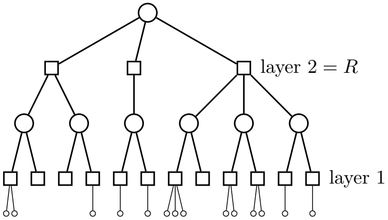
iii What does peeling mean in this setting?

Karlsruhe Institute of Technology

Peeling Algorithm

while $\exists$ □-*node of degree* 1 **do**
  remove it and its incident ○

$\hookrightarrow$ 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 □-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 "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-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

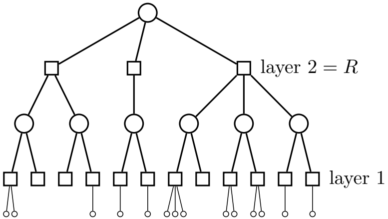
iii What does peeling mean in this setting?

Karlsruhe Institute of Technology

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
  remove it and its incident ○

$\hookrightarrow$ 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 □-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.

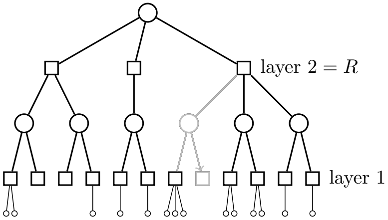
- replace "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-nodes with one child and no parent
  - no matter: such nodes are disconnected from the root anyway
- whether node is peeled only depends on subtree
  $\hookrightarrow$ one bottom up pass suffices for peeling

iii What does peeling mean in this setting?

Peeling Algorithm

while ∃ *childless* □*-node* **do**
  remove it and its incident ○

↪ not well defined outcome on T_α!
↪ 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 □-nodes, labeled bottom to top.
- Run peeling on T_α^R (later $R \to \infty$).

↪ 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 "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-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
  ↪ one bottom up pass suffices for peeling

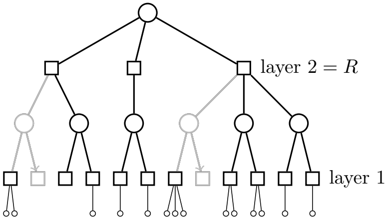
iii What does peeling mean in this setting?

Karlsruhe Institute of Technology

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
  remove it and its incident ○

$\hookrightarrow$ 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 □-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.

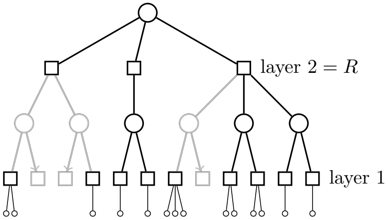
- replace "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-nodes with one child and no parent
  - no matter: such nodes are disconnected from the root anyway
- whether node is peeled only depends on subtree
  $\hookrightarrow$ one bottom up pass suffices for peeling

iii What does peeling mean in this setting?

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
remove it and its incident ○

$\hookrightarrow$ 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 □-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.

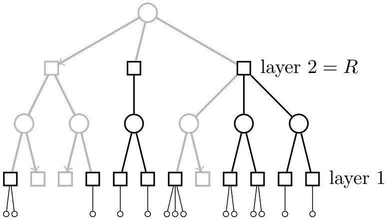
- replace "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-nodes with one child and no parent
  - no matter: such nodes are disconnected from the root anyway
- whether node is peeled only depends on subtree
  $\hookrightarrow$ one bottom up pass suffices for peeling

iii What does peeling mean in this setting?

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
remove it and its incident ○

↪ not well defined outcome on T_α!
↪ 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 □-nodes, labeled bottom to top.
- Run peeling on T_α^R (later $R \to \infty$).

↪ 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 "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-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
  ↪ one bottom up pass suffices for peeling

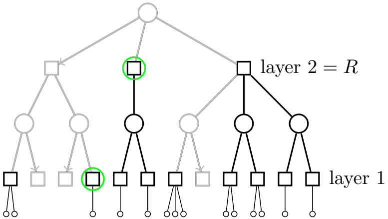
iii What does peeling mean in this setting?

