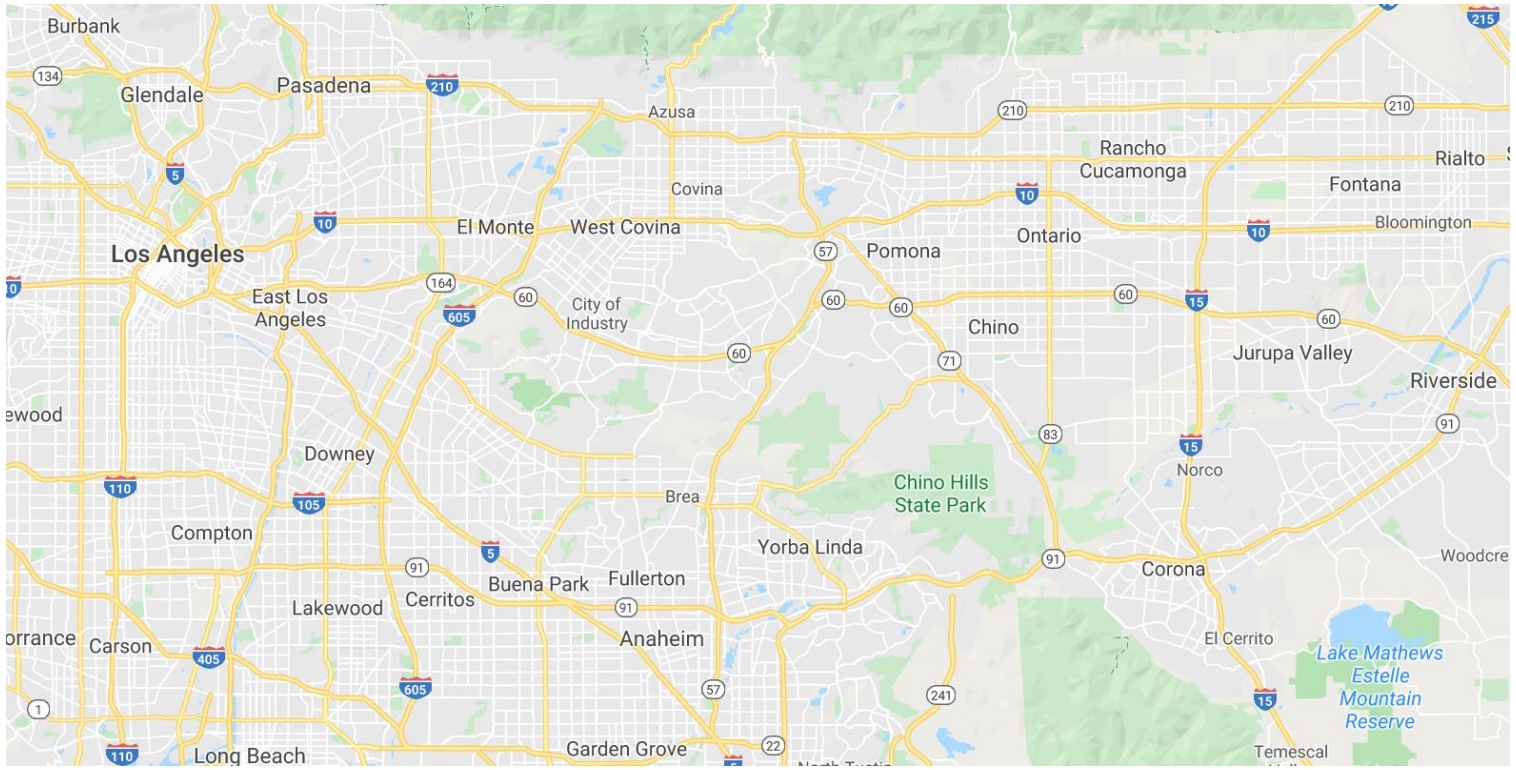
CS142: Algorithm Engineering

Geometry processing

Yan Gu

What is geometry processing?

- Graph studies the relationship of objects
- Geometry studies the locations of the objects themselves

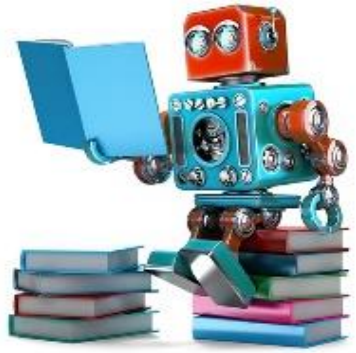


Lots of real-world applications

Every area requires geometry processing

Database / Data warehouses

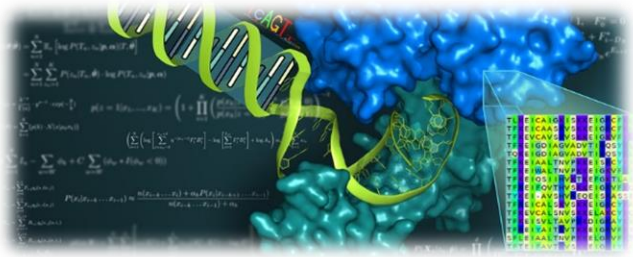
Data mining / Data science



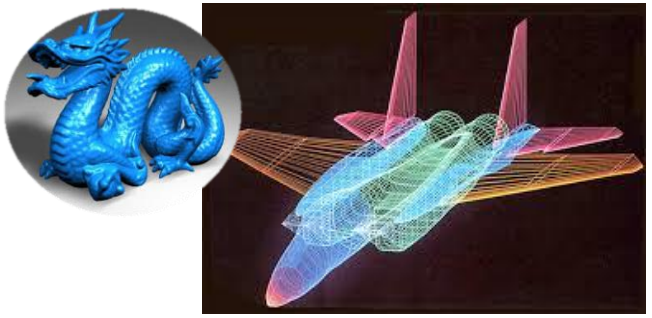
Machine learning / Artificial intelligence

Geometric Information Systems (GIS)

