

Shortest Path Algorithms

Jaehyun Park

CS 97SI
Stanford University

June 29, 2015

Outline

Cross Product

Convex Hull Problem

Sweep Line Algorithm

Intersecting Half-planes

Notes on Binary/Ternary Search

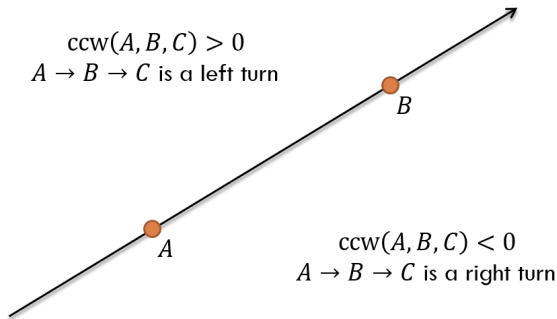
Cross Product

- ▶ Arguably the most important operation in 2D geometry
- ▶ We'll use it all the time

- ▶ Applications:
 - Determining the (signed) area of a triangle
 - Testing if three points are collinear
 - Determining the orientation of three points
 - Testing if two line segments intersect

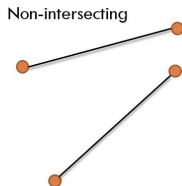
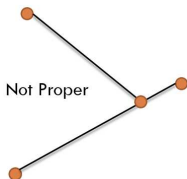
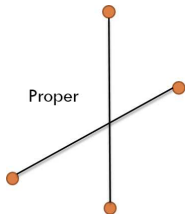
Cross Product

Define $ccw(A, B, C) = (B - A) \times (C - A) = (b_x - a_x)(c_y - a_y) - (b_y - a_y)(c_x - a_x)$



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:
 - $ccw(A, B, C) \times ccw(A, B, D) < 0$
 - *and* $ccw(C, D, A) \times ccw(C, D, B) < 0$

Non-proper Intersections

- ▶ We need more special cases to consider!
- ▶ e.g., If $ccw(A, B, C)$, $ccw(A, B, D)$, $ccw(C, D, A)$, $ccw(C, D, B)$ are all zeros, then two segments are collinear
- ▶ Very careful implementation is required

Outline

Cross Product

Convex Hull Problem

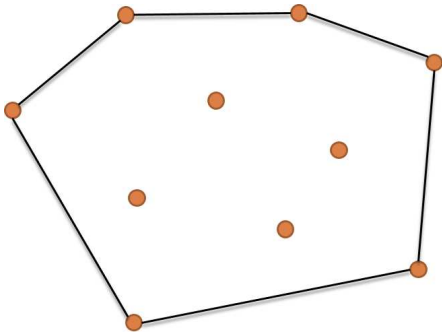
Sweep Line Algorithm

Intersecting Half-planes

Notes on Binary/Ternary Search

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 Algorithm

- ▶ AB is an edge of the convex hull iff $ccw(A, B, C)$ have the same sign for all other points C
 - This gives us a simple algorithm
- ▶ For each A and B :
 - If $ccw(A, B, C) > 0$ for all $C \neq A, B$:
 - ▶ Record the edge $A \rightarrow B$
- ▶ Walk along the recorded edges to recover the convex hull

Faster Algorithm: Graham Scan

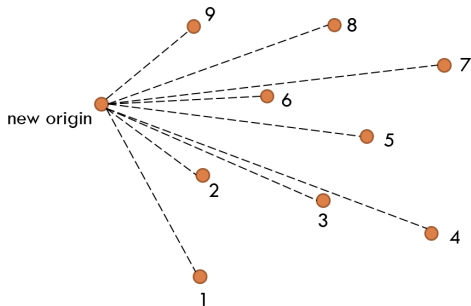
- ▶ We know that the leftmost given point has to be in the convex hull
 - We assume that there is a unique leftmost point
- ▶ Make the leftmost point the origin
 - So that all other points have positive x coordinates
- ▶ Sort the points in increasing order of y/x
 - Increasing order of angle, whatever 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 , 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