Karlsruhe Institute of Technology

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
  remove it and its incident ○

$\hookrightarrow$ 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 □-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 "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-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

iii What does peeling mean in this setting? (2)

Karlsruhe Institute of Technology

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* □*-node* **do**
  remove it and its incident ○

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* □*-node* **do**
  remove it and its incident ○

Proof.

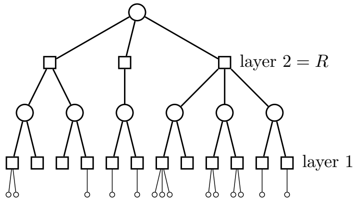
Assume when peeling T_α^R the sequence $\vec{x} = (x_1, \ldots, x_k)$ is a valid sequence of □-node choices. Then $\vec{x}$ is also valid when peeling T_α^{R+1}.

$$\text{peeling } T_\alpha^R \text{ removes the root} \Rightarrow \text{peeling } T_\alpha^{R+1} \text{ removes the root}$$

$$\text{root survives when peeling } T_\alpha^{R+1} \Rightarrow \text{peeling } T_\alpha^R \text{ removes the root}$$

$$q_{R+1} \leq q_R$$

□

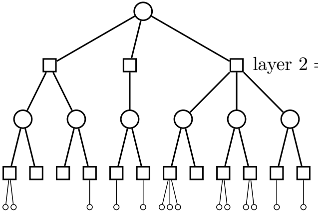


What does peeling mean in this setting? (3)

Karlsruhe Institute of Technology

Peeling T_α^R bottom up

for $i = 1$ **to** R **do** // □-layers bottom to top
  for *each* □*-node* v *in layer* i **do**
    if v *has no children* **then**
      remove v and its parent ○



Survival probabilities $p_i :=$ Pr[□-node in layer i is *not* peeled]

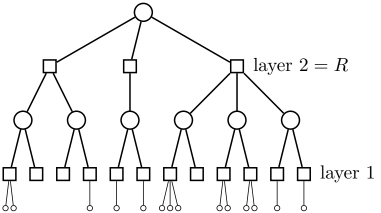
$$p_1 = \Pr[\square\text{-node has} \geq 1 \text{ child}]$$
$$= \Pr_{Y \sim Pois(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$

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** // □-layers bottom to top
  for *each* □*-node v in layer i* **do**
    if *v has no children* **then**
      remove v and its parent ○



Survival probabilities $p_i :=$ Pr[□-node in layer i is *not* peeled]

$$p_1 = \Pr[\square\text{-node has} \geq 1 \text{ child}]$$
$$= \Pr_{Y \sim Pois(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$
$$p_i = \Pr[\text{layer } i \ \square\text{-node } v \text{ has} \geq 1 \textit{ surviving} \text{ child}]$$
$$= \Pr_{X \sim Pois(3\alpha p_{i-1}^2)}[X > 0] = 1 - e^{-3\alpha p_{i-1}^2}.$$

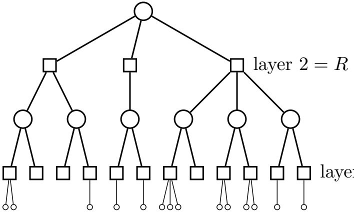
$Y :=$ number of (initial) children of v
$X :=$ number of surviving children of v
each child ○-node survives if both its □-children from layer $i-1$ survive $\rightsquigarrow$ probability p_{i-1}^2.
$\Rightarrow Y \sim Pois(3\alpha)$ and $X \sim Bin(Y, p_{i-1}^2)$.
$\Rightarrow X \sim Pois(3\alpha p_{i-1}^2)$. $\rightsquigarrow$ exercise!