Computational biology



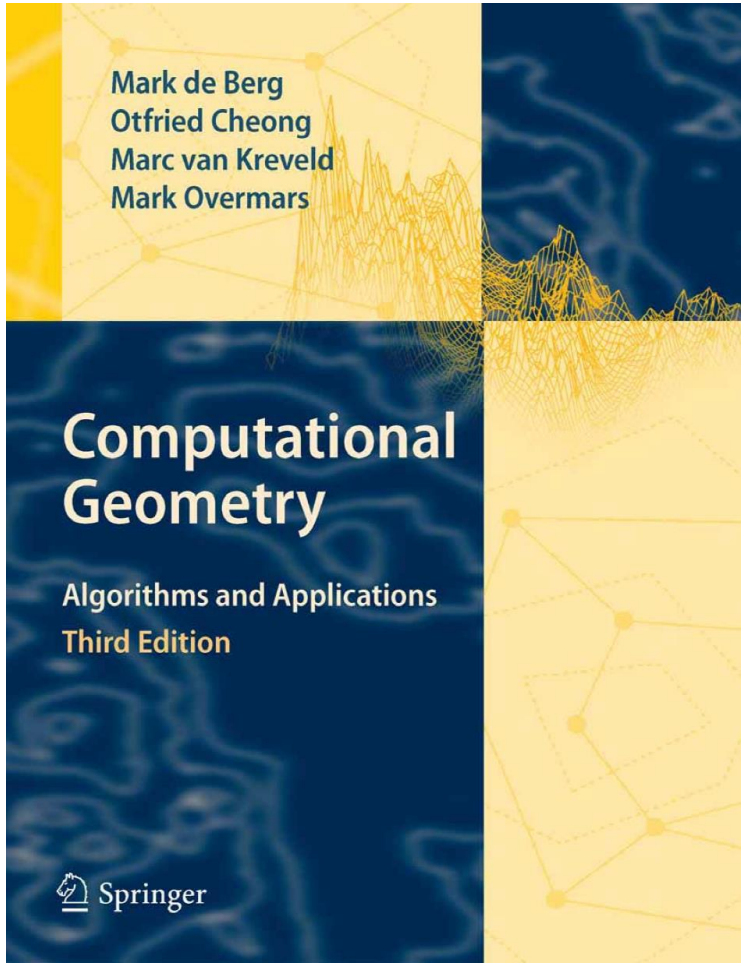
Computer graphics

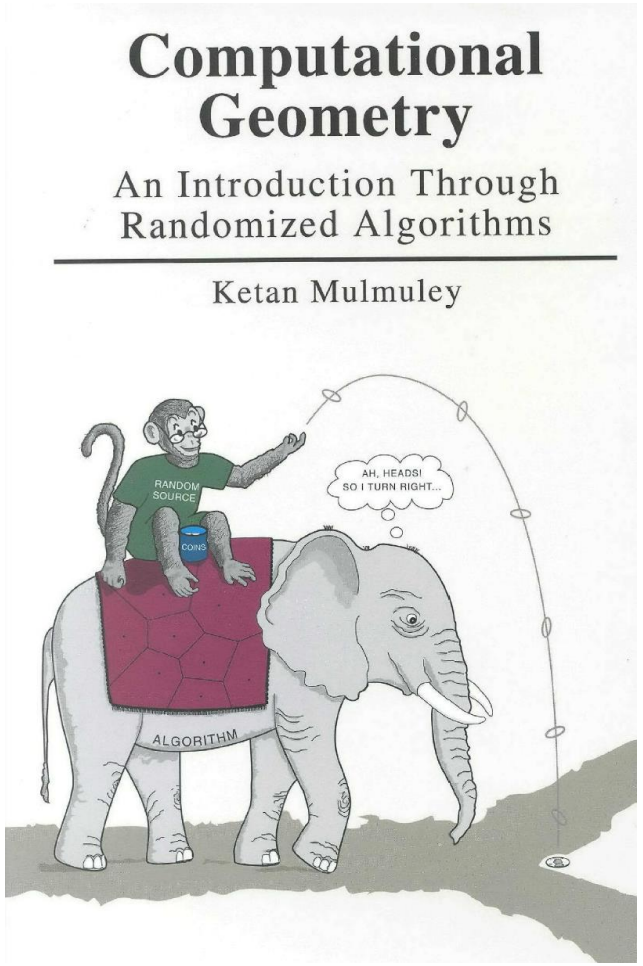


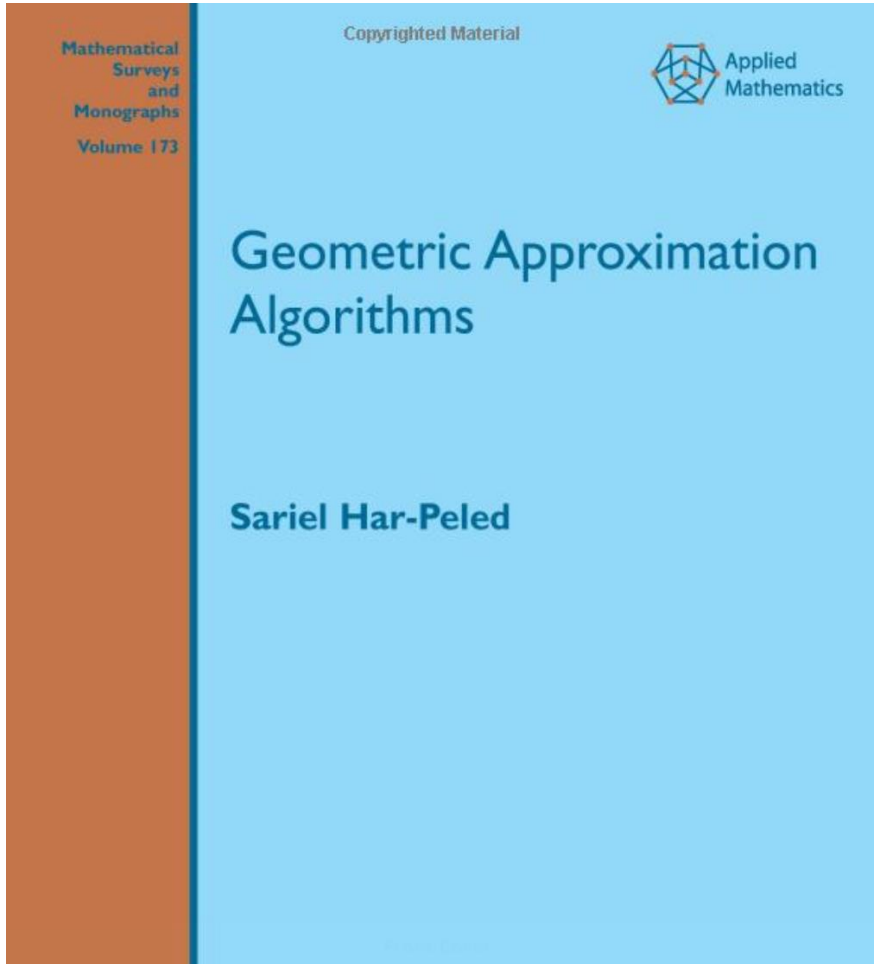
Computational Geometry

- Started in mid 70's
- Focused on abstracting the geometric problems, and design and analysis of algorithms for these problems
- Most problems well-solved in the sequential setting, but many problems in other settings remain open
- UCR CS 133 - Computational Geometry
- MIT 6.838 - Geometric Computation

Good references for sequential geometric algorithms







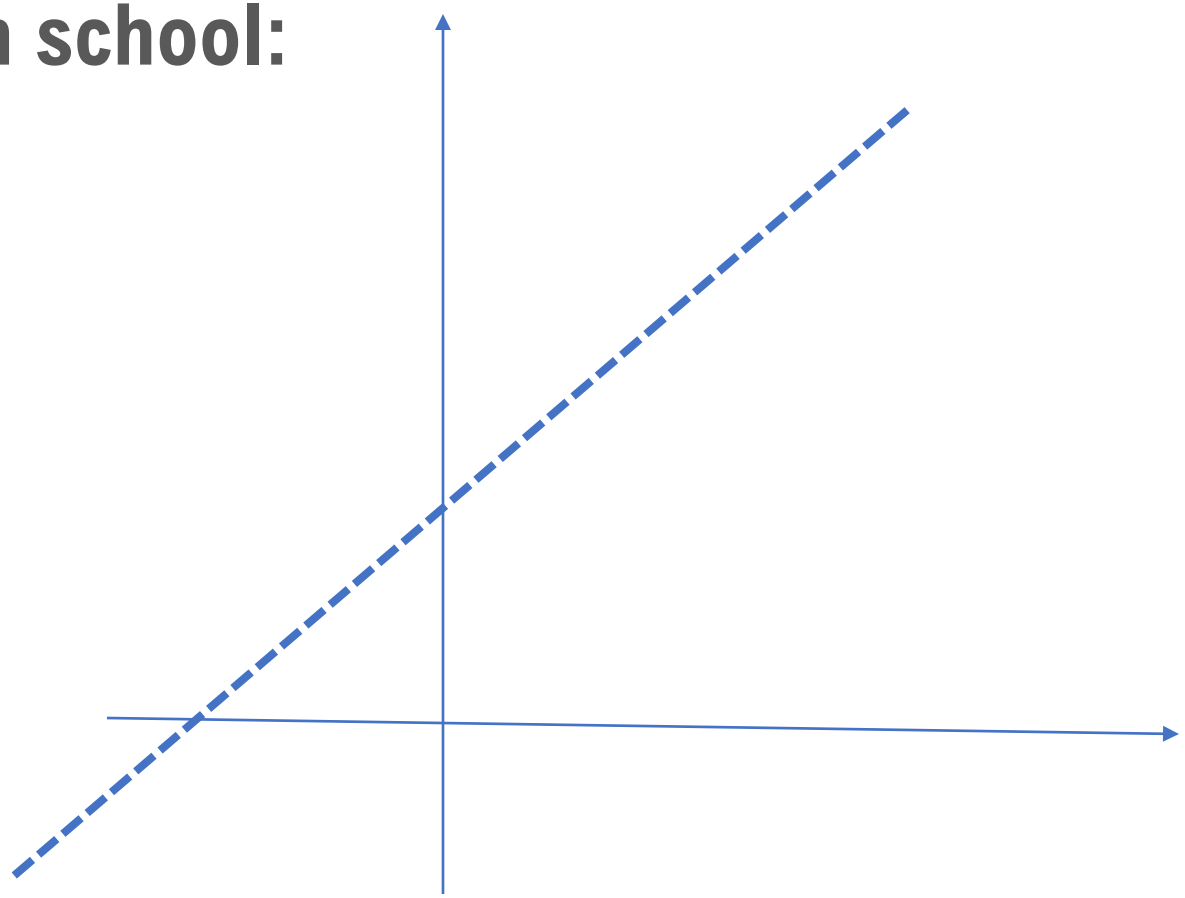
"Four Dutchmen" "The elephant book" Har-Peled's book

Why "computation" geometry?

Ways to represent a line

- **What you did in high school:**

$$y = kx + b$$



Floating-point error

- float a = 0.65f;
- float b = 0.6f;
- float c = a - b;

- What is c?
- 0.0499999523!