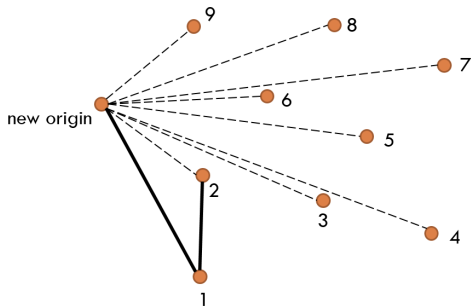
Example

Points are numbered in increasing order of y/x



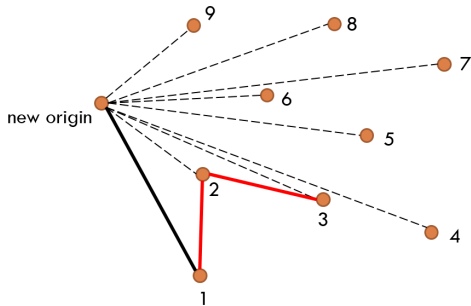
Example

Add the first two points in the chain



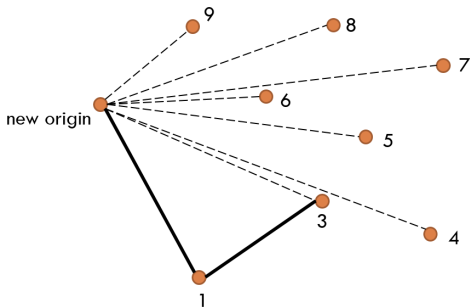
Example

Adding point 3 causes a concave corner 1-2-3: remove 2



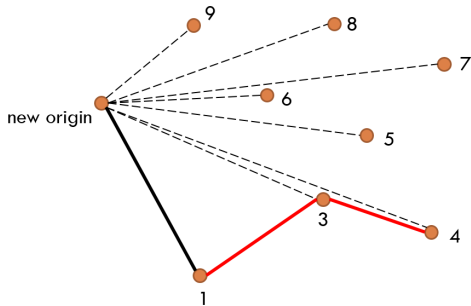
Example

That's better...



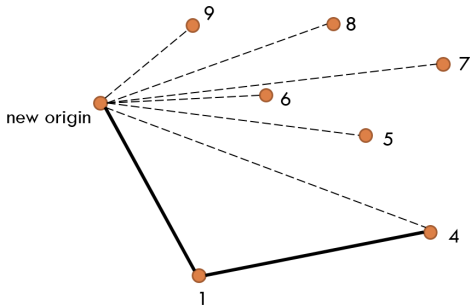
Example

Adding point 4 to the chain causes a problem: remove 3



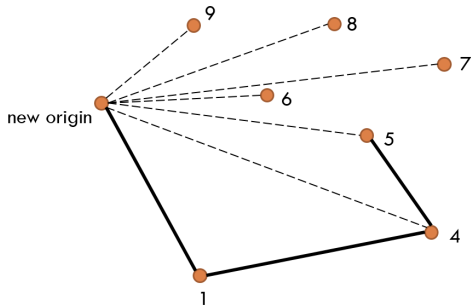
Example

Continue adding points...



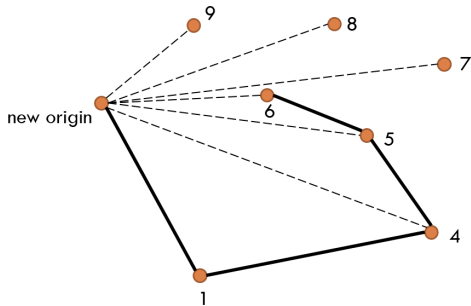
Example

Continue adding points...