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** // □-layers bottom to top
  for *each* □*-node v in layer i* **do**
    if *v has no children* **then**
      remove v and its parent ○



Survival probabilities $p_i :=$ Pr[□-node in layer i is *not* peeled]

$$p_1 = \Pr[\square\text{-node has} \geq 1 \text{ child}]$$
$$= \Pr_{Y \sim Pois(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$
$$p_i = \Pr[\text{layer } i\ \square\text{-node } v \text{ has} \geq 1 \textit{ surviving} \text{ child}]$$
$$= \Pr_{X \sim Pois(3\alpha p_{i-1}^2)}[X > 0] = 1 - e^{-3\alpha p_{i-1}^2}.$$

□-survival probabilities. With $p_0 := 1$ we have

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

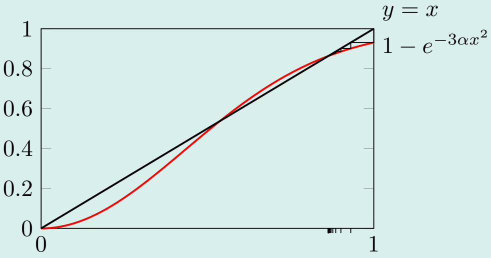
Moreover: $q_R := \Pr[\text{root survives}] = p_R^3$.

iv What role does $c_3^\Delta \approx 0.81$ play?

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \dots \end{cases}$$

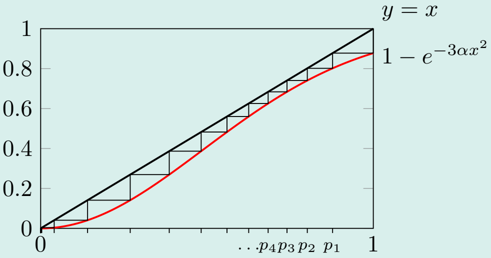
$\hookrightarrow$ consider $f(x) = 1 - e^{-3\alpha x^2}$

Case 1: $\exists x > 0 : f(x) = x$.



$\Rightarrow \lim_{i\to\infty} p_i = x^* = \max\{x \in [0,1] \mid f(x) = x\}.$

Case 2: $\forall x \in (0,1] : f(x) < x$



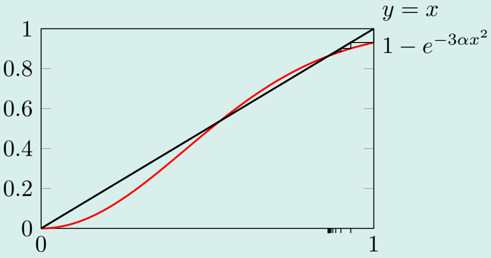
$\Rightarrow \lim_{i\to\infty} p_i = 0.$

iv What role does $c_3^\Delta \approx 0.81$ play?

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

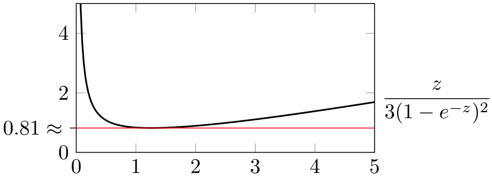
$\hookrightarrow$ consider $f(x) = 1 - e^{-3\alpha x^2}$

Case 1: $\exists x > 0 : f(x) = x$.



$$\Rightarrow \lim_{i\to\infty} p_i = x^* = \max\{x \in [0,1] \mid f(x) = x\}.$$

$$\begin{aligned}
\text{Case 1} &\Leftrightarrow \exists x > 0 : x = 1 - e^{-3\alpha x^2} \\
&\Leftrightarrow \exists x > 0 : x^2 = (1 - e^{-3\alpha x^2})^2 \\
&\Leftrightarrow \exists z > 0 : \frac{z}{3\alpha} = (1 - e^{-z})^2 \quad // \; z = 3\alpha x^2 \\
&\Leftrightarrow \exists z > 0 : \alpha = \frac{z}{3(1 - e^{-z})^2} \\
&\Leftrightarrow \alpha \geq \min_{z>0} \frac{z}{3(1 - e^{-z})^2} =: c_3^\Delta \approx 0.81
\end{aligned}$$