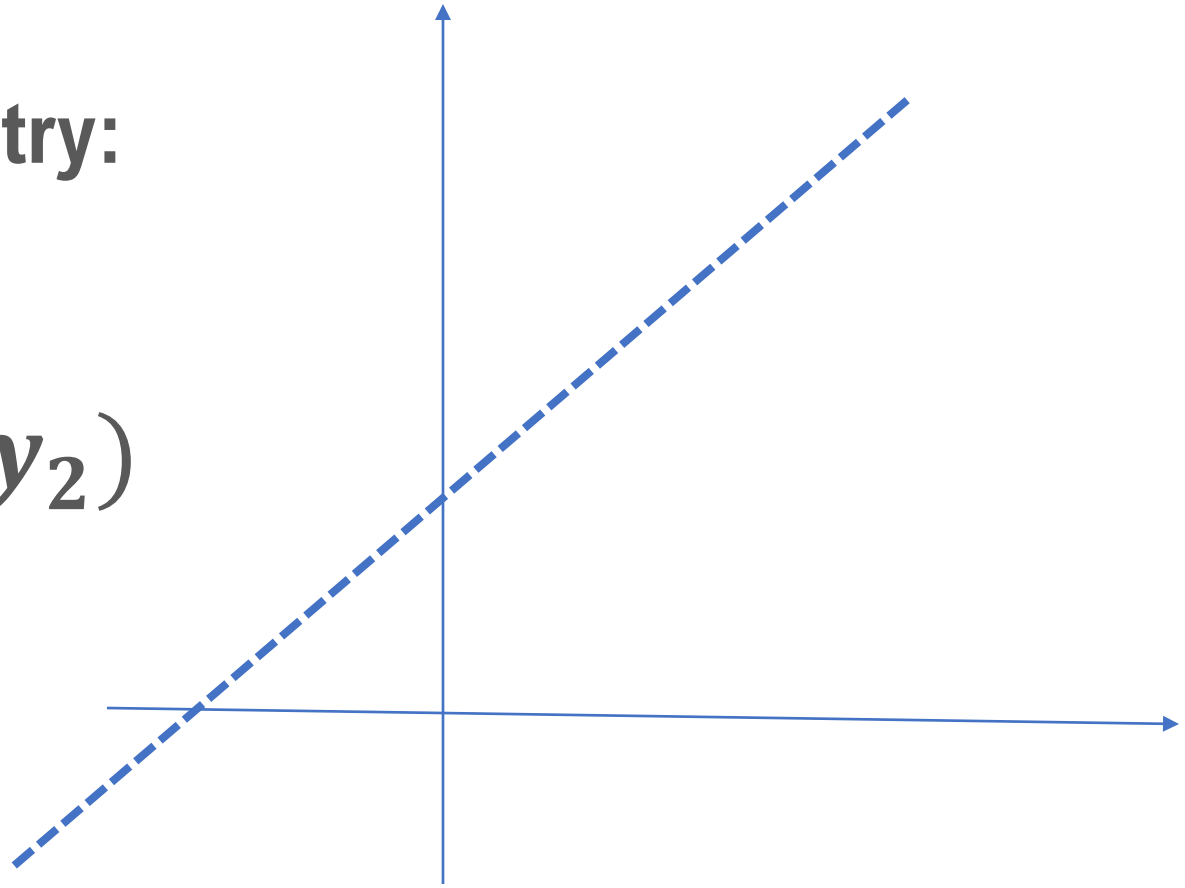
- Need to avoid division because we cannot divide a number by 0, but it is hard to check if a floating-point number is 0

Ways to represent a line

- **What we use in computation geometry:**

  Pick two points:

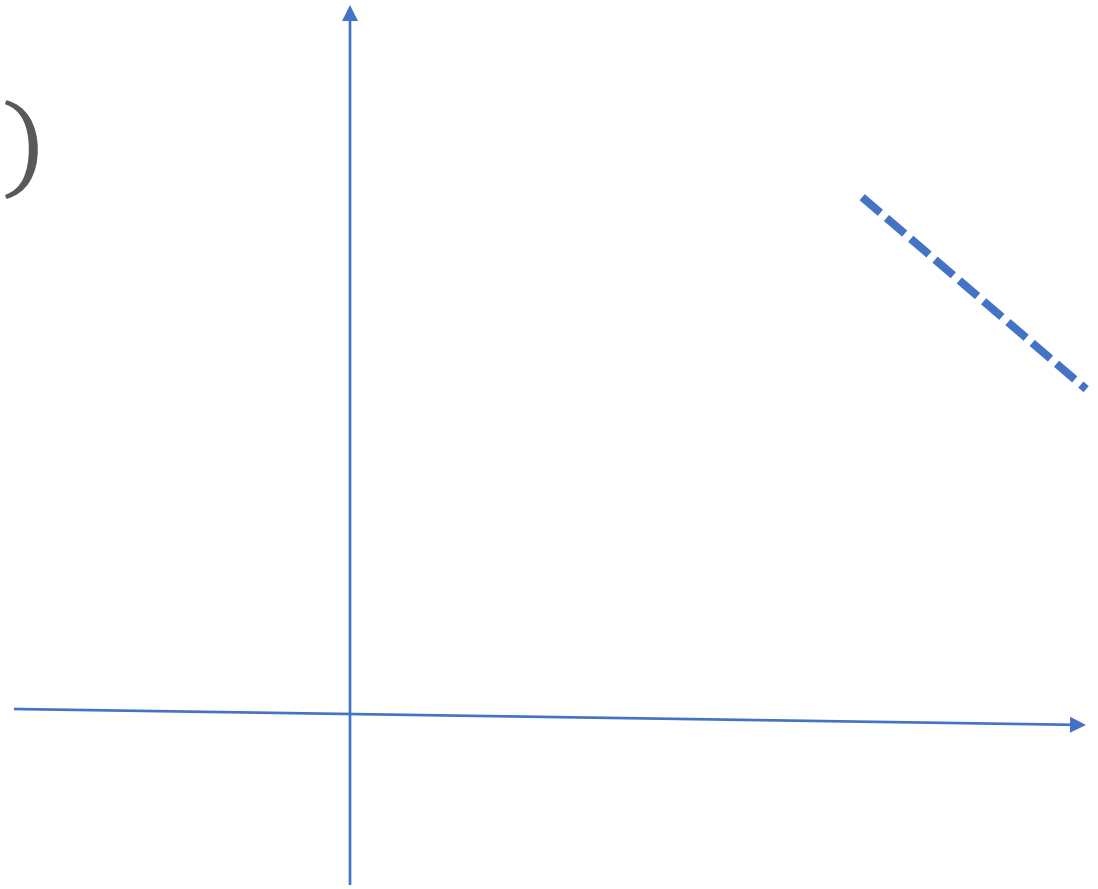
  $(x_1, y_1), (x_2, y_2)$



To represent a segment

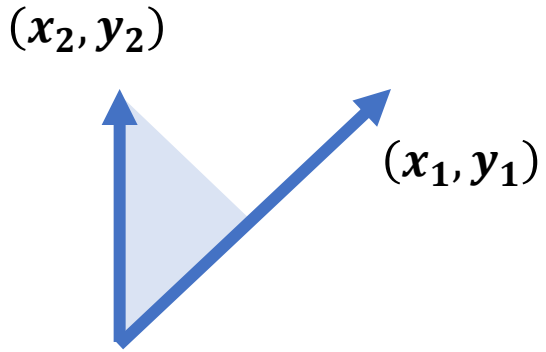
- **Use two endpoints:**

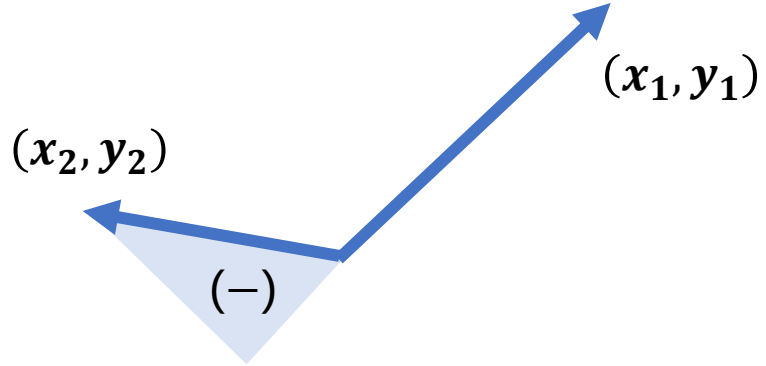
$$(x_1, y_1), (x_2, y_2)$$



Inner (dot, scalar) product

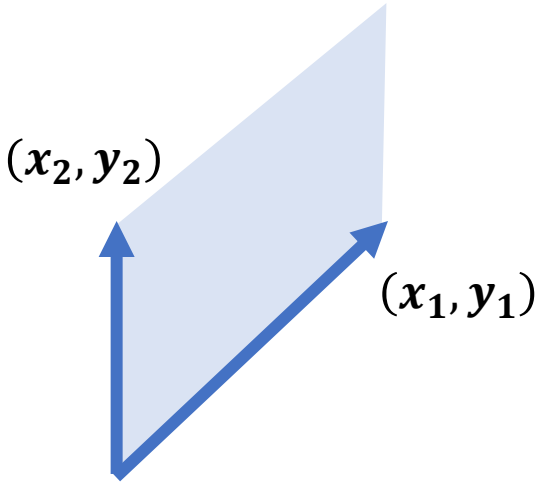
- $(x_1, y_1) \cdot (x_2, y_2) = (x_1 x_2 + y_1 y_2)$
- $(x_1, y_1) \cdot (x_2, y_2) = \|(x_1, y_1)\| \|(x_2, y_2)\| \cdot \cos \theta$
- **Digit sign indicates "front" or "back"**