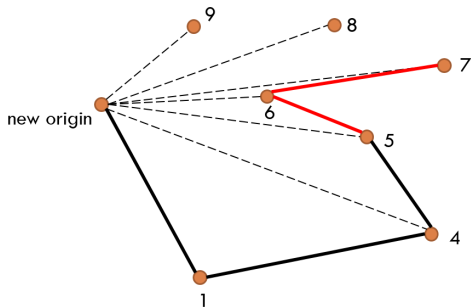
Example

Continue adding points...



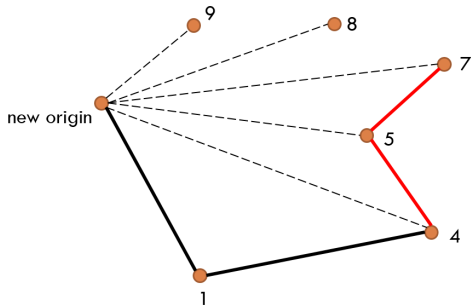
Example

Bad corner!



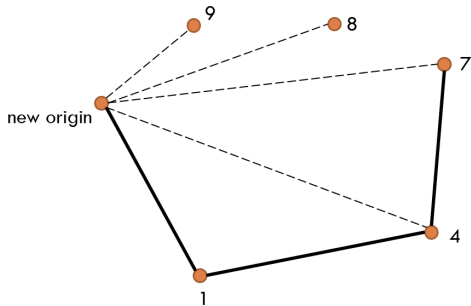
Example

Bad corner again!



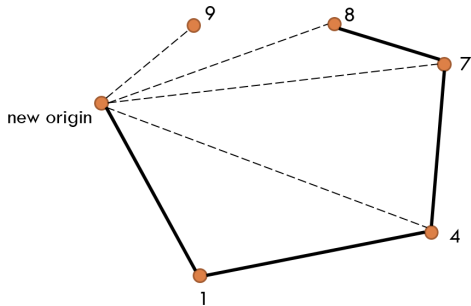
Example

Continue adding points...



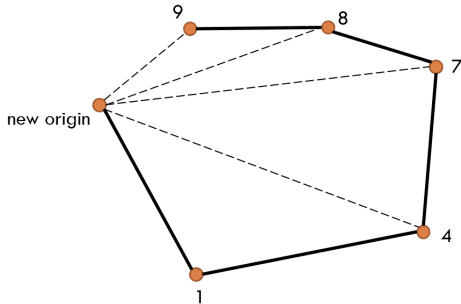
Example

Continue adding points...



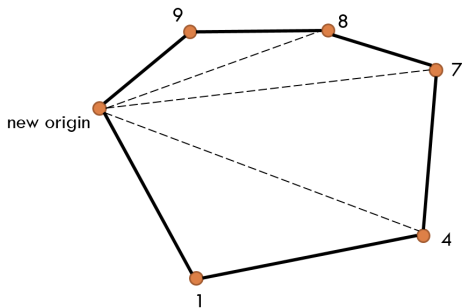
Example

Continue adding points...



Example

Done!



Pseudocode

- ▶ Set the leftmost point as $(0, 0)$, 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 , and C be the i th point
 - If $\text{ccw}(A, B, C) < 0$, pop S and go back
 - Push C to S
- ▶ Points in S form the convex hull

