Computational Geometry

Wrap-Up

Thomas Bläsius

Exam: General Comments

Goal Of The Exam

- we have to certify that you
  - know the content of the lecture
  - understand it
  - are able to use the learned techniques

Exam: General Comments

Goal Of The Exam

- we have to certify that you
  - know the content of the lecture
  - understand it
  - are able to use the learned techniques
- oral exam: easier to test the first two points
- focus: second point (but I will also try to make the third point relevant)

Exam: General Comments

Goal Of The Exam

- we have to certify that you
  - know the content of the lecture
  - understand it
  - are able to use the learned techniques
- oral exam: easier to test the first two points
- focus: second point (but I will also try to make the third point relevant)

Preparation

- step 1: recap and understand the content

Exam: General Comments

Goal Of The Exam

- we have to certify that you
  - know the content of the lecture
  - understand it
  - are able to use the learned techniques
- oral exam: easier to test the first two points
- focus: second point (but I will also try to make the third point relevant)

Preparation

- step 1: recap and understand the content
- step 2: explain the content to someone

Exam: General Comments

Goal Of The Exam

- we have to certify that you
  - know the content of the lecture
  - understand it
  - are able to use the learned techniques
- oral exam: easier to test the first two points
- focus: second point (but I will also try to make the third point relevant)

Preparation

- step 1: recap and understand the content
- step 2: explain the content to someone
- **Note:** Step 2 is exactly what happens in the exam. Please practice this!

Exam: General Comments

Goal Of The Exam

- we have to certify that you
  - know the content of the lecture
  - understand it
  - are able to use the learned techniques
- oral exam: easier to test the first two points
- focus: second point (but I will also try to make the third point relevant)

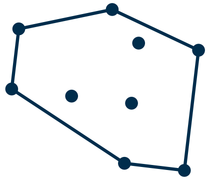
Preparation

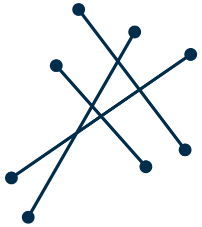
- step 1: recap and understand the content
- step 2: explain the content to someone
- **Note:** Step 2 is exactly what happens in the exam. Please practice this!
- the questions on the following slides may serve as a starting point for this

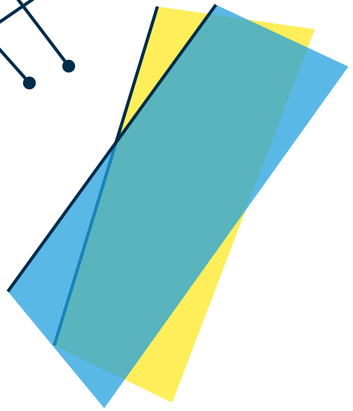
Overview

Basic Toolbox

- convex hull
- line intersection
- triangulation
- plane intersection

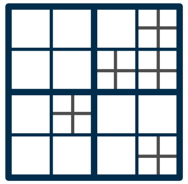


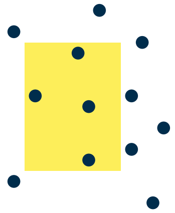


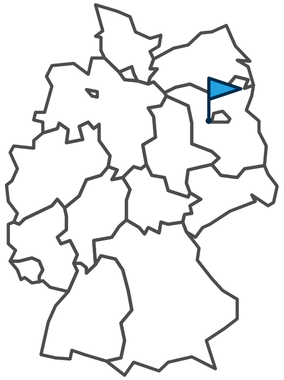


Geometric Data Structures

- orthogonal range searching
- space partitioning
- point location

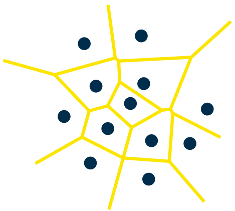


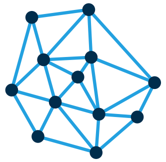




Advanced Toolbox

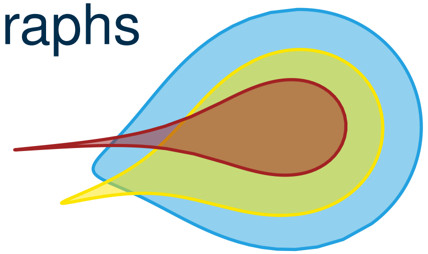
- Voronoi diagrams
- Delaunay triangulations
- origami
- complexity

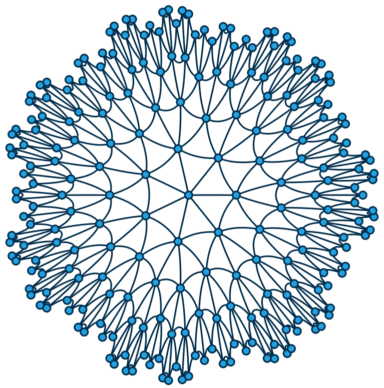




Related Topics

- What is geometry?
- hyperbolic geometry
- geometric graphs

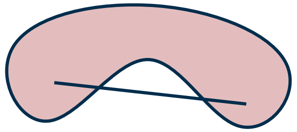


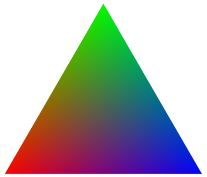


Convex Hull

Problem Definition

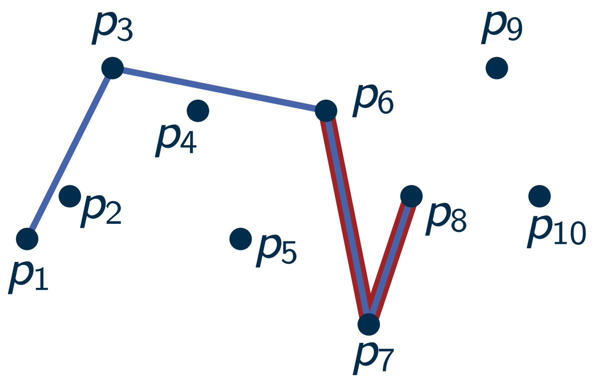
- What is a convex set? Different definitions?
- What problem did we consider?

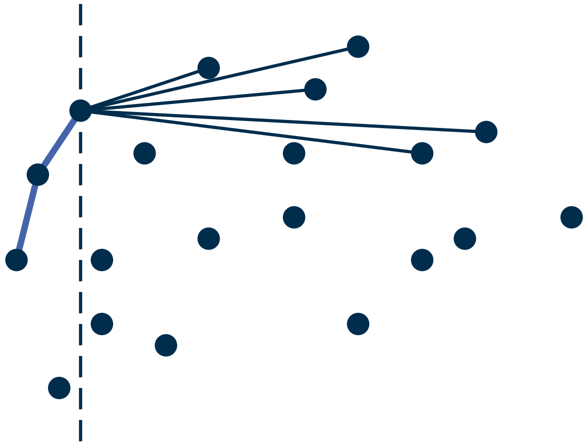




Algorithms: Andrews Algorithm (Graham Scan)/Gift Wrapping

- How does the algorithm work?
- Why is it correct?
- What is the running time? Why?
- Is it robust?





Lower Bounds

- Can it be done faster? Why not?