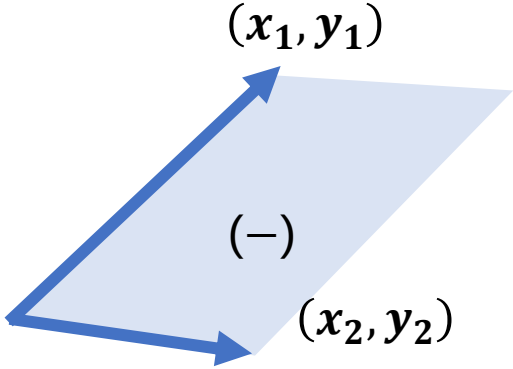




Outer (cross, vector) product

- $(x_1, y_1) \cdot (x_2, y_2) = (x_1 y_2 - y_1 x_2)$
- $(x_1, y_1) \cdot (x_2, y_2) = \|(x_1, y_1)\| \|(x_2, y_2)\| \cdot \sin \theta$
- **Digit sign indicates “right” or “left”**



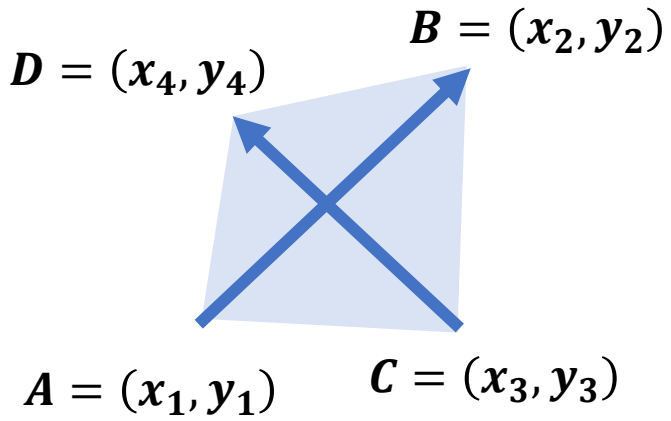


An example: checking if two segments intersect

$$(\overrightarrow{AB} \times \overrightarrow{AD}) \cdot (\overrightarrow{AB} \times \overrightarrow{AC}) < 0$$

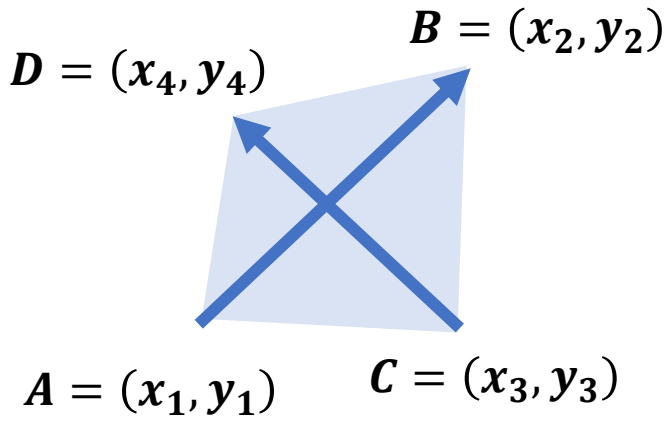
- And

$$(\overrightarrow{CD} \times \overrightarrow{CA}) \cdot (\overrightarrow{CD} \times \overrightarrow{CB}) < 0$$



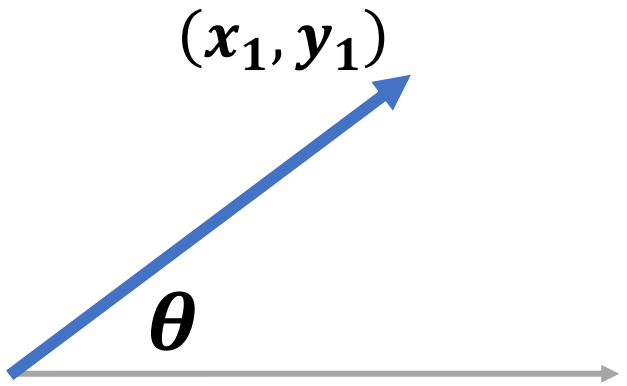
An example: finding the intersection of two segments

$$\frac{\left(\overrightarrow{AB} \times \overrightarrow{AD}\right) \cdot C - \left(\overrightarrow{AB} \times \overrightarrow{AC}\right) \cdot D}{\overrightarrow{AB} \times \overrightarrow{AD} - \overrightarrow{AB} \times \overrightarrow{AC}}$$



Another useful component: polar angle

- theta=atan2(y1,x1)

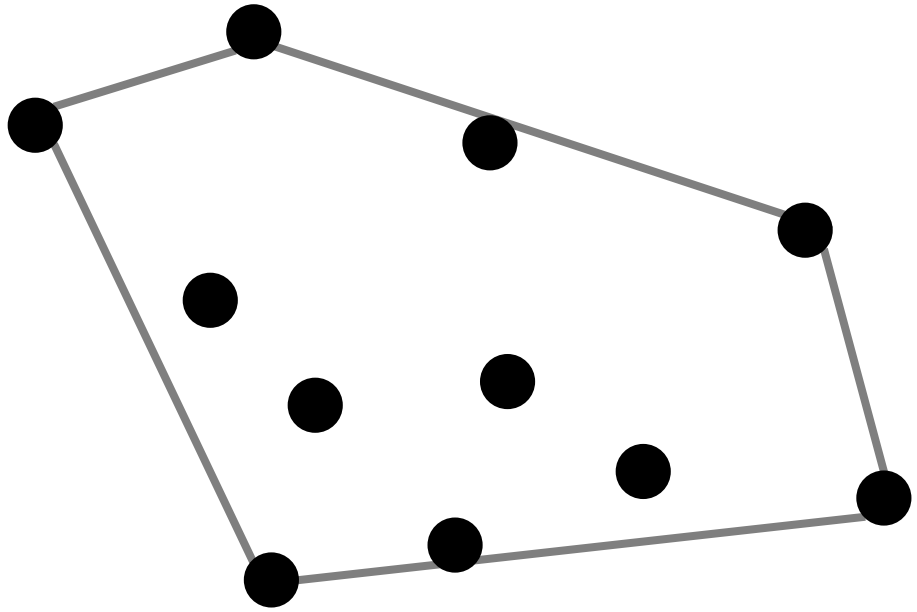


Convex Hull, Triangulation, and others

Convex hull

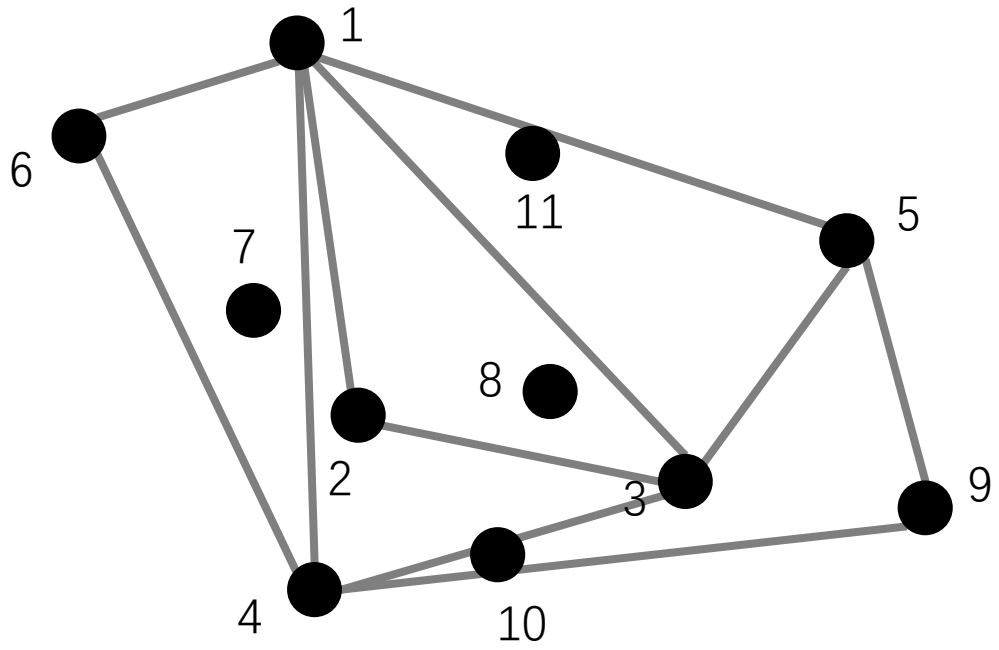
- A set is **convex** if every line segment connecting two points in the set is fully contained in the set