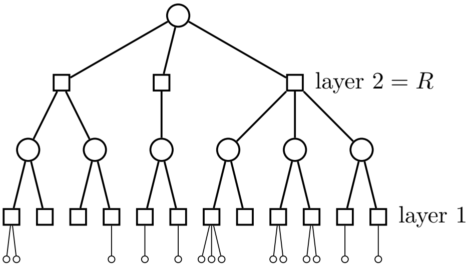
iv Interim Conclusion: What we learned about peeling T_α

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."



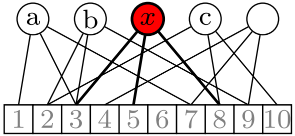
What does it mean for T_α to be locally like $G_{m,\alpha m}$?

Neighbourhoods in T_α and G

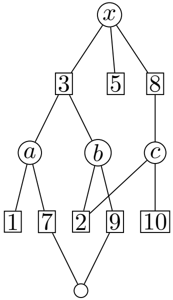
Let $R \in \mathbb{N}$. We consider

- 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.

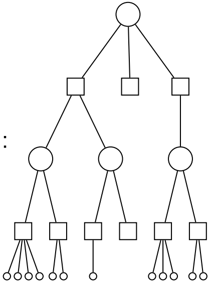
$G_{m,\alpha m}$:



$G_{m,\alpha m}^{x,R}$:



T_α^R :



What does it mean for T_α to be locally like $G_{m,\alpha m}$?

Karlsruhe Institute of Technology

Neighbourhoods in T_α and G

Let $R \in \mathbb{N}$. We consider

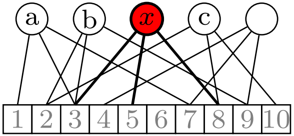
- T_α^R as before and
- for any fixed $x \in S$ the subgraph $G_{m,\alpha m}^{x,R}$ of $G_{m,\alpha m}$ induced by all nodes with distance at most $2R$ from x.

Lemma

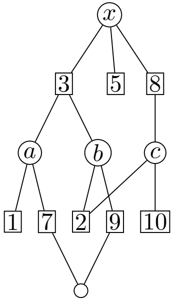
For any $R \in \mathbb{N}$, the **distribution** of $G_{m,\alpha m}^{x,R}$ converges the distribution of T_α^R, i.e.

$$\forall T : \lim_{m \to \infty} \Pr[G_{m,\alpha m}^{x,R} = T] = \Pr[T_\alpha^R = T].$$

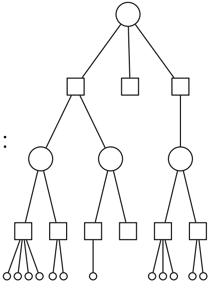
$G_{m,\alpha m}$:



$G_{m,\alpha m}^{x,R}$:



T_α^R :



Distribution of T_α^R

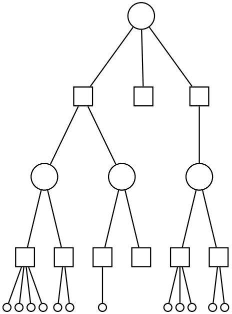
Lemma

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 □-nodes in level order. Then

$$\Pr[T_\alpha^R = T_y] = \prod_{i=1}^{k} \Pr_{Y \sim Pois(3\alpha)}[Y = y_i].$$

e.g. for $y = (2, 0, 1, 4, 2, 1, 0, 3, 2)$:

$T_y =$



v No cycles in $G^{x,R}_{m,\alpha m}$

Lemma

Assume $R = \mathcal{O}(1)$. The probability that $G^{x,R}_{m,\alpha m}$ contains a cycle is $\mathcal{O}(1/m)$.