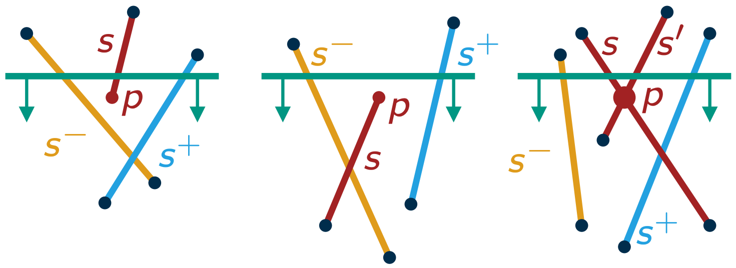
~~Convex Hull On A Simple Polygon~~

- ~~Can this be done faster?~~
- ~~Why? How do you use the given polygon?~~

Segment Intersection & Sweep-Line

Intersection Of Line Segments

- What is the general approach?
- What special cases do we need to consider?
- How do we handle multiple events at the same point?

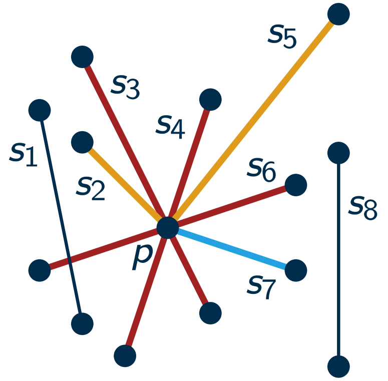


Running Time

- What is the running time? How can we show that?

Memory Consumption

- Is linear memory sufficient? How do we achieve that?
- Do we need $\Omega(n^2)$ memory, if we are not careful?



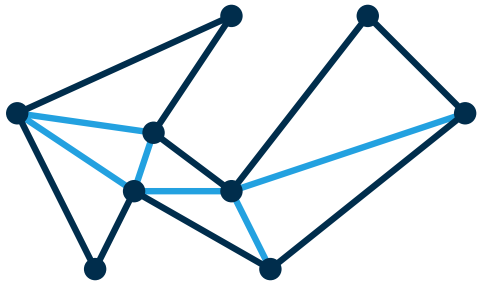
Doubly-Connected Edge List

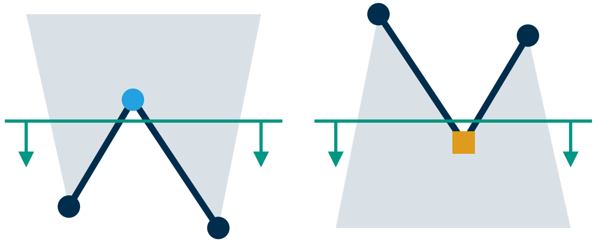
- What is it? What is it good for?
- What is the connection to segment intersection?

Triangulating Polygons

Basics

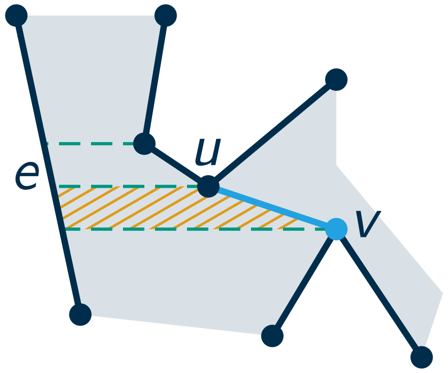
- What is the idea?
- What does y-monotone mean?
- What are split and merge vertices and how do they relate to y-monotonicity?





Eliminating Split Vertices

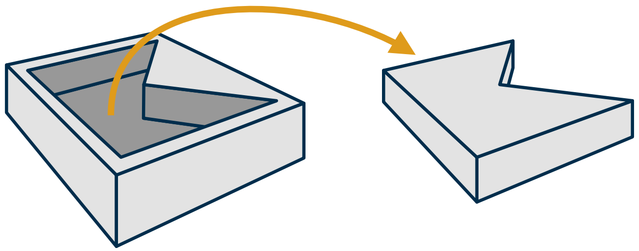
- How can we get rid of split vertices?
- What is the role of the helper?
- Why are the inserted edges crossing-free?
- How does this yield an algorithm? What events are there? How do we handle them?
- What running time do we get?



Triangulating

- What is the idea for triangulating y-monotone polygons?

Linear Programs

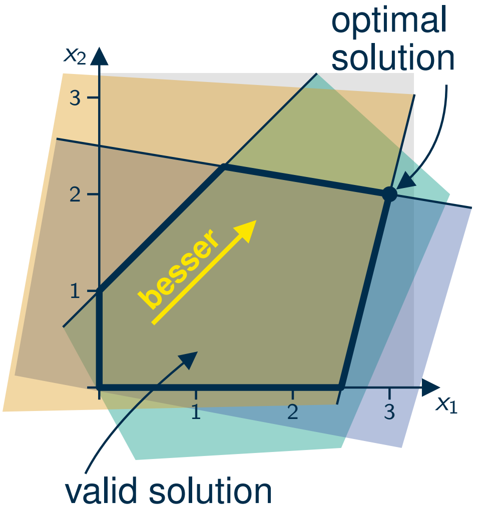


Molds

- What is the mold problem? How can it be solved using a 2D-LP?

Linear Programming

- What is an LP? What is the connection to geometry?

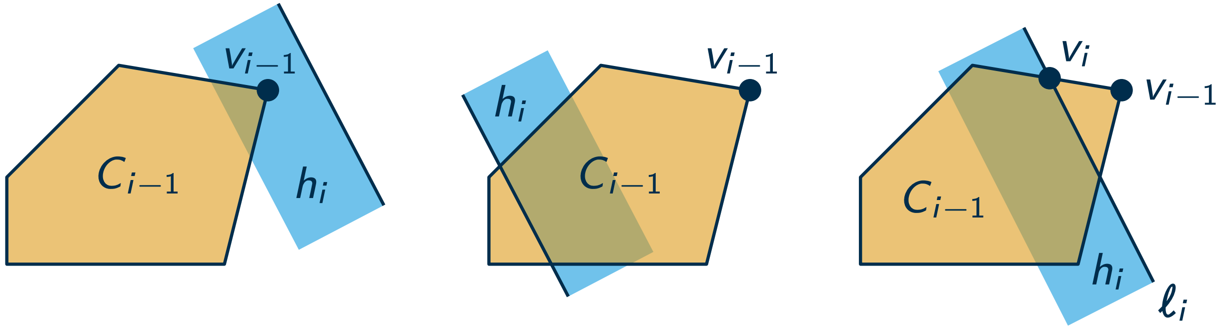


Algorithm 1: Divide And conquer

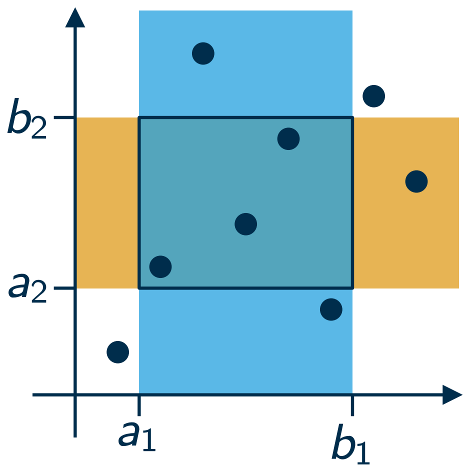
- What is the idea? What running time do we get?

Algorithm 2: Incremental Approach

- What is the idea?
- How does it reduce to a 1D-LP in each step?
- Why does a random order help?
- What is the running time? Why?
- How does the backwards-analysis help?