- Example applications: nearest/farthest point to a line, point

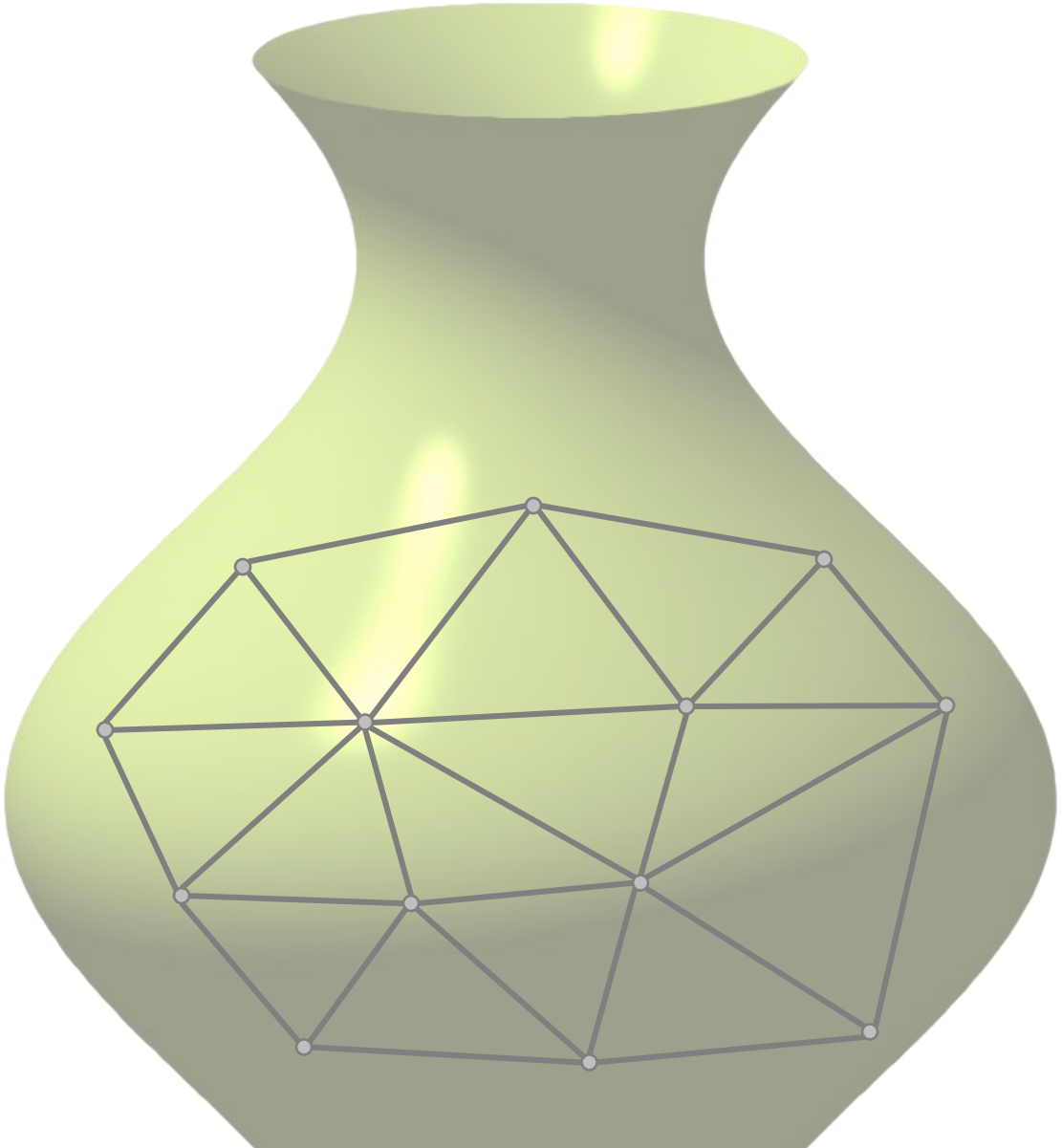


How to construct a convex hull

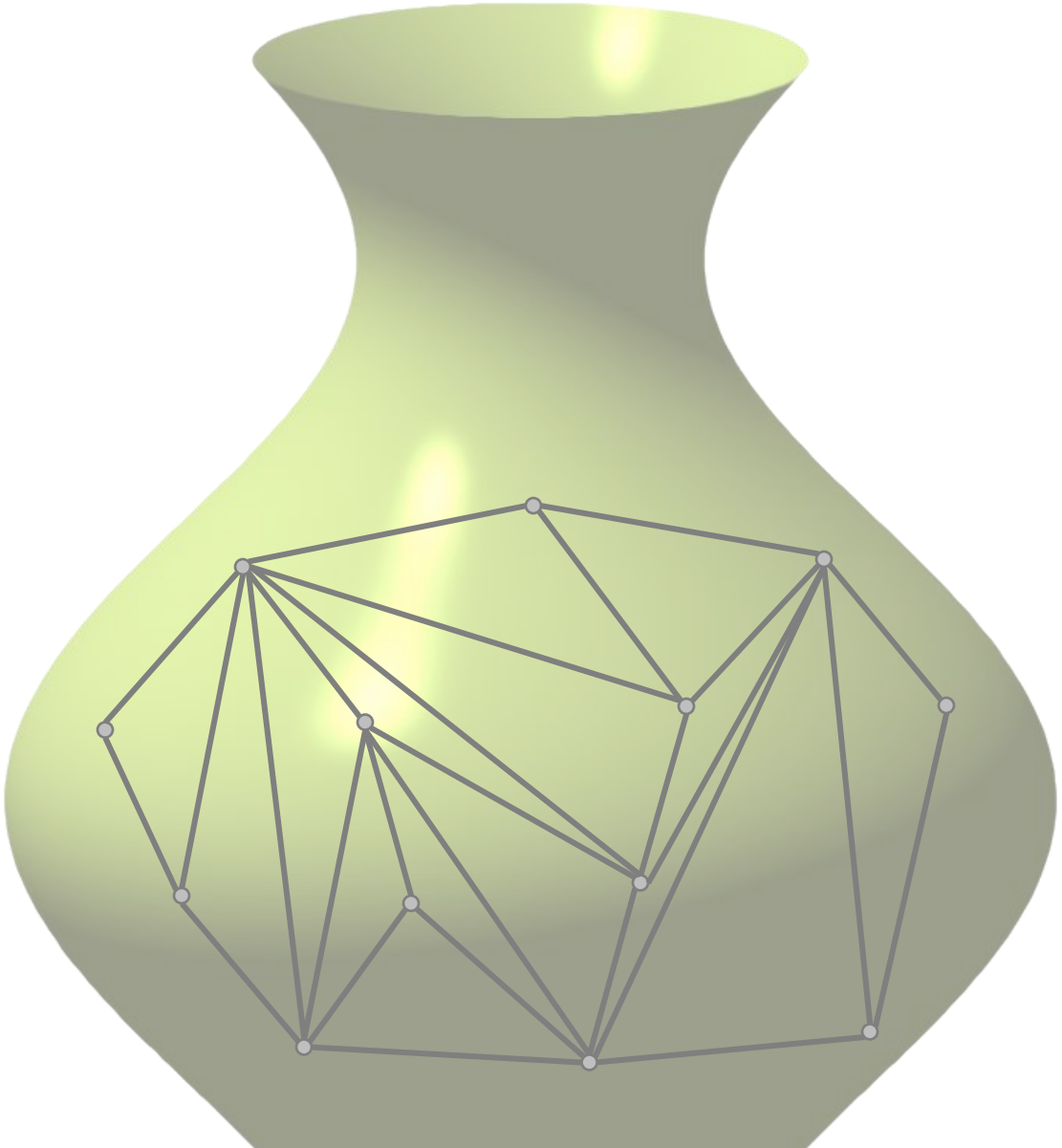
- **Dozens of algorithms sequentially, a few parallel ones for 2D**
- **Randomized incremental construction**
- **Only requires $O(\log n)$ rounds if adding all possible points**
  [BGSS SPAA 2020]



What's triangulation for?

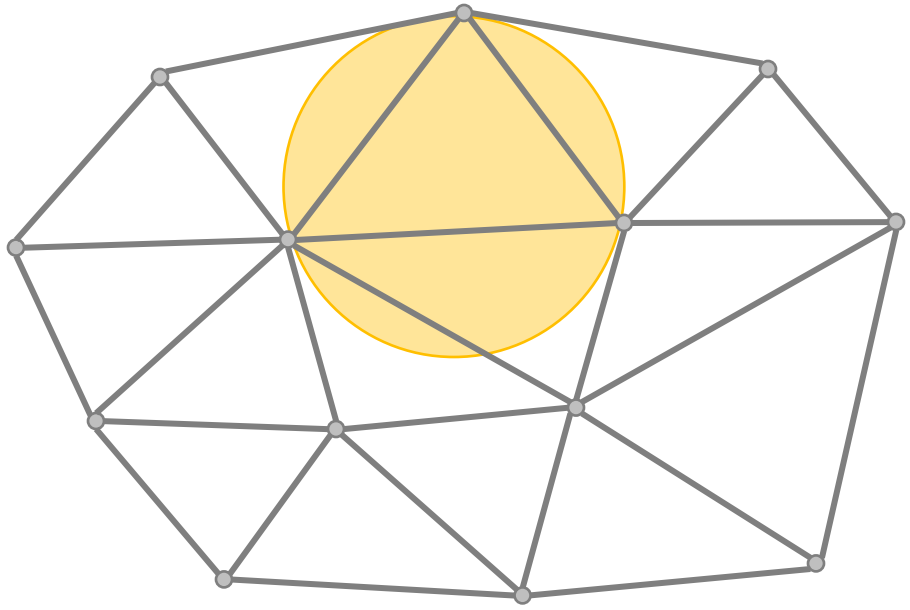


What's a good triangulation?