No cycles in $G^{x,R}_{m,\alpha m}$

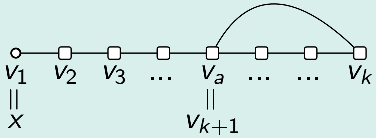
Lemma

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 $(v_1 = x, v_2, \ldots, v_k, v_{k+1} = v_a)$ of nodes with $a \in [k]$
- of length $k \leq 4R$ (consider BFS tree for x and additional edge in it)
- for each $i \in \{1, \ldots, k\}$ an index $j_i \in \{1, 2, 3\}$ of the hash function connecting v_i and v_{i+1}. (If $a = k - 1$ then $j_k \neq j_{k-1}$.)



No cycles in $G^{x,R}_{m,\alpha m}$

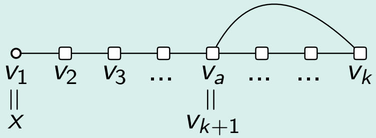
Lemma

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 $k \leq 4R$ (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 $j_k \neq j_{k-1}$.)



$\Pr[\exists \text{cycle in } G^{x,R}_{m,\alpha m}] \leq \Pr[\exists 2 \leq k \leq 4R : \exists v_2, \dots, v_k : \exists a \in [k] : \exists j_1, \dots, j_k \in [3] : \forall i \in [k] : h_{j_i} \text{ connects } v_i \text{ to } v_{i+1}]$

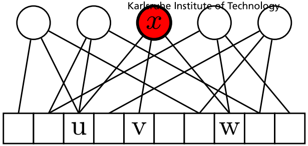
$$\leq \sum_{k=2}^{4R} \sum_{v_2,\dots,v_k} \sum_{a=1}^{k} \sum_{j_1,\dots,j_k} \prod_{i=1}^{k} \Pr[h_{j_i} \text{ connects } v_i \text{ to } v_{i+1}] \leq \sum_{k=2}^{4R} (\max\{m,n\})^{k-1} \cdot k \cdot 3^k \left(\tfrac{1}{m}\right)^k = \tfrac{1}{m} \sum_{k=2}^{4R} k \cdot 3^k = \mathcal{O}(1/m). \qquad \square$$

Distribution of $G^{x,R}_{m,\alpha m}$

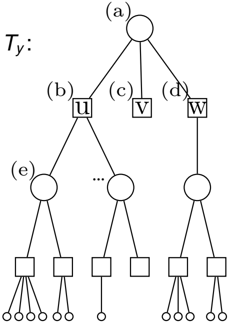
Lemma

Let T_y be a possible outcome of T^R_α as before. Then

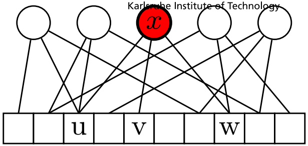
$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \overset{m\to\infty}{\longrightarrow} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$



T_y:



v Distribution of $G^{x,R}_{m,\alpha m}$



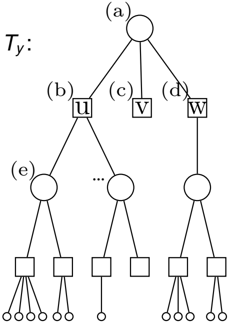
Lemma

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.

T_y:

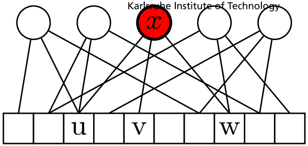


The following things have to "go right" for $G^{x,R}_{m,\alpha m} = T_y$.

- 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 $\sim \mathcal{U}([m])$.

Distribution of $G^{x,R}_{m,\alpha m}$



Lemma

Let T_y be a possible outcome of T^R_α as before. Then

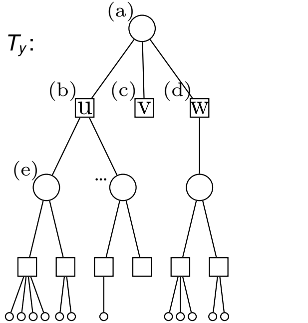
$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

"Proof by example", using T_y shown on the right.