Orthogonal Range Queries

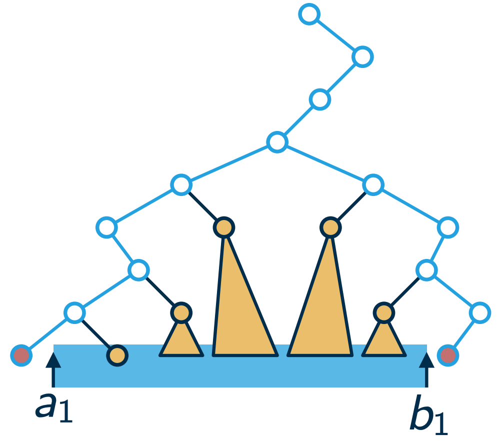


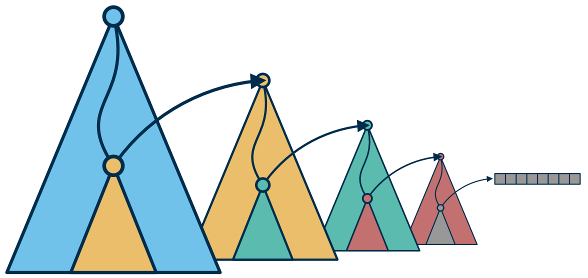
Basics

- What is the setting?

Range Trees

- What is a 2D-range tree?
- How can we answer queries and how long does it take?
- How can we build it and how long does it take? How much memory?
- How does it generalize to more dimensions?





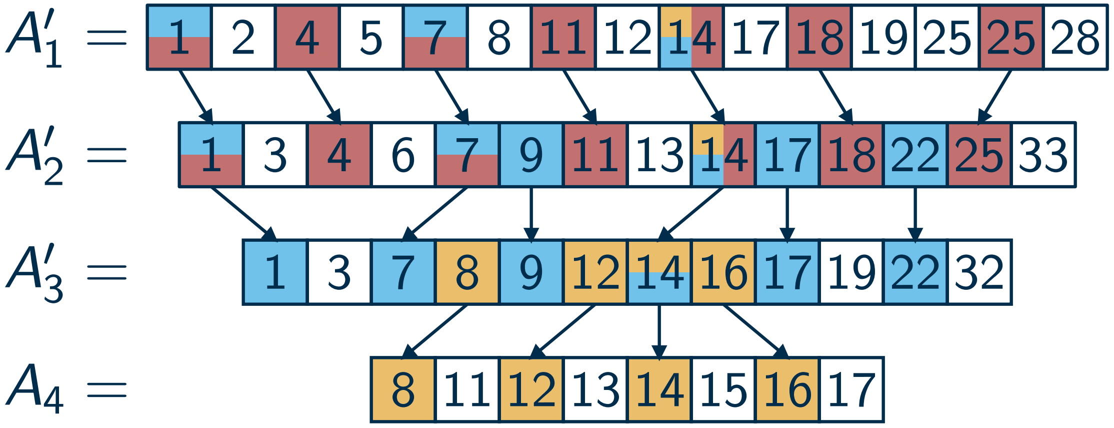
Improved Queries

- How can we save $\log n$ searches?
- How does this generalize to fractional cascading?
- How to answer queries of the form $[-\infty, b_2] \times [-\infty, b_3]$? And $[a_1, b_1] \times [-\infty, b_2] \times [-\infty, b_3]$?
- Can we get rid of the ∞? What is the impact on running time and memory?

Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

$A_1 =$	2	5	8	12	17	19	25	28
$A_2 =$	3	4	6	11	13	18	25	33
$A_3 =$	1	3	7	9	17	19	22	32
$A_4 =$	8	11	12	13	14	15	16	17



$A'_1 =$ 1 2 4 5 7 8 11 12 14 17 18 19 25 25 28

$A'_2 =$ 1 3 4 6 7 9 11 13 14 17 18 22 25 33

$A'_3 =$ 1 3 7 8 9 12 14 16 17 19 22 32

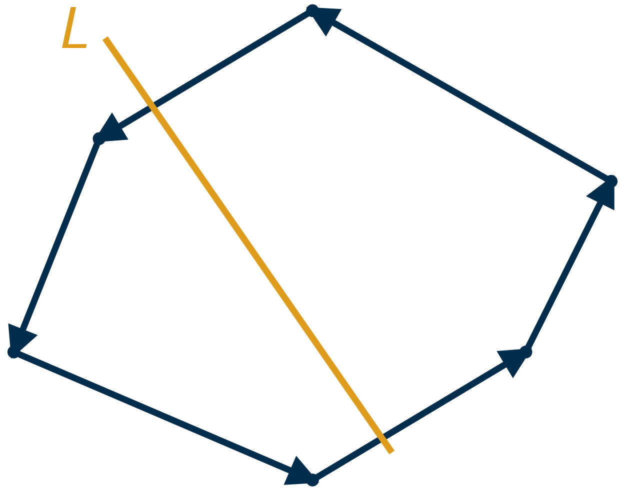
$A_4 =$ 8 11 12 13 14 15 16 17

Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?