Outline

Cross Product

Convex Hull Problem

Sweep Line Algorithm

Intersecting Half-planes

Notes on Binary/Ternary Search

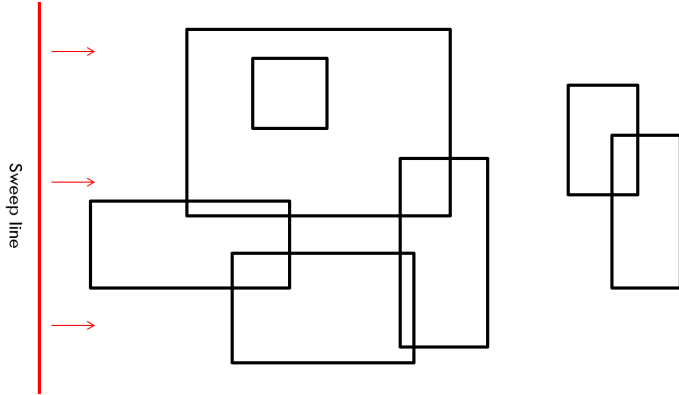
Sweep Line Algorithm

- ▶ 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

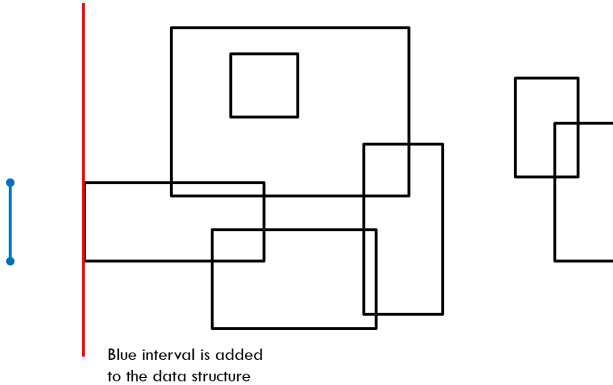
Sweep Line Algorithm

- ▶ 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

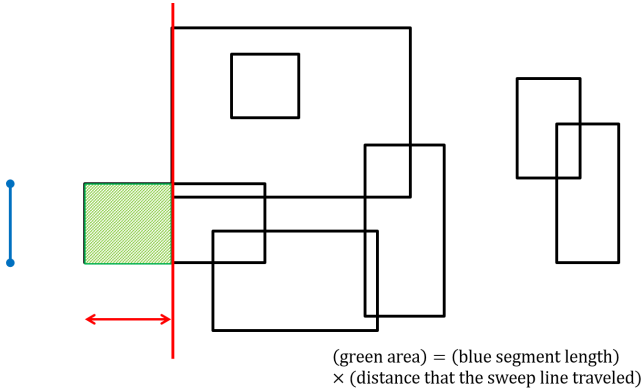
Example



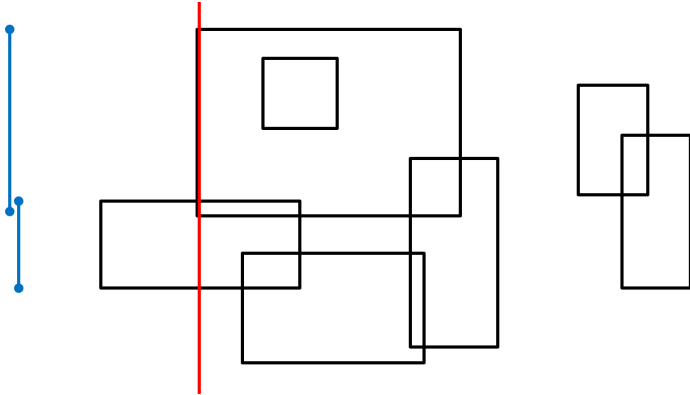
Example



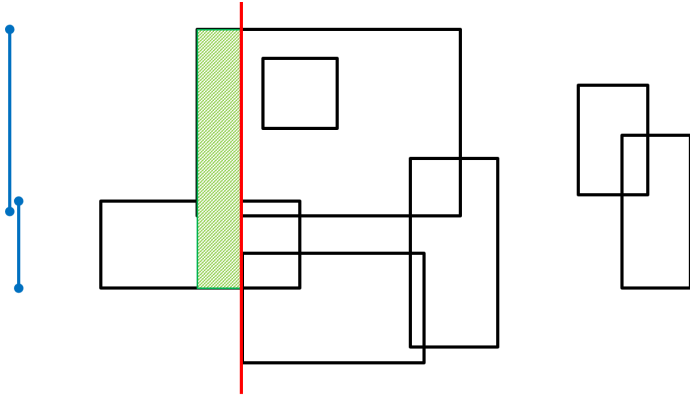
Example



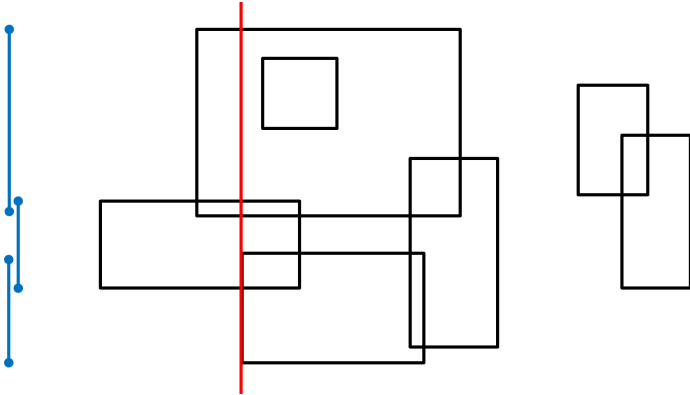
Example



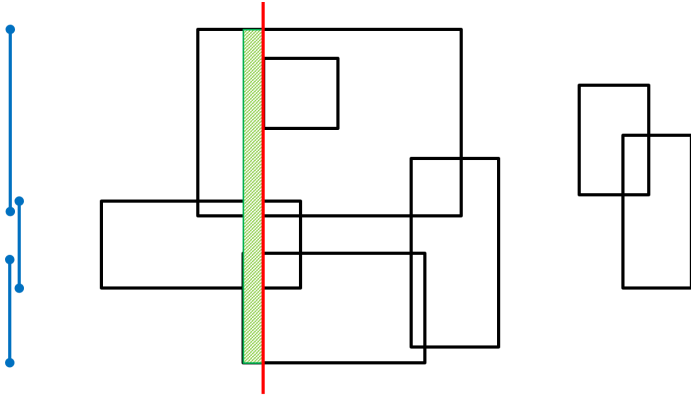
Example



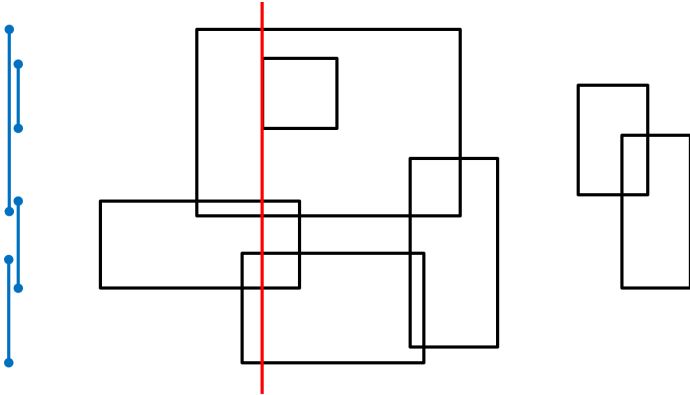
Example



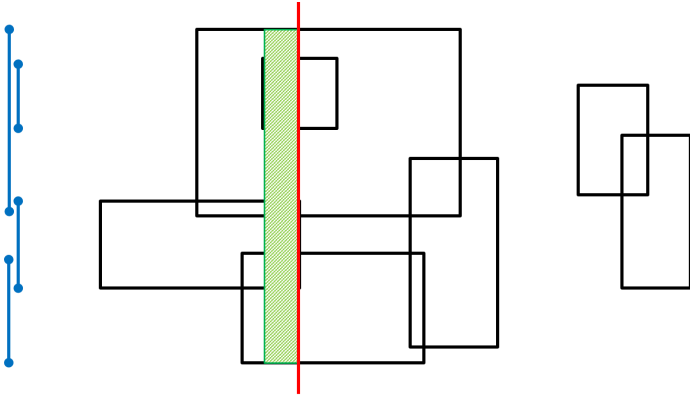
Example



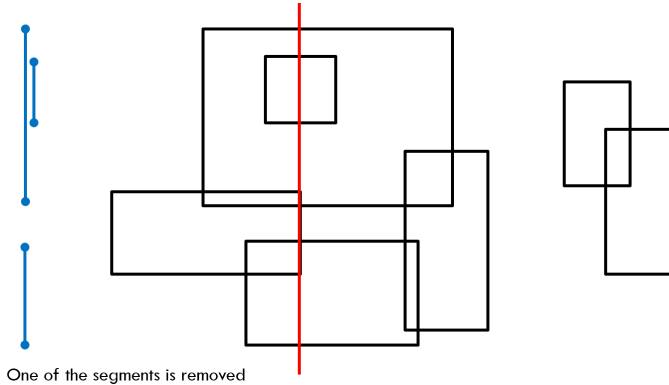
Example



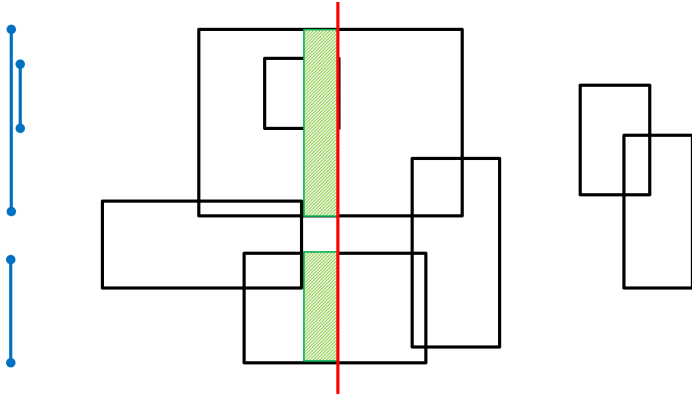
Example



Example



Example



