# Lecture 7: Computational Geometry

## CS 491 CAP

Uttam Thakore

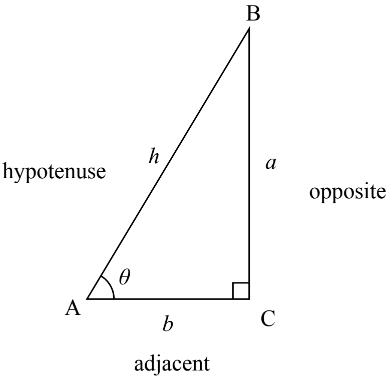
Friday, October 7$^{th}$, 2016

Credit for many of the slides on solving geometry problems goes to the Stanford CS 97SI course lecture on computational geometry (http://web.stanford.edu/class/cs97si/09-computational-geometry.pdf)

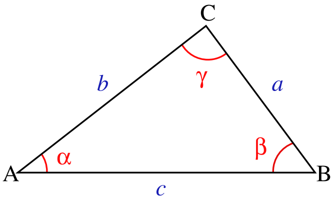
# Table of Contents

# sin, cos, & tan

- $\sin\theta = \frac{\text{opposite}}{\text{hypotenuse}}$
- $\cos\theta = \frac{\text{adjacent}}{\text{hypotenuse}}$
- $\tan\theta = \frac{\text{opposite}}{\text{adjacent}}$

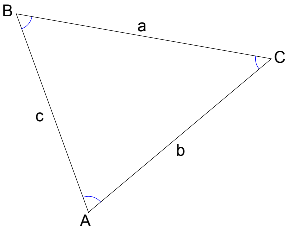


# Law of sines & law of cosines




- Law of sines: $\frac{\sin(\alpha)}{a} = \frac{\sin(\beta)}{b} = \frac{\sin(\gamma)}{c}$
- Law of cosines: $a^2 = b^2 + c^2 - 2bc\cos(\alpha)$
  - Put another way: $\cos(\alpha) = \frac{-a^2+b^2+c^2}{2bc}$
  - Similar for $b$ and $c$

# Heron's formula

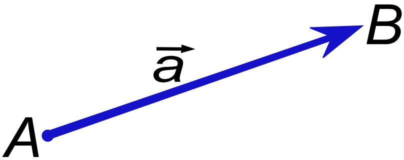



- Area of triangle $= \sqrt{s(s-a)(s-b)(s-c)}$
- $s =$ semiperimeter $= \frac{a+b+c}{2}$
- But there is an easier way to find the area of a triangle...

# Table of Contents

# What is a vector?

- Direction
- Magnitude
- Alternatively, magnitude & angle from positive $x$-axis

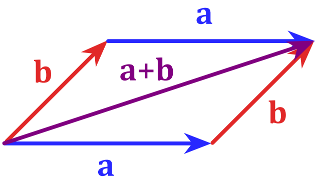


# Why are vectors important?

- Basis for large number of geometry problems
- Basic operations on vectors provide very useful information

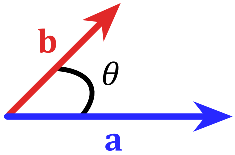
# Vector Addition

- Place beginning of one vector at end of other
- New vector from beginning of first vector to end of last



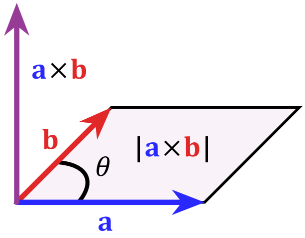
# Vector Dot Product

- $\mathbf{A} \cdot \mathbf{B} = x_1 \times x_2 + y_1 \times y_2$
- Why is this important?
  - $\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}||\mathbf{B}| \cos\theta$
  - Can use to find $\cos\theta$



# Vector Cross Product

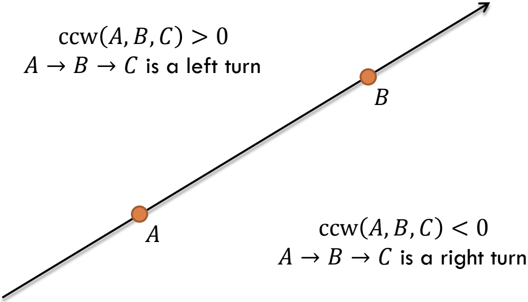
- $\mathbf{A} \times \mathbf{B} = x_1 \times y_2 - y_1 \times x_2$
- Why is this important?
  - $\mathbf{A} \times \mathbf{B} = |\mathbf{A}||\mathbf{B}| \sin\theta$
  - Can use to find $\sin\theta$
  - Significance:
    - For $\theta < 0$, $\sin\theta < 0$
    - For $\theta > 0$, $\sin\theta > 0$
  - Also, $\mathbf{A} \times \mathbf{B} =$ area of parallelogram created by $\mathbf{A}$ and $\mathbf{B}$
  - $\frac{\mathbf{A} \times \mathbf{B}}{2}$ is the area of the triangle