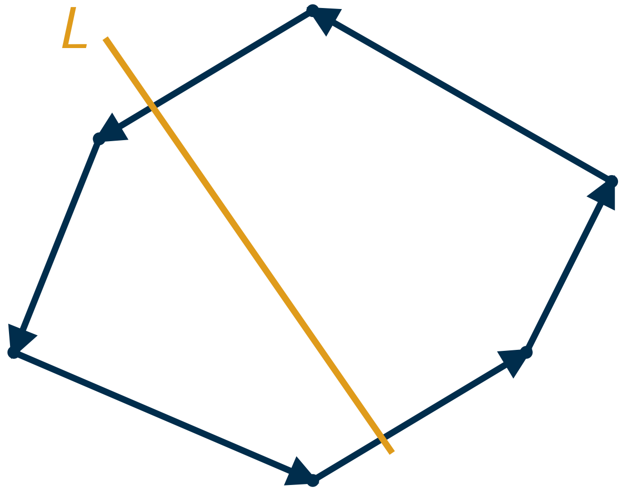


Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array

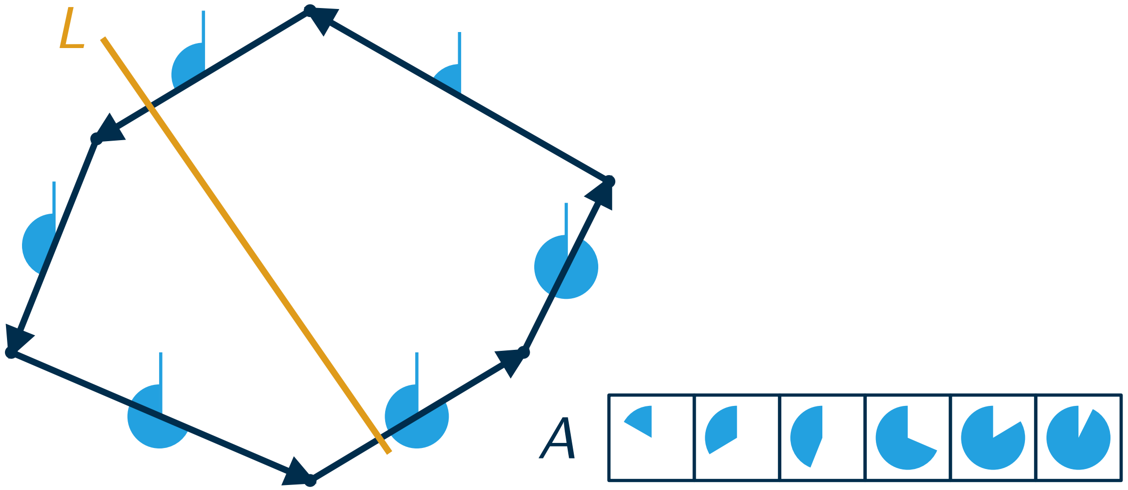


Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges



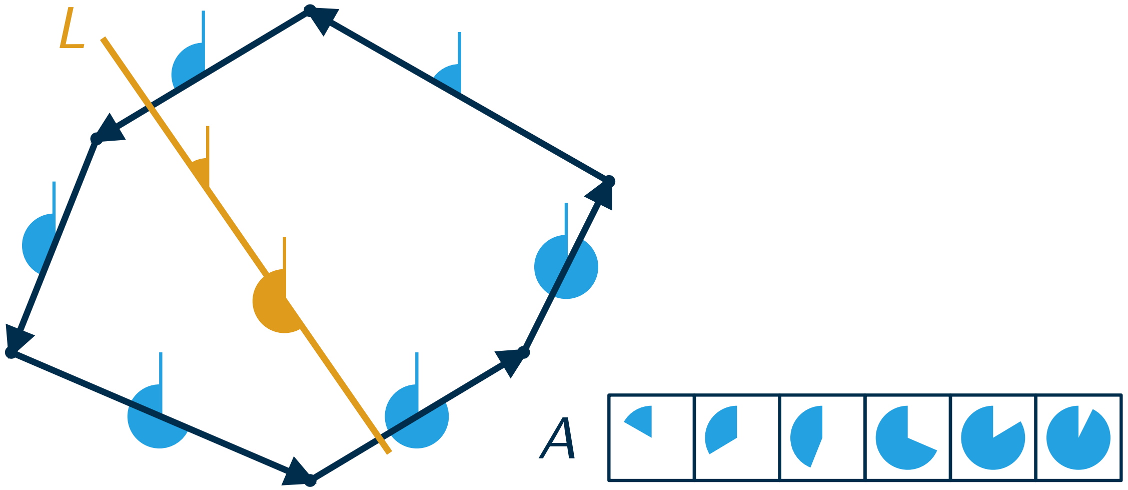
Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.

(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges
  - L has two angles

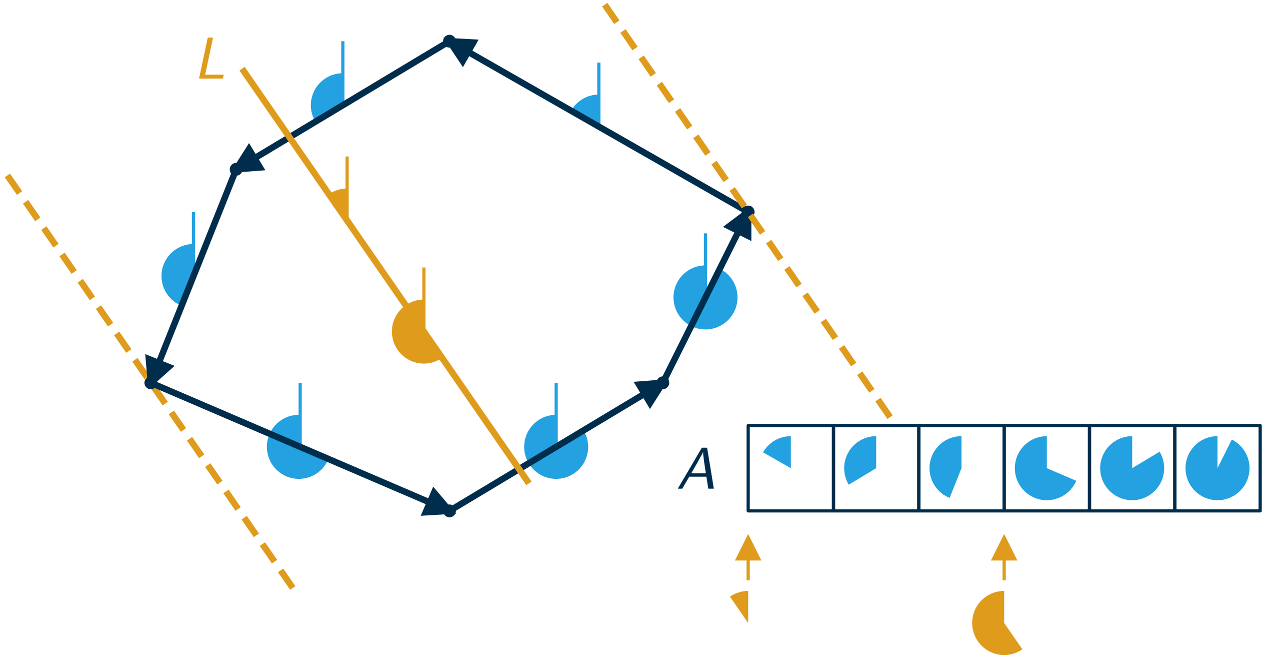


Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges
  - L has two angles
  - searching for them in A yields “tangents”



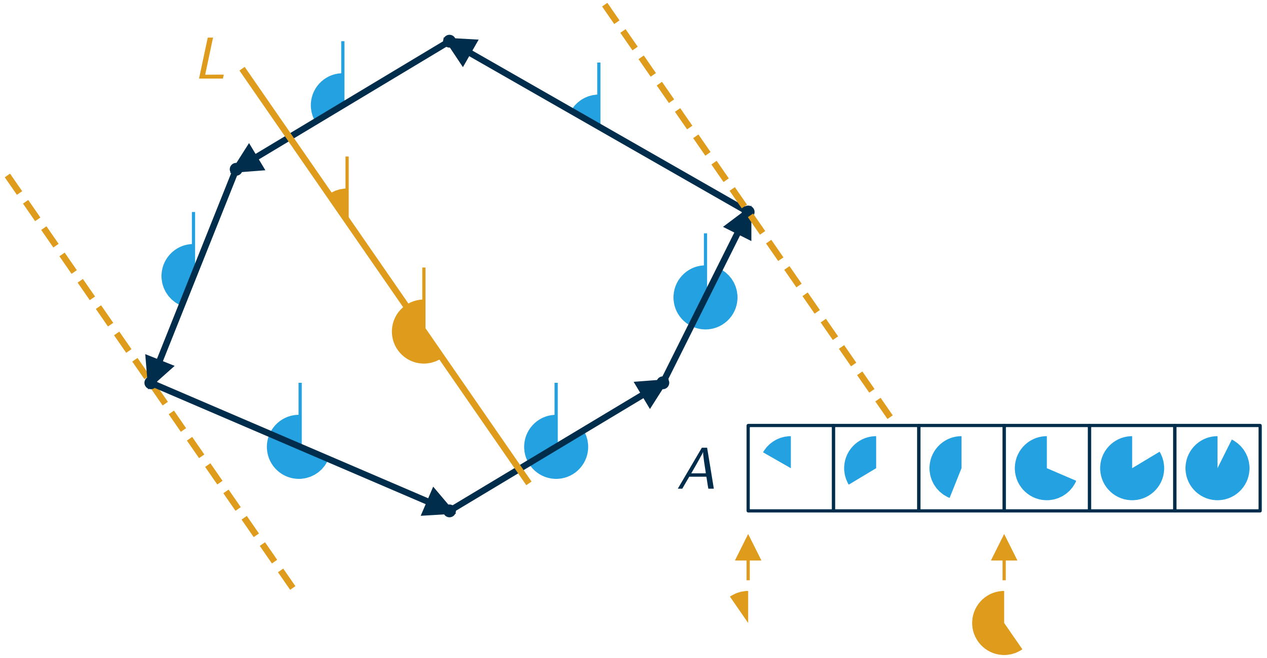
Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.