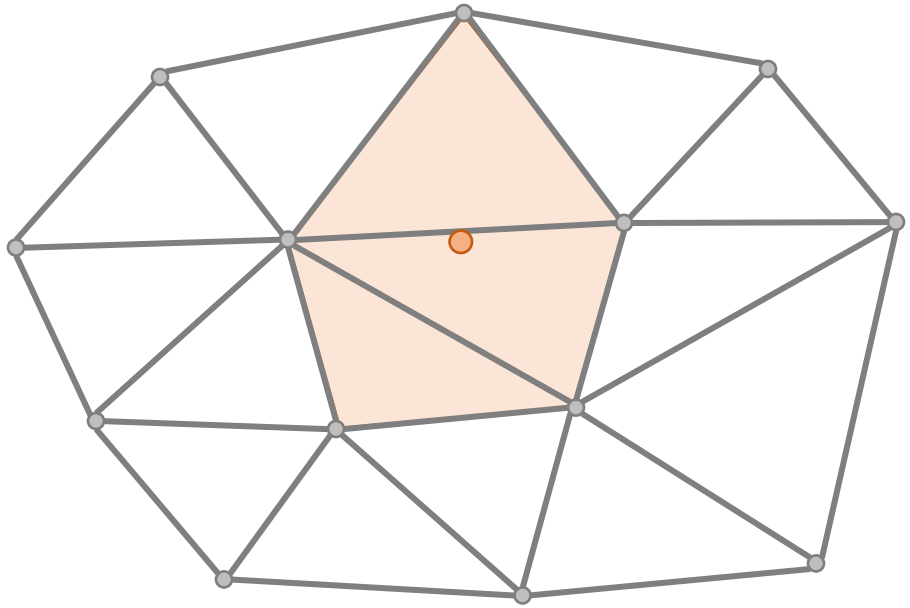
Delaunay triangulation (DT)

- No points are in the circumcircle of each triangle
- Getting practical parallel algorithms for DT is notoriously hard



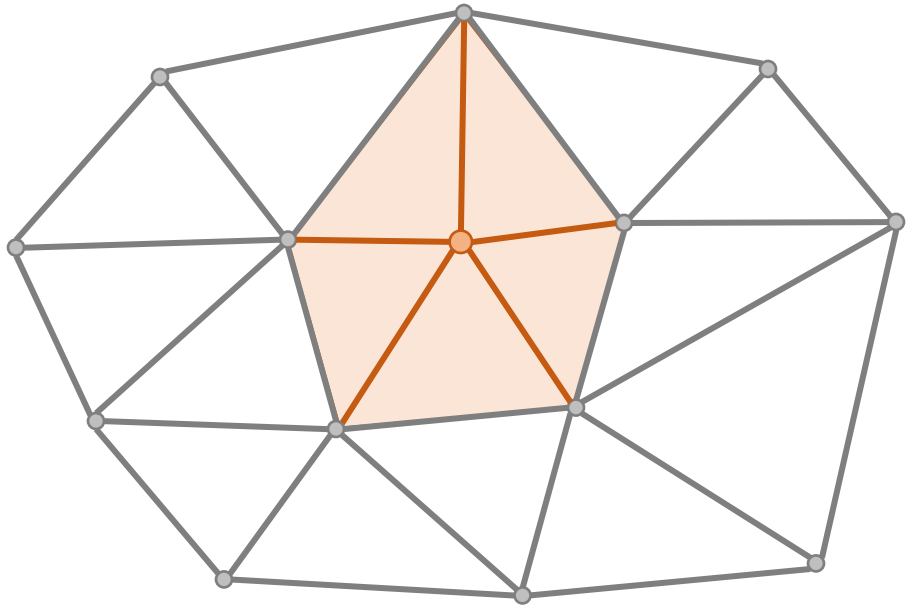
Delaunay triangulation (DT)

- **Again, consider the incremental construction**



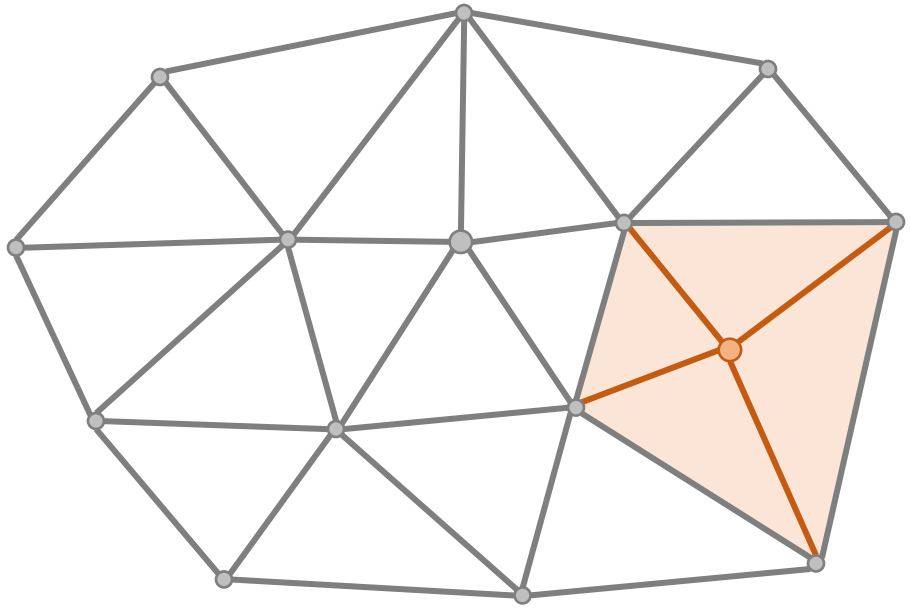
Delaunay triangulation (DT)

- **Again, consider the incremental construction**



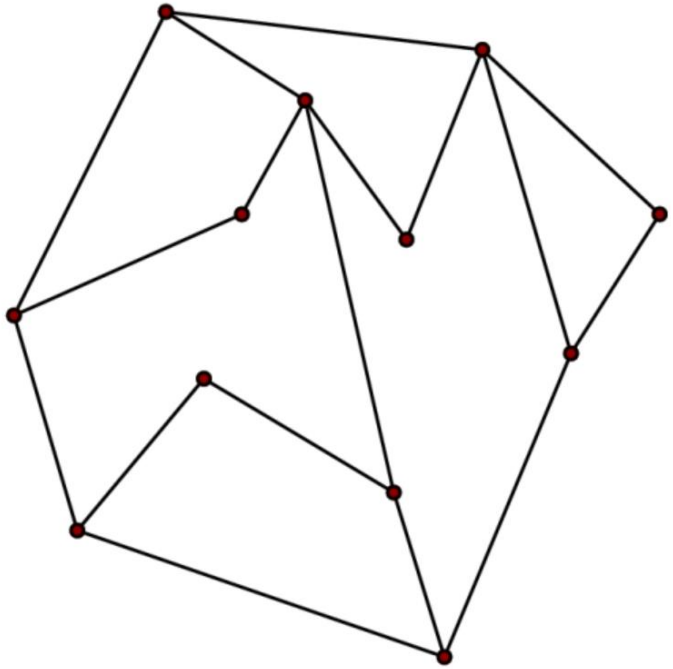
Delaunay triangulation (DT)

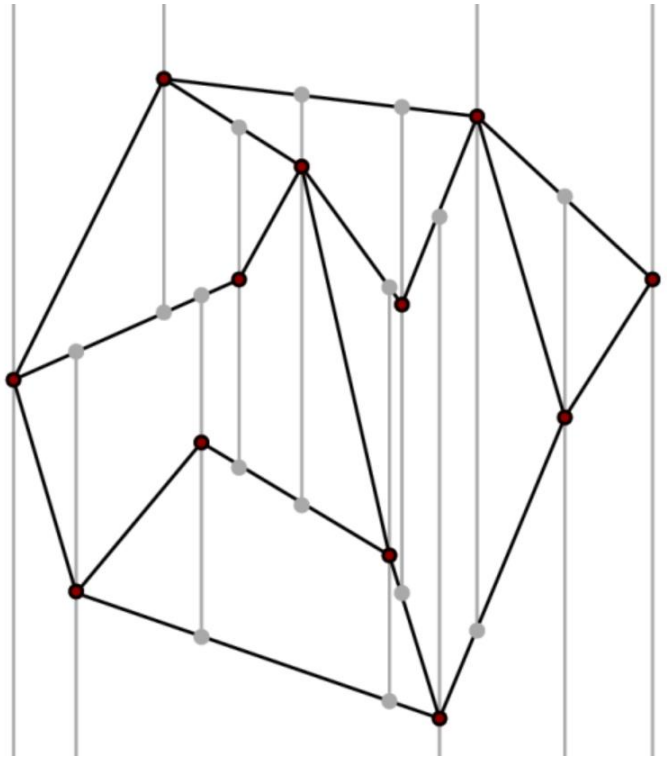
- Again, consider the incremental construction
- 2D parallel version is given in [BGSS SPAA 2016 / JACM 2020]
- Higher-D version is in [BGSS SPAA 2020], since k-D DT can be subsumed by $(k+1)$-D convex hull