# Table of Contents

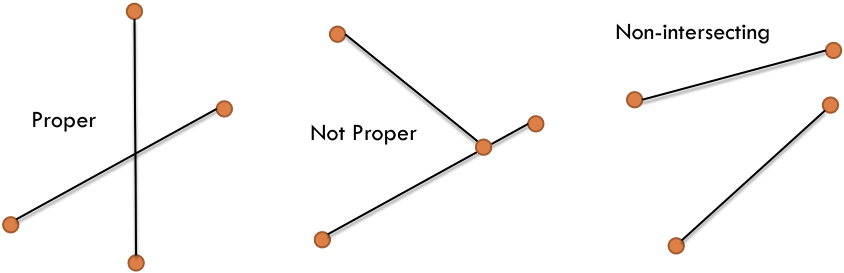
# Cross Product

- Define $\mathrm{ccw}(A, B, C) = (B - A) \times (C - A)$



# Segment-Segment Intersection Test

- Given two segments $AB$ and $CD$
- Want to determine if they intersect properly: two segments meet at a single point that are strictly inside both segments



# Segment-Segment Intersection Test

- Assume that the segments intersect
  - From $A$'s point of view, looking straight to $B$, $C$ and $D$ must lie on different sides
  - Holds true for the other segment as well
- The intersection exists and is proper if:
  - $\mathrm{ccw}(A, B, C) \times \mathrm{ccw}(A, B, D) < 0$
  - AND $\mathrm{ccw}(C, D, A) \times \mathrm{ccw}(C, D, B) < 0$

# Segment-Segment Intersection Test

- Determining non-proper intersections
  - We need more special cases to consider!
  - e.g. If $\mathrm{ccw}(A, B, C)$, $\mathrm{ccw}(A, B, D)$, $\mathrm{ccw}(C, D, A)$, $\mathrm{ccw}(C, D, B)$ are all zeros, then two segments are collinear
  - Very careful implementation is required

# Table of Contents

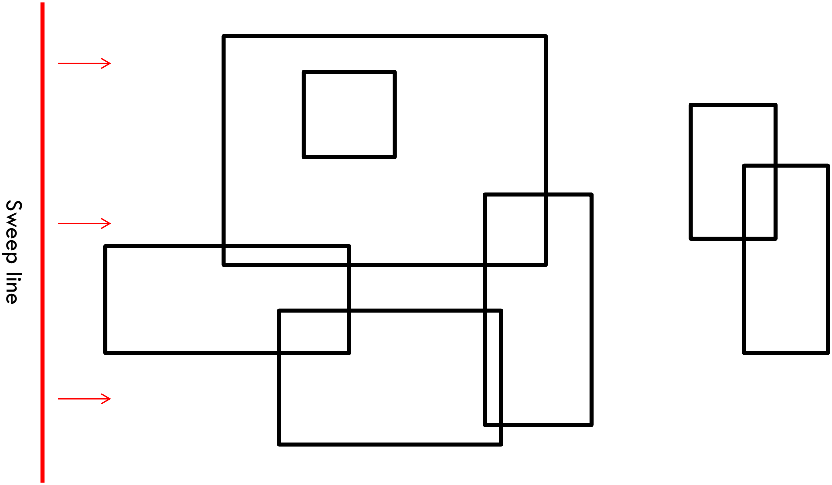
# Sweep Line Algorithm

- A problem solving strategy for geometry problems
- The main idea is to maintain a line (with some auxiliary data structure) that sweeps through the entire plane and solve the problem locally
- We can’t simulate a continuous process, (e.g. sweeping a line) so we define *events* that causes certain changes in our data structure
  - And process the events in the order of occurrence
- We’ll cover one sweep line algorithm

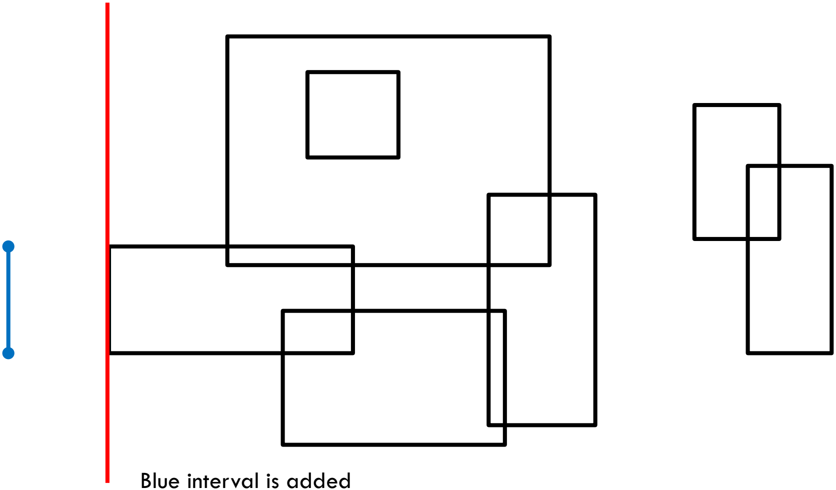
# Sweep Line Algorithm

- Problem: Given $n$ axis-aligned rectangles, find the area of the union of them
- We will sweep the plane from left to right
- Events: left and right edges of the rectangles
- The main idea is to maintain the set of "active" rectangles in order
  - It suffices to store the $y$-coordinates of the rectangles