(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges
  - L has two angles
  - searching for them in A yields “tangents”
  - $L \cap P \neq \emptyset \Leftrightarrow L$ lies between the tangents



Interlude: Fractional Cascading

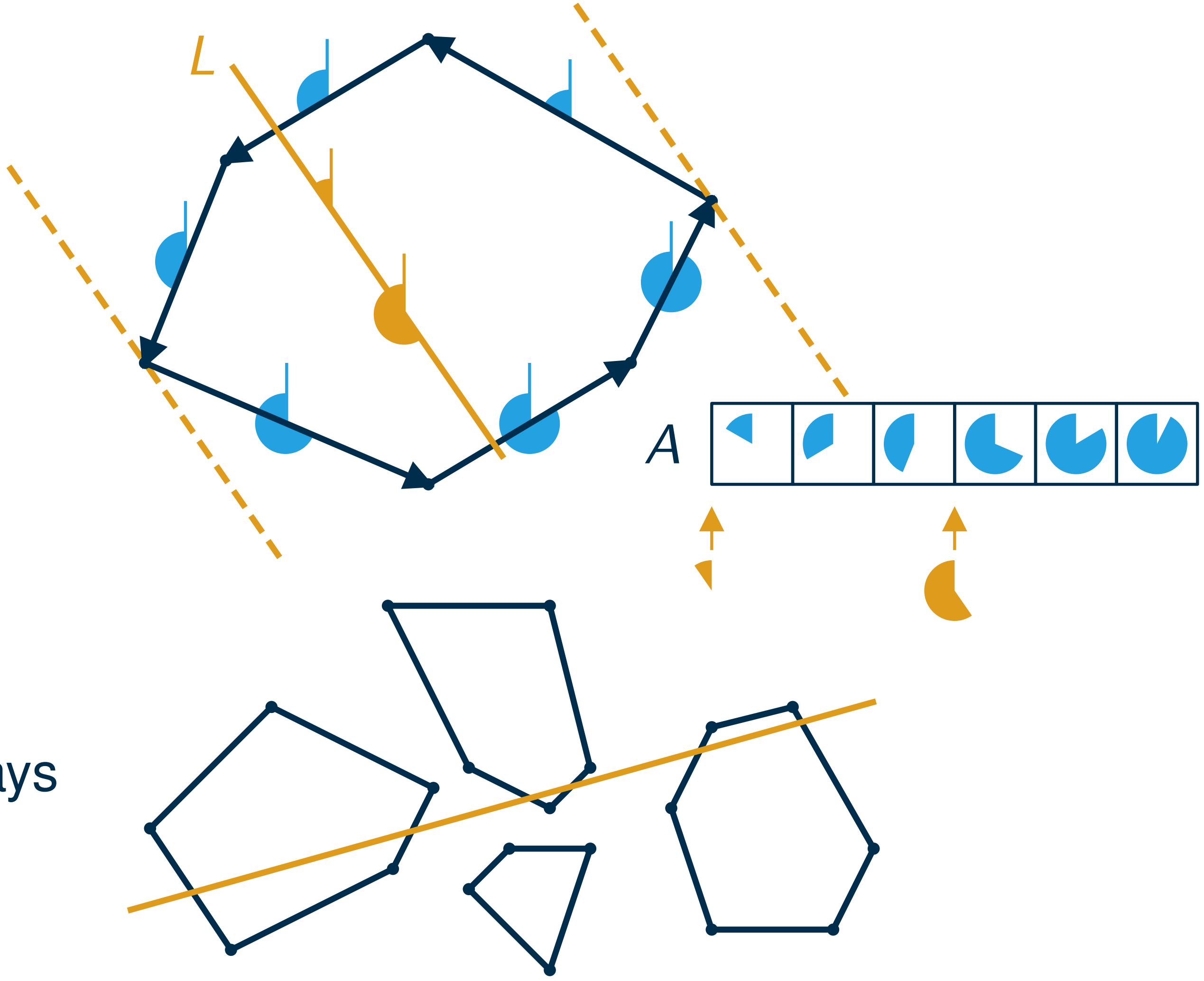
Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges
  - L has two angles
  - searching for them in A yields “tangents”
  - $L \cap P \neq \emptyset \Leftrightarrow L$ lies between the tangents

Now With Multiple Convex Polygons

- ℓ convex polygons → search in ℓ angle arrays



Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.

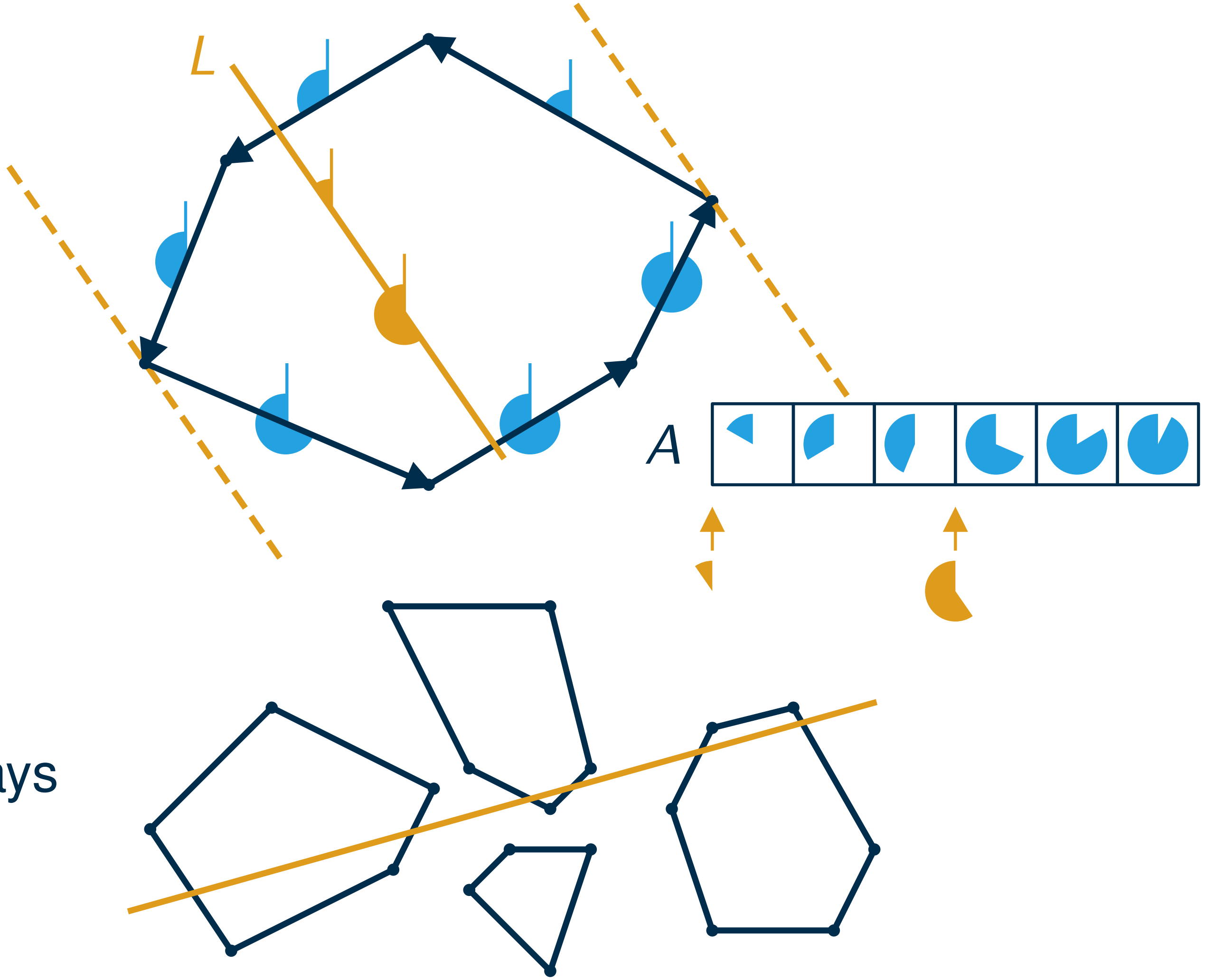
(linear preprocessing and memory)

Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges
  - L has two angles
  - searching for them in A yields “tangents”
  - $L \cap P \neq \emptyset \Leftrightarrow L$ lies between the tangents

Now With Multiple Convex Polygons

- ℓ convex polygons $\rightarrow$ search in ℓ angle arrays
- fractional cascading $\rightarrow O(\ell + \log n)$

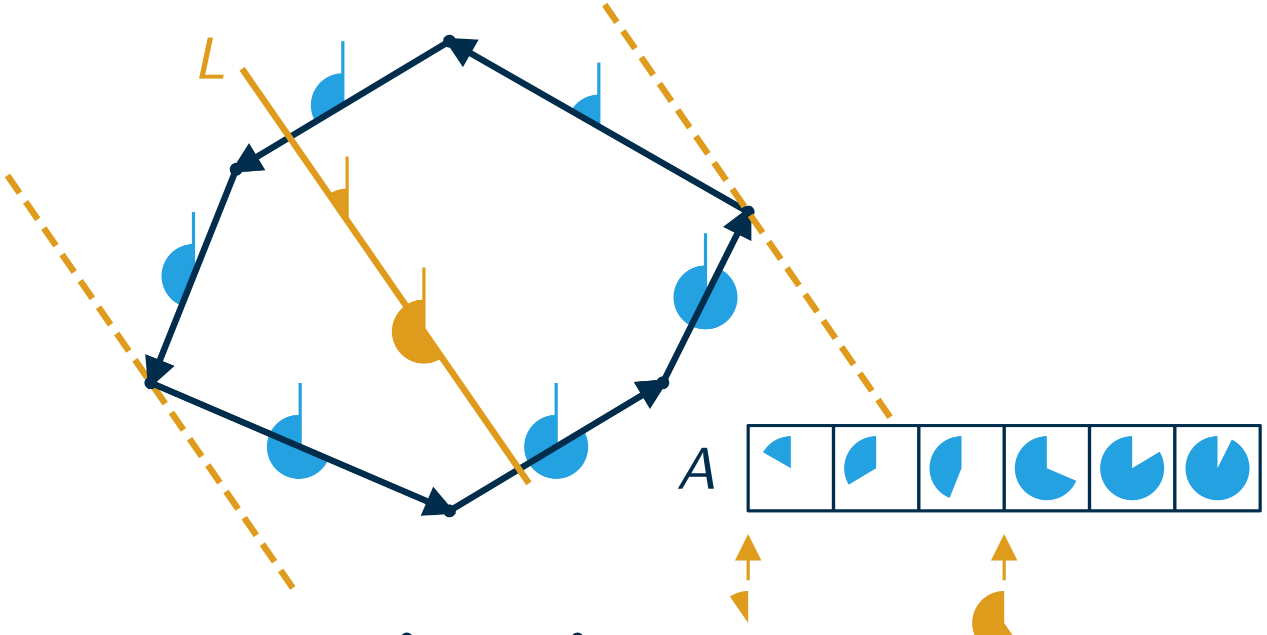


Interlude: Fractional Cascading

Core Insight: Searching for the same x in ℓ sorted arrays of size n can be done in $\ell + \log n$ time.
(linear preprocessing and memory)

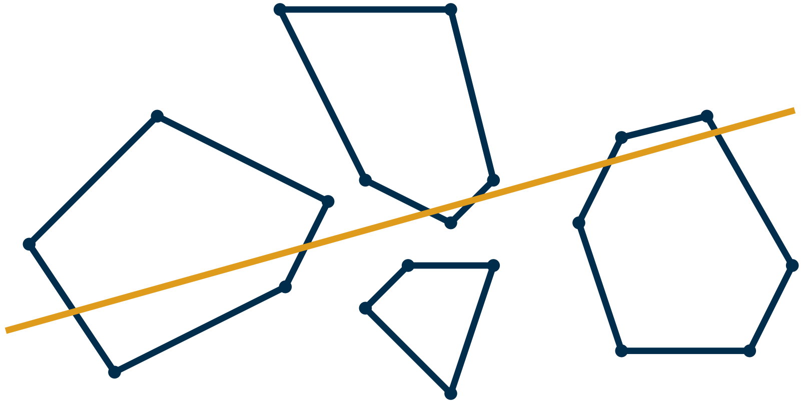
Intersecting Lines With Convex Polygons

- data: convex polygon P
- queries: Does a given line L intersect P?
- goal: reduce it to binary search in an array
  - array A: angle sequence of the edges
  - L has two angles
  - searching for them in A yields “tangents”
  - $L \cap P \neq \emptyset \Leftrightarrow L$ lies between the tangents



Now With Multiple Convex Polygons

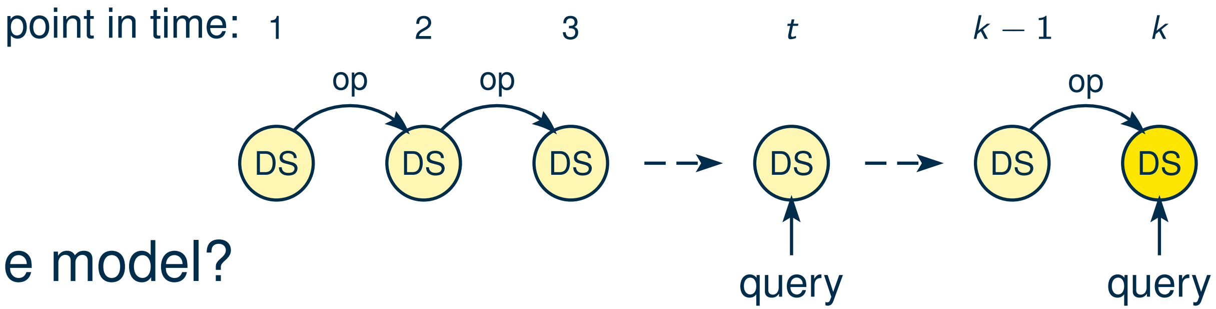
- ℓ convex polygons $\rightarrow$ search in ℓ angle arrays
- fractional cascading $\rightarrow O(\ell + \log n)$
- for nested polygons: pruning possible

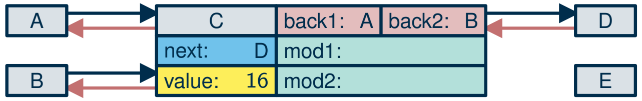


Time Travel and Point Location

Persistence

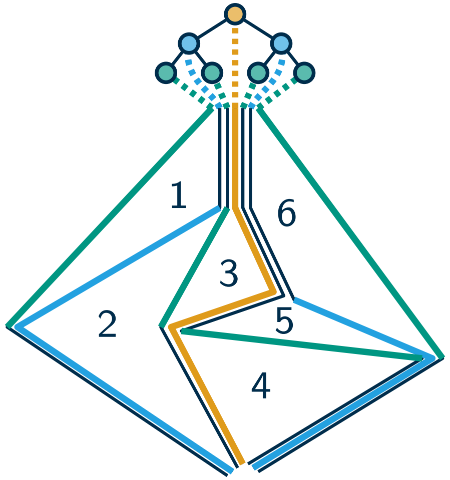
- What is a persistent data structure?
- What types of persistence are there?
- What is a data structure in the pointer machine model?
- How do we make such a data structure persistent?
  - What are atomic operations? How do we make them persistent?
  - Why are cascading recursive calls not too expensive?
  - Why do we need bounded in-degree?