Point location

- In a 2D plane, decide which polygon a query point belongs to
- Example solution: trapezoidal decomposition (open)





Range Searches

Example range searches

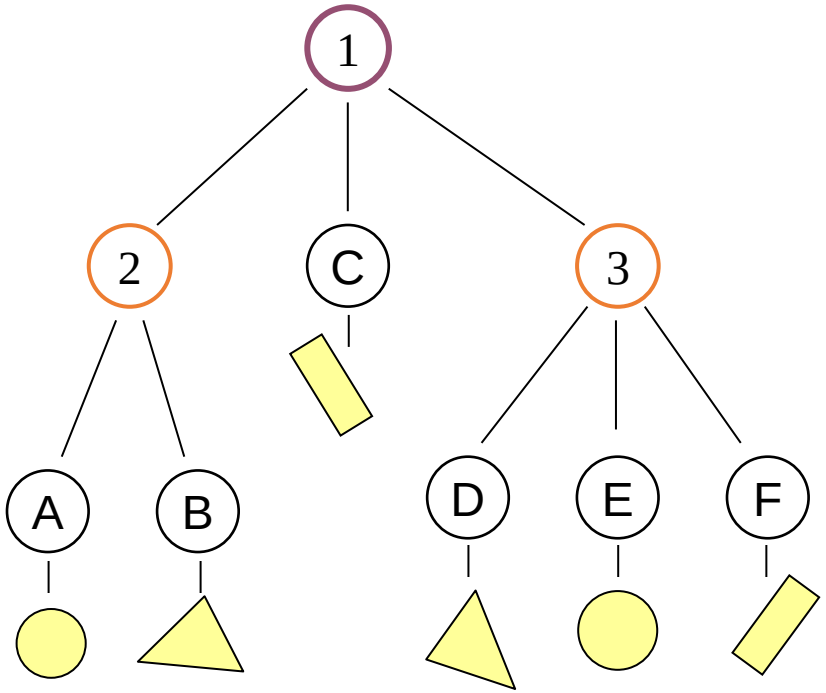
- Nearest neighbor search
- k-nearest neighbor (kNN) search
- Near neighbor counting
- Rectangular range search
- Rectangle queries
- Three-sided queries
- Segment (stabbing) queries
- Ray-scene intersection queries
- Collision detection
- etc.

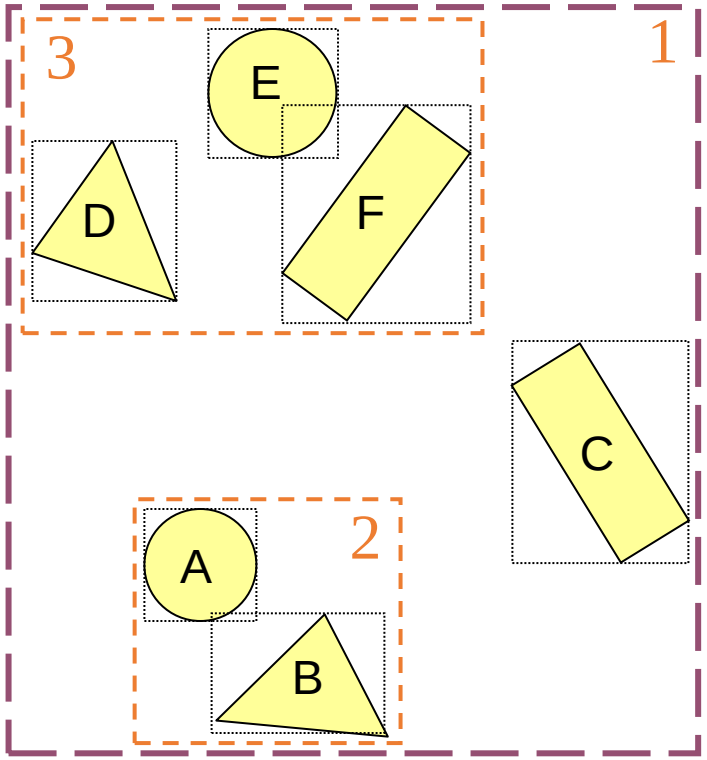
Building blocks in other algorithms / systems

Classic (sequential) data structures

- **Quad/octree**
- **k-d tree**
- **Interval tree**
- **Segment tree**
- **Range tree**
- **Priority tree**
- **Other augmented trees**
  - R-tree, bounding volume hierarchy (BVH)

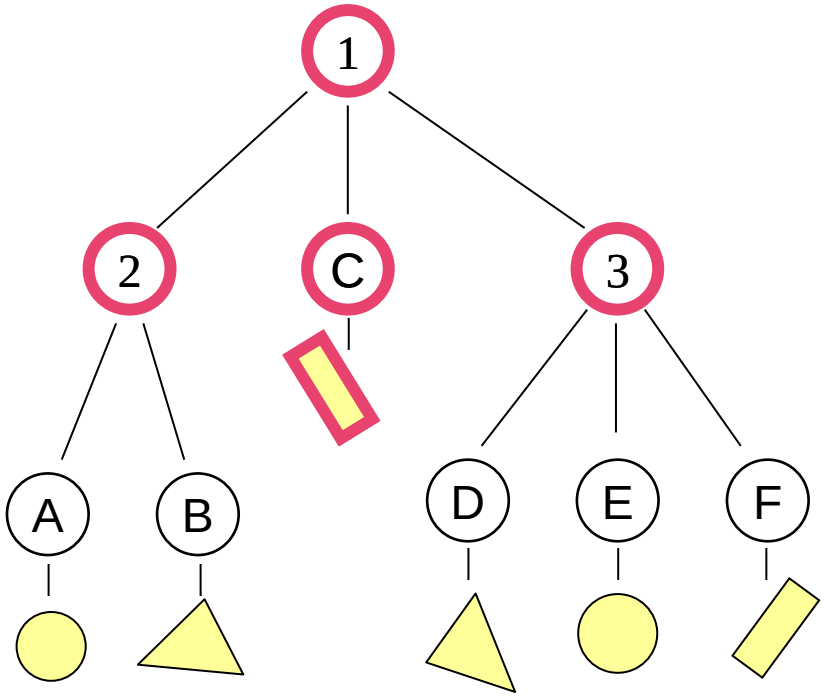
Why trees?