# Example

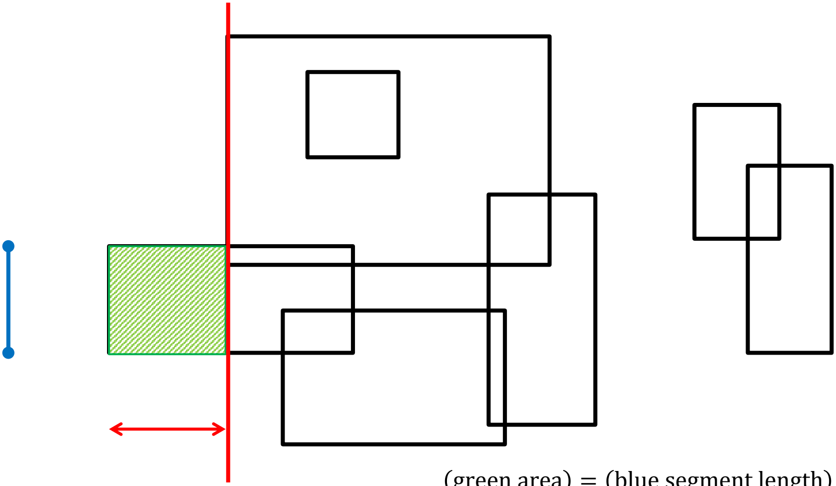


# Example



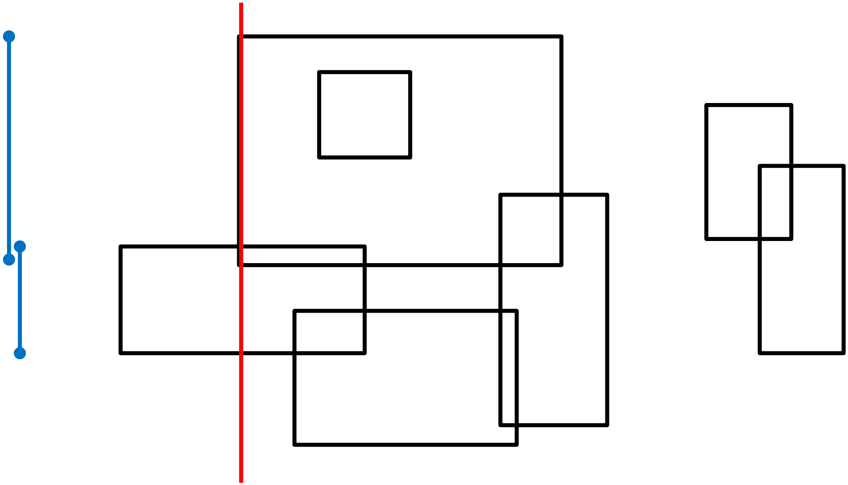
Blue interval is added
to the data structure

# Example

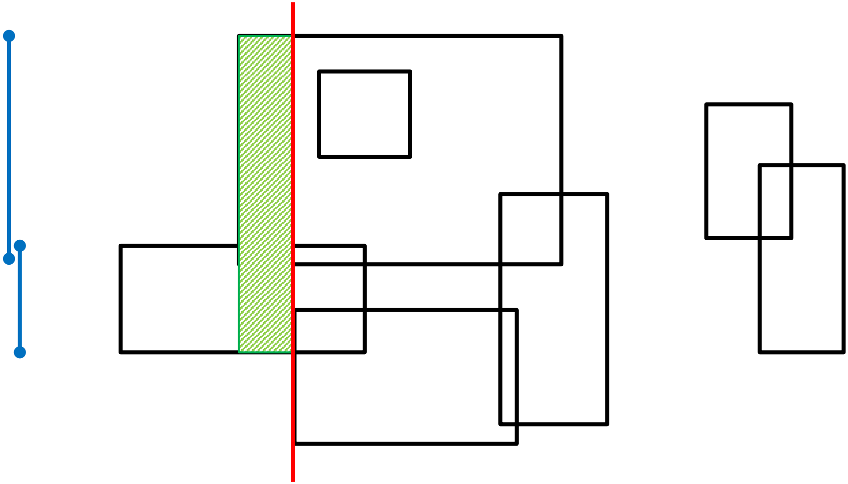


(green area) = (blue segment length)
× (distance that the sweep line traveled)