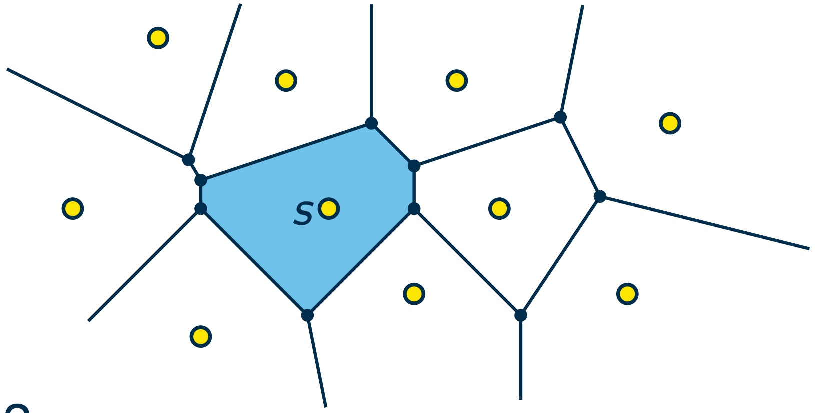


Point-Location

- What is the Problem? How can we solve it using persistence?
- What running time and memory consumption do we get?
- What is an alternative solution? How does fractional cascading help?



Voronoi Diagrams

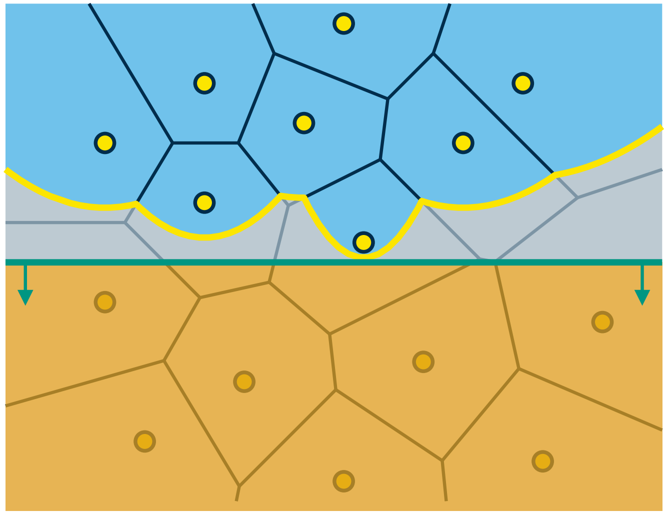


Basics

- What is the Voronoi diagram?
- Which points are vertices of the Voronoi diagram?

Sweep-Line Algo

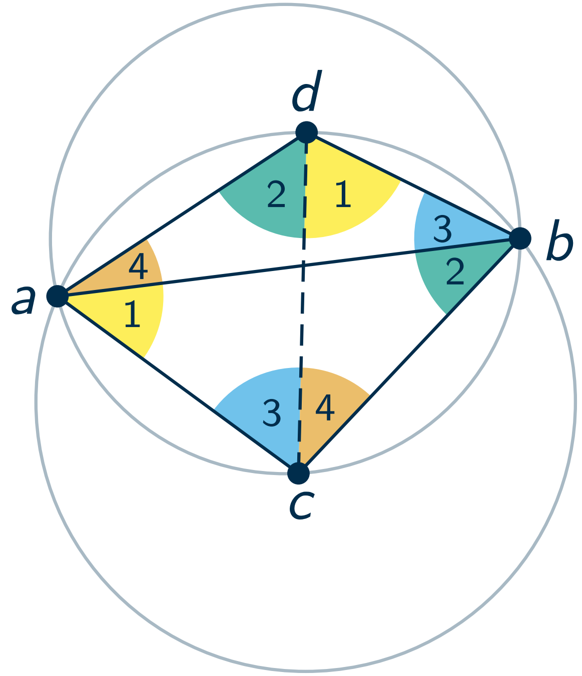
- What is different compared to previous sweep-line algorithms?
- What is the beach line? What is its shape and why?
- In what sense does the beach line only change at discrete places?
- What are these places? What are our events?
- How do we handle these events?
- How do we get the Voronoi diagram in the end?
- What about special cases?



Delaunay Triangulation

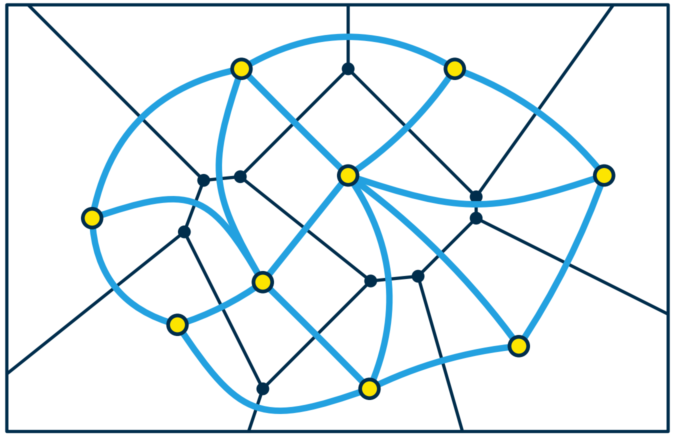
Improving Triangulations

- What is the angle vector of a triangulation?
- How can we iteratively improve a given triangulation?
- What are forbidden edges?
- Which edges are forbidden and what is the connection to circles?
- How does (the generalization of) Thales' theorem help?

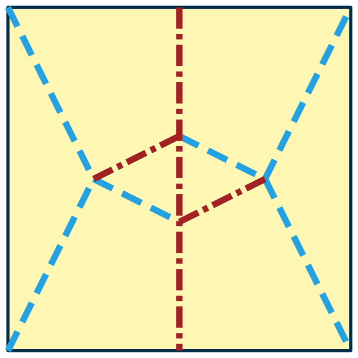


(Locally) Optimal Triangulations

- What is the Delaunay triangulation? Is it unique?
- What is the relation to the Voronoi diagram?
- Why is the relation the way it is?