Pseudo-pseudocode

- ▶ 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

Outline

Cross Product

Convex Hull Problem

Sweep Line Algorithm

Intersecting Half-planes

Notes on Binary/Ternary Search

Intersecting Half-planes

- ▶ Representing a half-plane: $ax + by + c \leq 0$
- ▶ The intersection of half-planes is a convex area
 - If the intersection is bounded, it gives a convex polygon
- ▶ Given n half-planes, how do we compute the intersection of them?
 - *i.e.*, Find vertices of the convex area
- ▶ There is an easy $O(n^3)$ algorithm and a hard $O(n \log n)$ one
 - We will cover the easy one

Intersecting Half-planes

- ▶ For each half-plane $a_i x + b_i y + c_i \leq 0$, define a straight line $e_i : a_i x + b_i y + c_i = 0$
- ▶ For each pair of e_i and e_j :
 - Compute their intersection $p = (p_x, p_y)$
 - Check if $a_k p_x + b_k p_y + c_k \leq 0$ for all half-planes
 - ▶ If so, store p in some array P
 - ▶ Otherwise, discard p
- ▶ Find the convex hull of the points in P

Intersecting Half-planes

- ▶ The intersection of half-planes can be unbounded
 - But usually, we are given limits on the min/max values of the coordinates
 - Add four half-planes $x \geq -M$, $x \leq M$, $y \geq -M$, $y \leq M$ (for large M) to ensure that the intersection is bounded
- ▶ Time complexity: $O(n^3)$
 - Pretty slow, but easy to code

Outline

Cross Product

Convex Hull Problem

Sweep Line Algorithm

Intersecting Half-planes

Notes on Binary/Ternary Search

Notes 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

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 enough”:
 - $m_1 := s + (e - s)/3, m_2 := e - (e - s)/3$
 - If $f(m_1) \leq f(m_2)$, set $e := m_2$
 - Otherwise, set $s := m_1$