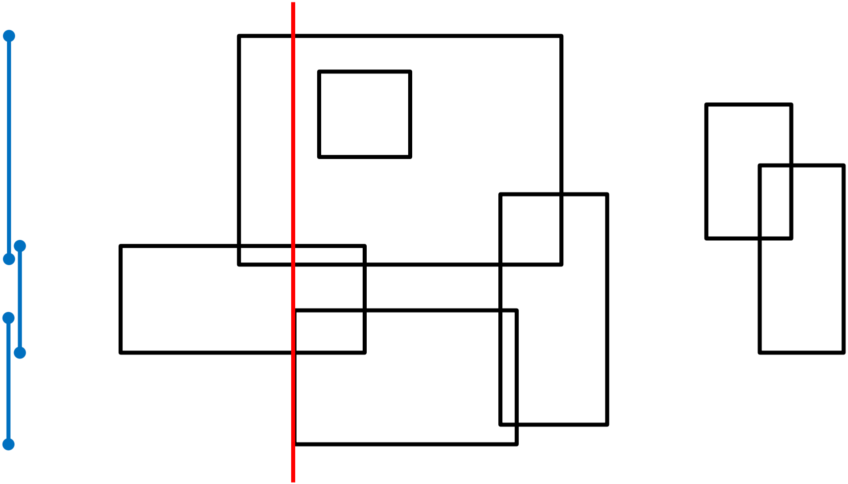
# Example



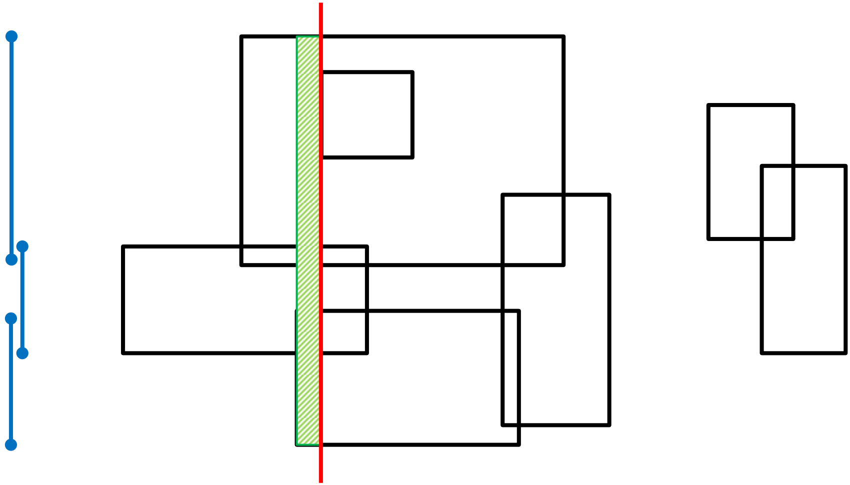
# Example



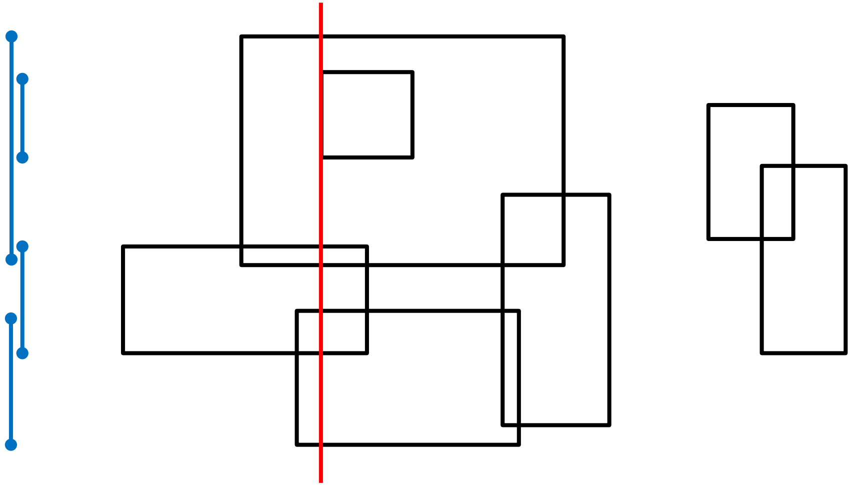
# Example



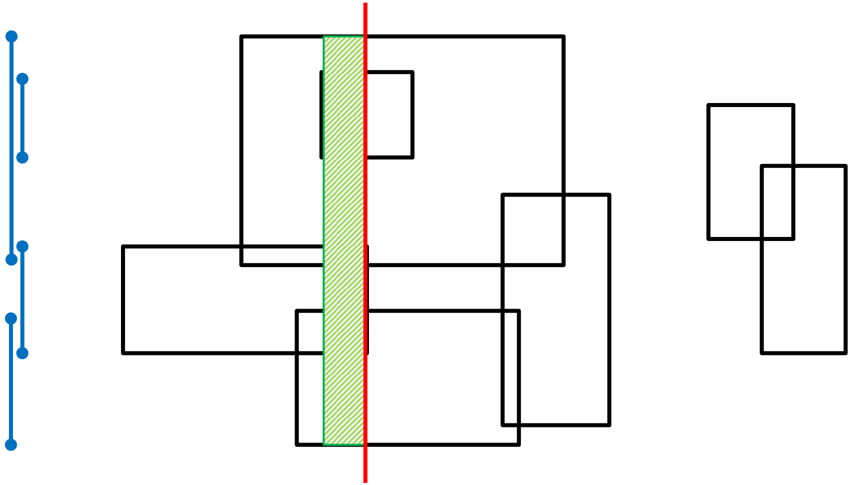
# Example



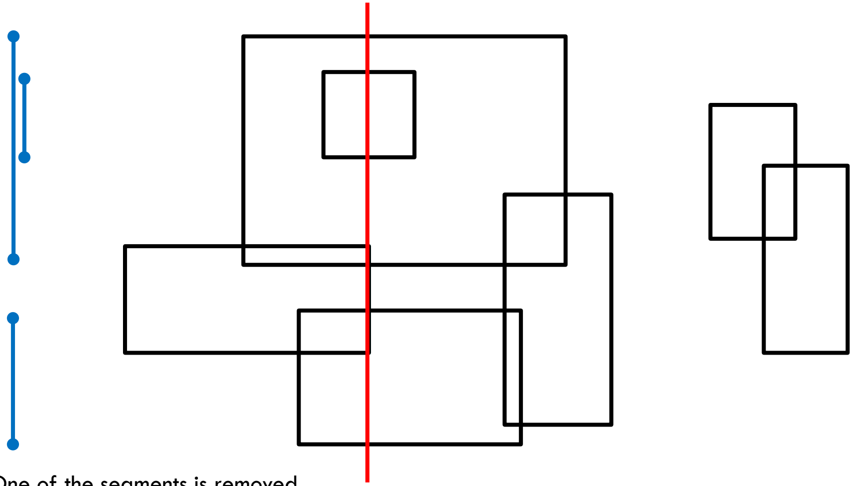
# Example



# Example

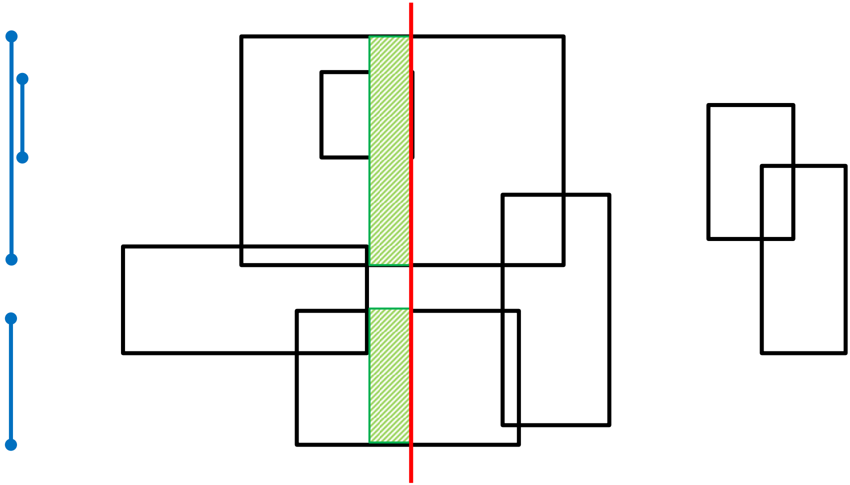


# Example



One of the segments is removed

# Example



# Pseudopseudocode

- If the sweep line hits the left edge of a rectangle
  - Insert it to the data structure
- Right edge?
  - Remove it
- Move to the next event, and add the area(s) of the green rectangle(s)
  - Finding the length of the union of the blue segments is the hardest step
  - There is an easy $O(n)$ method for this step

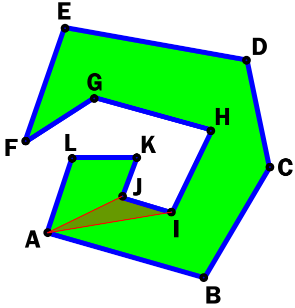
# Notes on Sweep Line Algorithms

- Sweep line algorithm is a generic concept
  - Come up with the right set of events and data structures for each problem
- Exercise problems
  - Finding the perimeter of the union of rectangles
  - Finding all $k$ intersections of $n$ line segments in $O((n+k)\log n)$ time