- Exactly $y_1 = 2$ of the remaining hash values are u.
  $\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-3,\frac{1}{m})}[Y = 2] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y = 2]$. → 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 $\sim\mathcal{U}([m]\setminus\{u\})$. → exercise

T_y:

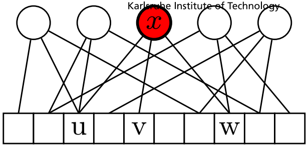


v Distribution of $G^{x,R}_{m,\alpha m}$

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.

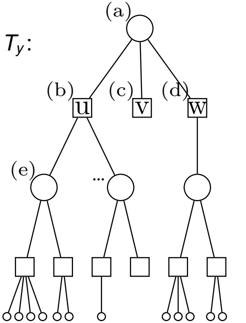
- c None of the remaining hash values are v.
  $\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-5,\frac{1}{m-1})}[Y = 0] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y = 0]$.

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.
  $\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-5,\frac{1}{m-2})}[Y = 1] \xrightarrow{m\to\infty} \Pr_{Y\sim Pois(3\alpha)}[Y = 1]$.

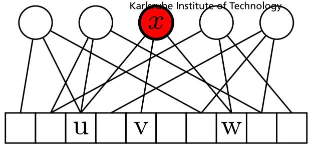
. . .

T_y:



Distribution of $G^{x,R}_{m,\alpha m}$

Karlsruhe Institute of Technology



Lemma

Let T_y be a possible outcome of T^R_α as before. Then

$$\Pr_{h_1,h_2,h_3 \sim \mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \overset{m\to\infty}{\longrightarrow} \prod_{i=1}^{k} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

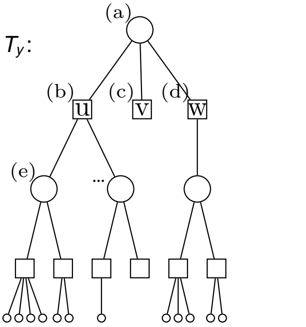
Proof sketch in general (some details ommitted)

- General case at i-th □-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.

$$\Pr_{Y\sim Bin(3\lfloor \alpha m\rfloor - c_i, \frac{1}{m-i+1})}[Y = y_i] \overset{m\to\infty}{\longrightarrow} \Pr_{Y\sim Pois(3\alpha)}[Y = y_i].$$

  Moreover, those y_i hash values must belong to distinct fresh keys. Probability $\overset{m\to\infty}{\longrightarrow}$ 1
  ↪ otherwise we'd have a cycle.

- General case for ◯-node. The two children must be fresh: probability $\overset{m\to\infty}{\longrightarrow}$ 1
  ↪ 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 $\mu_m := \Pr[x \text{ is removed when peeling } G_{m,\alpha m}] \xrightarrow{m\to\infty} 1$

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 $\lim_{R\to\infty} q_R = 0$

$\mathcal{Y}^R :=$ {all possibilities for T_α^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 $\sum_{T\in\mathcal{Y}^R} \Pr[T_\alpha^R = T] = 1$.

vi $\mu_m := \Pr[x \text{ is removed when peeling } G_{m,\alpha m}] \xrightarrow{m\to\infty} 1$

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”

$$\geq \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} \in \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}]$$

$$= \lim_{m\to\infty} \sum\nolimits_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \Pr[G^{x,R}_{m,\alpha m} = T]$$

$$= \sum\nolimits_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} = T]$$ *finite* sums commute with limit

$$= \sum\nolimits_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \Pr[T_\alpha^R = T]$$ 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: $\Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_2]$ $\square$

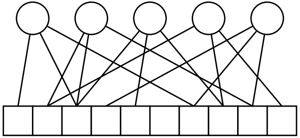
vii Proof of the Peeling Theorem

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

Karlsruhe Institute of Technology

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 $G_{m,\alpha m}$?