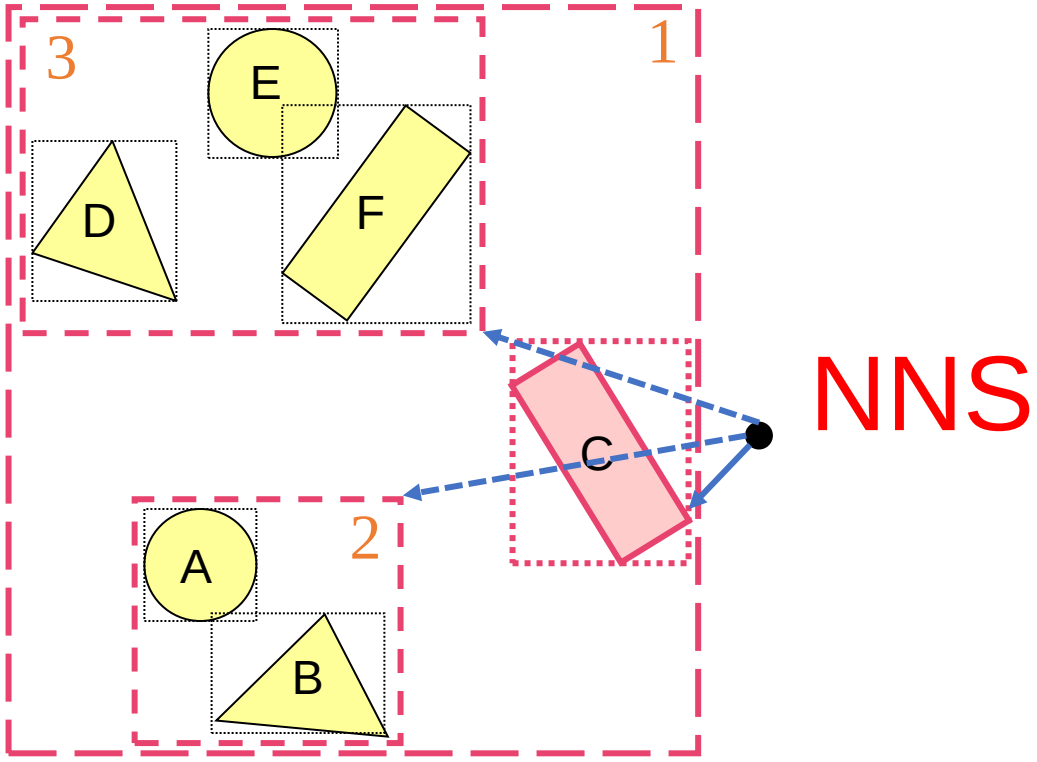




Each interior tree node acts as a fast-pass check for the subtree nodes

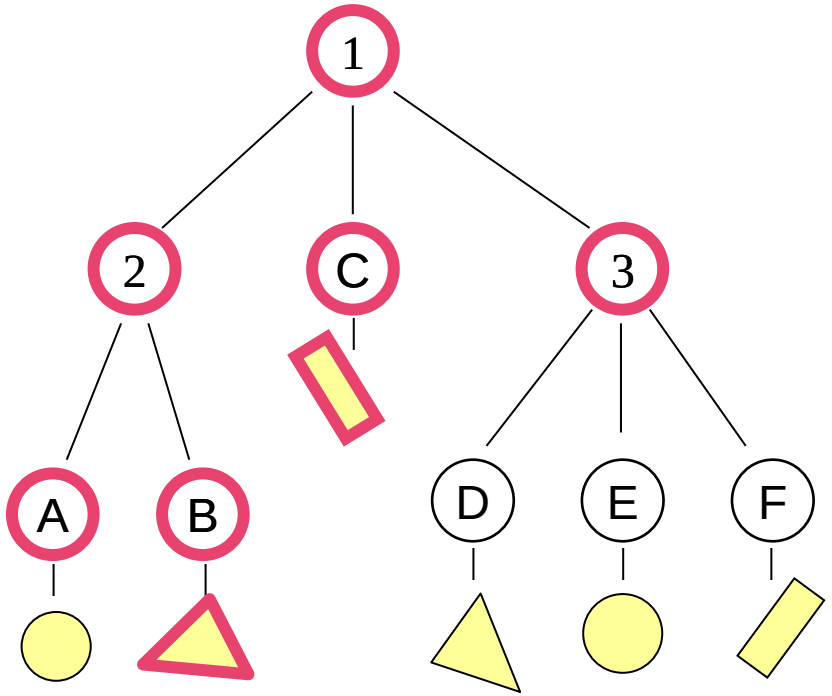
- **No need to traverse the subtree if the traversal can be pruned**

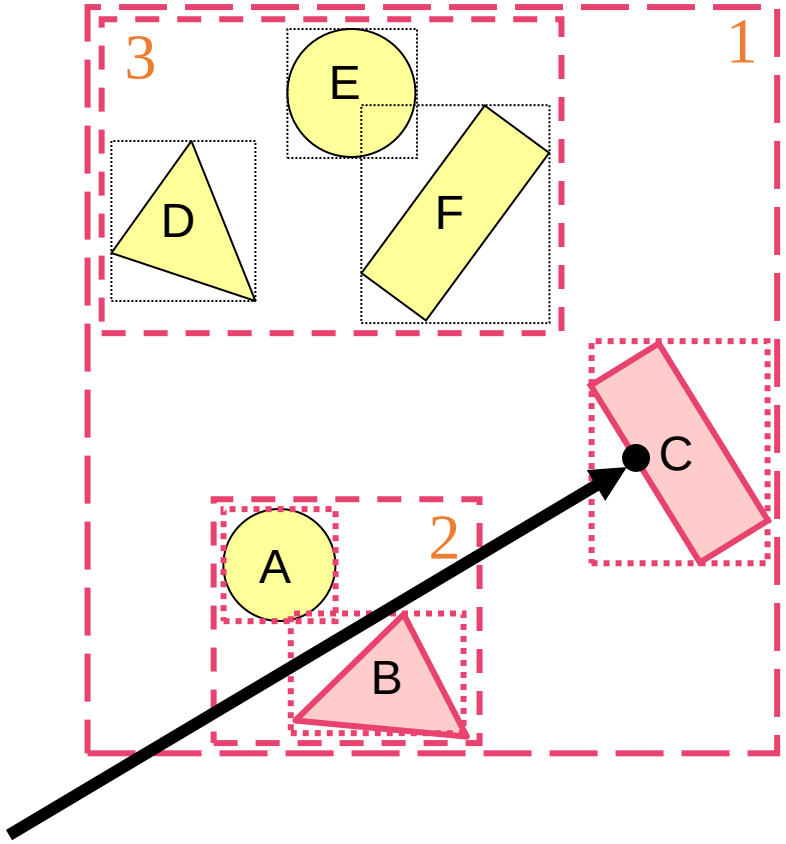




Each interior tree node acts as a fast-pass check for the subtree nodes

- No need to traverse the subtree if the query misses the bounding box





Example range searches

- Nearest neighbor search
- k-nearest neighbor (kNN) search
- Near neighbor counting
- Rectangular range search
- Rectangle queries
- Three-sided queries
- Segment (stabbing) queries
- Ray-scene intersection queries
- Collision detection
- etc.

Classic (sequential) data structures

- **Quad/octree**
- **k-d tree**
- **Interval tree**
- **Segment tree**
- **Range tree**
- **Priority tree**
- **Other augmented trees**
  - R-tree, bounding volume hierarchy (BVH)

“Modern” Geometry Problems

Problems that are less "classic" (from MIT 6.838)

- Clustering
- Range searches in "high" (>5) dimensions
- Low-distortion embeddings

- Geometric algorithms for modern architecture
- Geometric algorithms for streaming data

Course announcement

Mon 2/15	Presidents Day	Training 4 out
Wed 2/17	Geometric algorithms	
Mon 2/22	Graph algorithms	Project proposal due
Wed 2/24	Road ahead	
Mon 3/1	Training problems analysis (by students)	
Wed 3/3	Training problems analysis (by students)	
Mon 3/8	Project presentation 1	
Wed 3/10	Project presentation 2	

Score distribution for 142

- Four problem-solving assignments (20%)
- Three performance-engineering assignments (30%)
- Final project (20%)

- Quiz (5%)
- Final exam (25%)

- Class participation (10%, bonus)

Tentative score-to-grade mapping

- A+: very top performance in the class
- A: 85%
- A-: 80%
- B+: 75%
- B: 70%
- C: 65%
- D: 60%

Performance so far

- **More than 50% of you are doing very well**
- **Many of you are heading toward an A+**

- **However, a few of you have not done much**
- **You need to submit reports**
  - This is the **fifth** time I mentioned it in the class?
  - If you missed some before, you can send it to me via email (with 20% penalty)
  - The rule is the rule
- **This is an elective course**
  - For any reason you don't like it, you can drop it
  - If you fail, it's unlikely that you can make up (not sure about your next offering)

Project

Project

- Implement an algorithm that you like
  - Can be sequential or parallel
  - You should optimize the performance
  - You will write a report and give a short class presentation

Examples

- **Parallel algorithms:**
  - Parallel mergesort, parallel BFS, parallel hashtable, parallel median-finding, parallel LCS, and other challenging problems

- **Sequential algorithms:**
  - BSTs (AVL tree, red-black tree, treap), Prim and Dijkstra with efficient priority queue, I/O-efficient dynamic programming, convex hull, nearest point, Rabin-Karp / KMP, other non-trivial graph algorithms

- **Don’t make it too hard or too easy**

Rules

- **Up to two in a group, but less recommended**
  - Unless you have a clear plan on how to distribute the work evenly
- **Before next Wednesday (2/24), submit a brief proposal about:**
  - What you want to do, what algorithm you want to implement, how you are going to evaluate it
  - I will review it and give you optional feedback (like too easy or too hard for you)
- **You will give an 8-min class presentation on March 8 and 10 (25%)**
  - Others can attend optionally
- **Final report due March 12 (75%)**

Training problems analysis

Mon 2/15	Presidents Day	Training 4 out
Wed 2/17	Geometric algorithms	
Mon 2/22	Graph algorithms	Project proposal due
Wed 2/24	Road ahead	
Mon 3/1	Training problems analysis (by students)	
Wed 3/3	Training problems analysis (by students)	
Mon 3/8	Project presentation 1	
Wed 3/10	Project presentation 2	

What is that?

- **There are in total 11 problems (A in Training 1, A and C in Training 3 are not included)**
- **One of you will give a 10-minute presentation about one problem**
  - What the problem is
  - What your solution is and why it is correct
  - How to program and what optimization you use
  - If there are other interesting solutions
  - (You need to have solved that problem already)
- **You will get 3-5 bonus candies**
  - 3 candies by default
  - Up to 2 bonus candies for good presentations (talk quality and problem difficulty)

The process

- **Reservation link**

- **Then, I will run a bipartite-graph matching and assign you to a problem**

- **Q&As are welcome, and I will also be part of that**

Training 1			
B	Maximum Identity Matrix		
C	Go Straight		
Training 2			
A	Ski	Yan	Yihan
B	Counting Candies	Yihan	
C	Sort the train		
Training 3			
B	Symmetry Makes Perfect		
D	More Office Hours		
Training 4			
A	Selling candies	Yan	Xiaojun
B	Easy sorting	Yan	
C	Share candies		
D	Swans	Xiaojun	