# Table of Contents

# Triangulation!

1. Start at any point (let's call it $A$)
2. Go through all other points, summing areas of triangles made by vectors from $A$ (using cross product)



# Why does this work?

- For convex polygons, obvious
- For concave polygons, when we turn inwards, area is negative
- Sum of the negative and positive triangles equals area of polygon

# Table of Contents

# Note on Binary Search

- Usually, binary search is used to find an item of interest in a sorted array

- There is a nice application of binary search, often used in geometry problems
  - Example: finding the largest circle that fits into a given polygon
    - Don't try to find a closed form solution or anything like that!
    - Instead, binary search on the answer

# Table of Contents

# Ternary Search

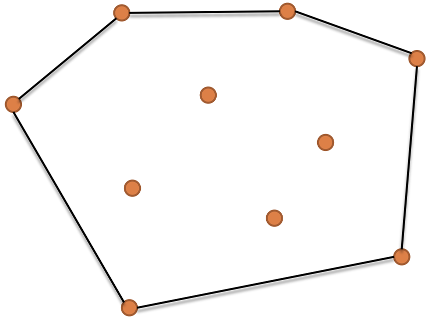
- Another useful method in many geometry problems
- Finds the minimum point of a "convex" function $f$
  - Not exactly convex, but let's use this word anyway
- Initialize the search interval $[s, e]$
- Until $e - s$ becomes small:
  - $m_1 = s + (e - s)/3, m_2 = e - (e - s)/3$
  - If $f(m_1) \le f(m_2)$, then set $e$ to $m_2$
  - Otherwise, set $s$ to $m_1$

# Table of Contents

# Table of Contents

# Convex Hull Problem

- Given $n$ points on the plane, find the smallest convex polygon that contains all the given points
  - For simplicity, assume that no three points are collinear



# Simple $O(n^3)$ algorithm

- $AB$ is an edge of the convex hull iff $\mathrm{ccw}(A, B, C)$ have the same sign for all other given points $C$
  - This gives us a simple algorithm

- For each $A$ and $B$:
  - If $\mathrm{ccw}(A, B, C) > 0$ for all $C \neq A, B$:
    - Record the edge $A \to B$
- Walk along the recorded edges to recover the convex hull

# Faster Algorithm: Graham Scan

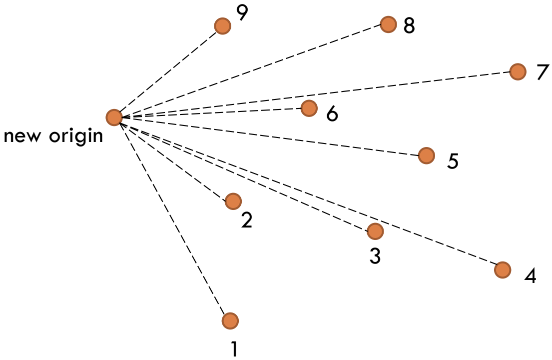
- We know that the leftmost given point has to be in the convex hull
  - We assume that there is a unique leftmost point
- Make the leftmost point the origin
  - So that all other points have positive $x$ coordinates
- Sort the points in increasing order of $y/x$
  - Increasing order of angle, whatever you like to call it
- Incrementally construct the convex hull using a stack

# Incremental Construction

- We maintain a *convex chain* of the given points
- For each $i$, we do the following:
  - Append point $i$ to the current chain
  - If the new point causes a concave corner, remove the bad vertex from the chain that causes it
  - Repeat until the new chain becomes convex