Origami

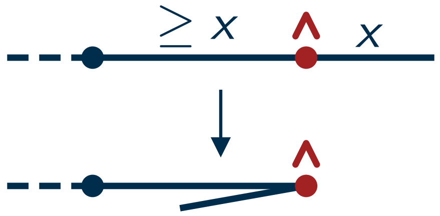


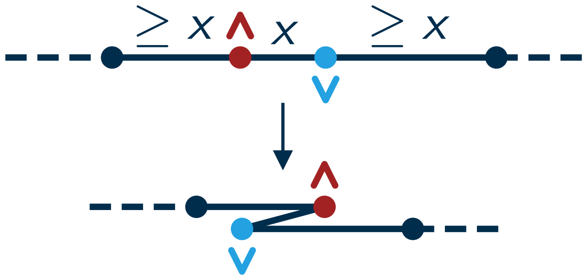
Flat Foldability

- What is a crease pattern? What is a mountain/valley pattern?

1D-Case

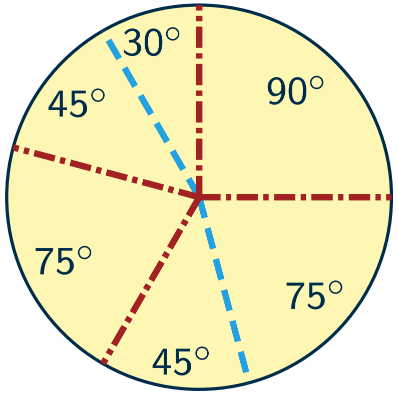
- What are crimp and end-fold?
- Are these reduction rules safe? Why?
- Are they always applicable? Why?
- How does this yield an algorithm?





2D-Case: One Vertex

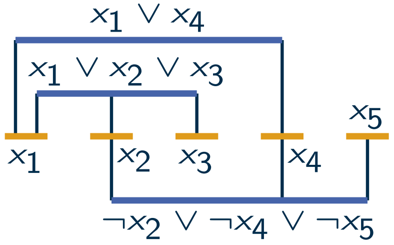
- What is the relation to the 1D-case?
- Which crease and mountain/valley patterns are flat foldable?



Multiple Vertices

- What is local flat foldability?
- What is the idea for fold-and-cut?

Hard Problems

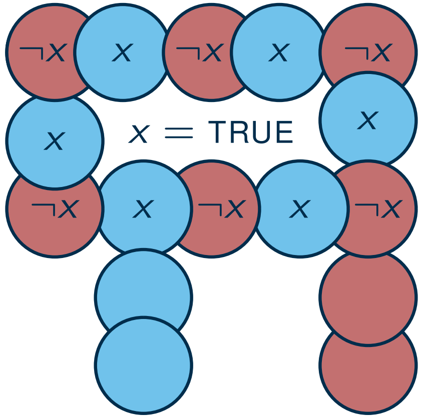


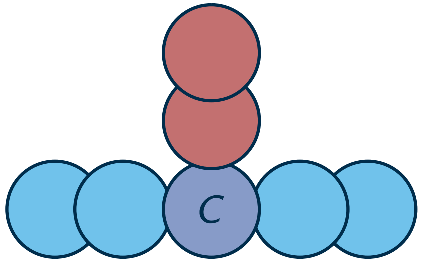
Reductions From Hard SAT-Variants

- Which SAT-variants are useful for geometric problems?
- What do we need to do in a reduction? What do we then need to show?

Example 1: Proportional Symbol Maps

- What is the problem? What variants exist?
- How do we prove hardness?

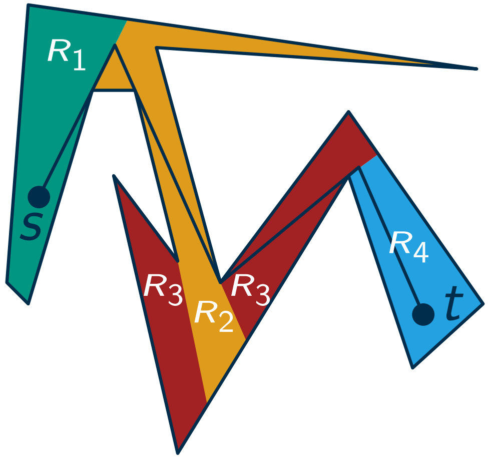




Example 2: Unit Disk Graph Recognition

- What is the problem?
- How do we prove hardness?
- Why do we need a splitter gadget?
- Why can the graph not be drawn completely differently?
- What is the complexity class $\exists\mathbb{R}$?

Searching In Sublogarithmic Time

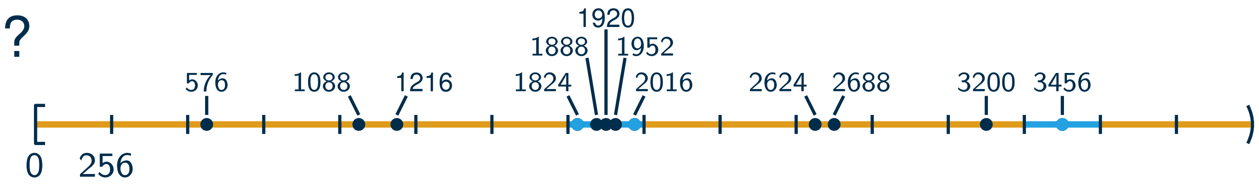


Model Of Computation

- What are RAM, real RAM, and word RAM?
- Which operations are supported in constant time?

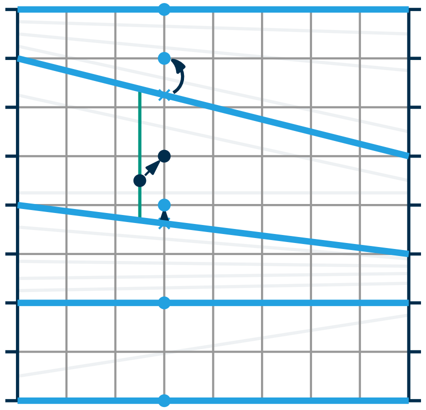
Real RAM

- Why is it useful for us? What are the pitfalls?



Word RAM: Searching

- Which two ideas do we use to speed-up the search?
- How do we branch in every search step? How high is the decision tree?
- How do we make the decision in which child do descend? In $O(1)$ time?



Word RAM: 2D-Search

- How does it relate to the 1D-case? What are the difficulties?
- What is the core idea to solve them?

Hyperbolic Geometry

Axiomatic Approach

- What is it?
- How to define geometric things without intuition?
- How are the basic terms filled with life?
- What is the Euclidean / absolute / hyperbolic plane?

Hyperbolic Plane

- What is it? Which statements still hold?
- How can I get some intuition? What is the Poincaré disk?

Hyperbolic Uniform Disk Graphs

- What is different compared to Euclidean UDGs?
- What properties do we have in the strongly hyperbolic setting?
- What can we do with it?

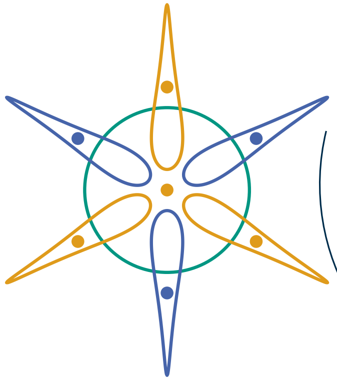
Axiom Group I: Incidence
two points define a line; every line contains two points; there are three non-collinear points

Axiom Group II: Distance
distance is a metric; tightness of triangle inequality if and only if collinear

Axiom Group III: Order
there is a point in every direction with every distance; lines split the plane into half planes

Axiom Group IV: Motion
two motions that map segments of equal length onto each other (preserving orientation)

Axiom Group V: Euclidean Parallel Axiom
line ℓ and point $P \notin \ell \Rightarrow$ at most one line through P parallel to ℓ



Good Luck!