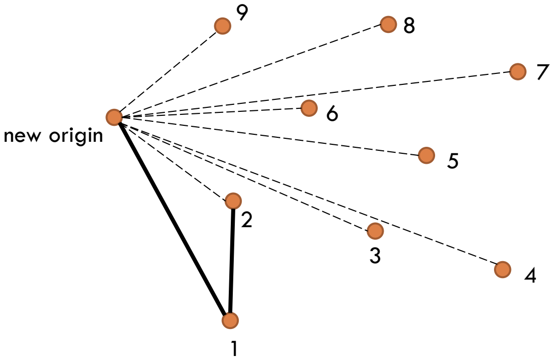
# Example

- Points are numbered in increasing order of $y/x$



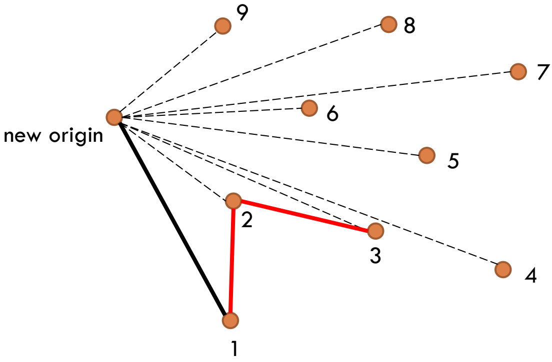
# Example

- Add the first two points in the chain



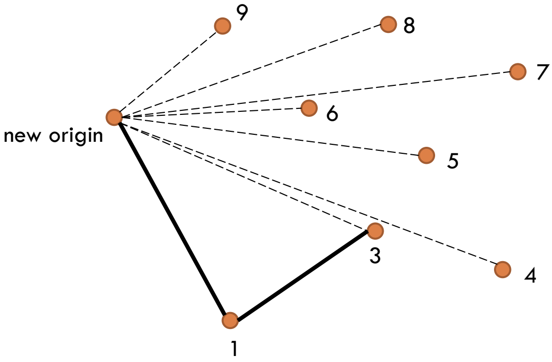
# Example

- Adding point 3 causes a concave corner 1-2-3
  - Remove 2



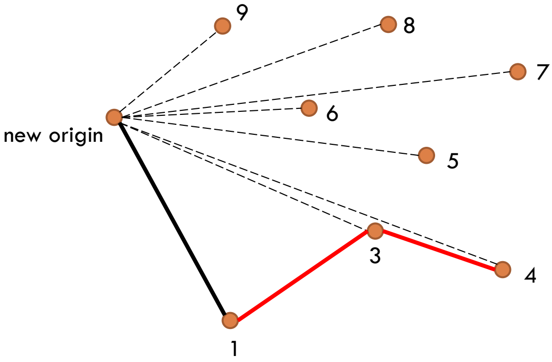
# Example

- That's better…



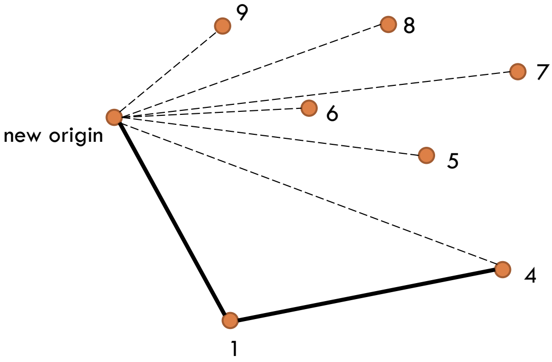
# Example

- Adding 4 to the chain causes a problem
  - Remove 3



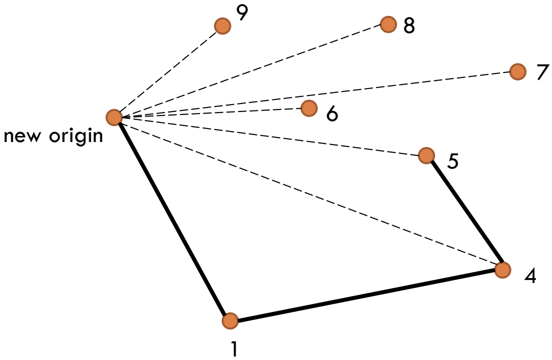
# Example

- Continue adding points…



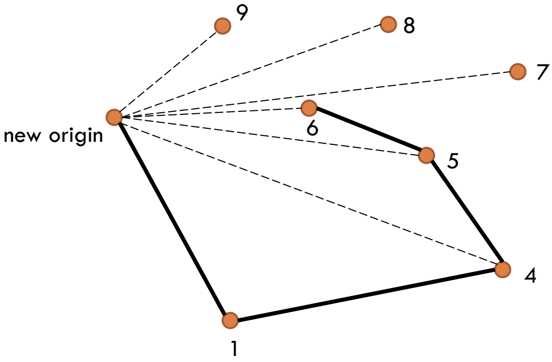
# Example

- Continue adding points…



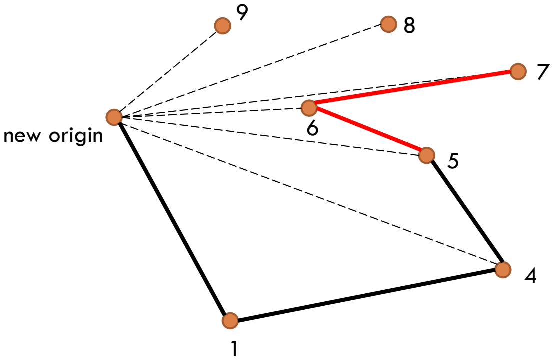
# Example

- Continue adding points…



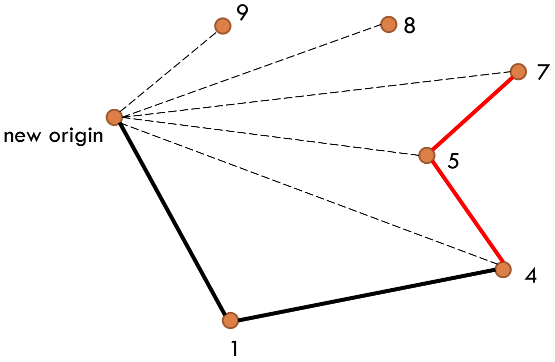
# Example

- Bad corner!



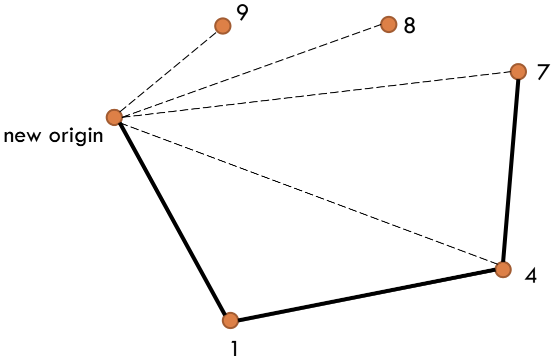
# Example

- Bad corner again!



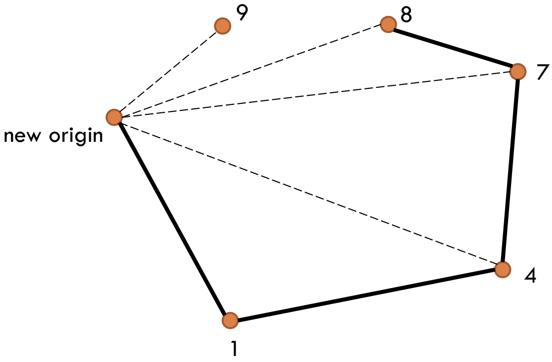
# Example

- Continue adding points…



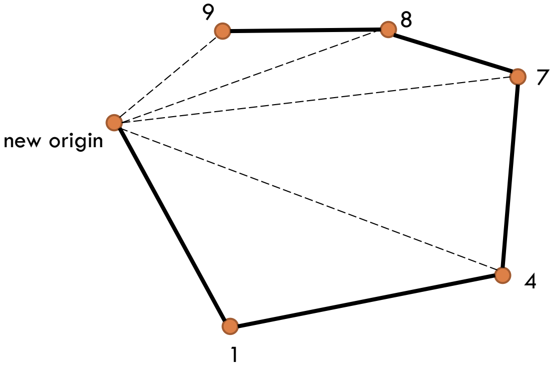
# Example

- Continue adding points…



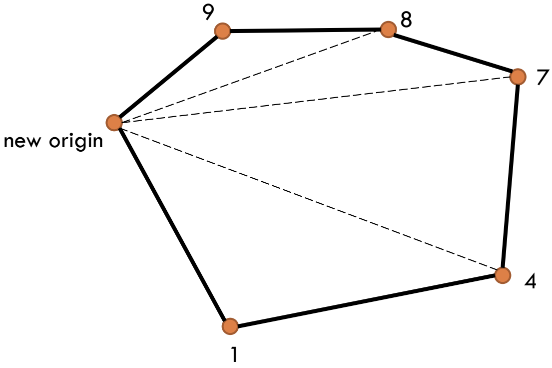
# Example

- Continue adding points…



# Example

- Done!



# Pseudocode

- Set the leftmost point $(0,0)$, and sort the rest of the points in increasing order of $y/x$
- Initialize stack $S$
- For $i = 1 \dots n$:
  - Let $A$ be the second topmost element of $S$, $B$ be the topmost element of $S$, $C$ be the $i$th point
  - If $\mathrm{ccw}(A, B, C) < 0$, pop $S$ and go back
  - Push $C$ to $S$
- Points in $S$ form the convex hull

# Graham Scan Complexity

- Sorting of points takes $O(n \log n)$ time
- Construction of the hull requires only traversal of all points once, with removal from the stack at most once, so $O(n)$ time
- Total runtime complexity: $O(n \log n)$

# Table of Contents

# There are more geometry concepts to know!

- Circle concepts – chord length, inscribed and circumscribed polygons, circles and intersecting tangent/secant lines
- Sphere concepts – great circle distance
- Polygons – checking if point is inside polygon (extension of concepts covered today)
- For formulas, algorithms, and implementations, see Chapter 7 of Competitive Programming by Steven Halim

Thankfully, most of these concepts do not appear in our regionals

- However, you should know these for World Finals!

# Table of Contents

## Some geometry resources

- TopCoder Algorithm Tutorials - Basic Geometry Concepts
- TopCoder Algorithm Tutorials - Line Intersection and its Applications
- TopCoder Algorithm Tutorials - Practice TopCoder geometry problems
- Ahmed-Aly list of geometry problems
- Stanford Introduction to ICPC Course
  - Geometry Lecture Slides
- Stanford ICPC Notebook (contains geometry algorithm implementations in C++)
- Chapter 7 of Competitive Programming by Steven Halim