CPSC 490 – Problem Solving in Computer Science

Lecture 17: Rotating Calipers

Jason Chiu and Raunak Kumar
Based on slides by Nasa Rouf (2014)

2017/03/03

University of British Columbia

Problem 1: Convex Hull Again

Find the two points that are farthest apart in Euclidean distance.

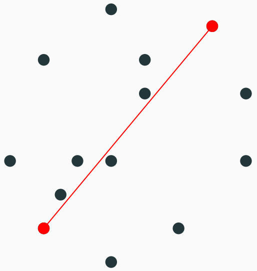


Figure 1: Finding the pair of points that are farthest apart

Problem 1: Possible Solutions

Observation: answer is on the convex hull.

Brute force: try all pairs of points on convex hull, $O(n^2)$

What about trying every possible end point and binary search for the other point?

Problem 1: Possible Solutions

Observation: answer is on the convex hull.

Brute force: try all pairs of points on convex hull, $O(n^2)$

What about trying every possible end point and binary search for the other point?

No! Adjacent points on hull could have same distance from your start point.

Problem 1: The Greedy Solution

Simple “2-pointer” greedy:

1. Initialize 2 pointers on two adjacent points on hull
2. Advance the second pointer as far as possible while distance increases
3. Advance the first pointer once and go back to 2
4. End once the first pointer has traveled full circle around hull

Time complexity: $O(n)$ after getting convex hull

Why does this work? When pointer 1 advances, can use geometry to prove that it is never optimal to move pointer 2 back.

Problem 2: Biggest Triangle

Find the three points that form biggest area triangle

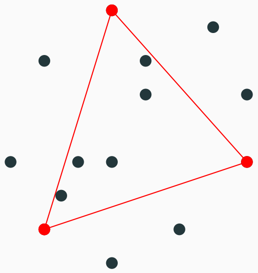


Figure 2: Forming triangle by choosing 3 points

Problem 2: Solution

Compute convex hull and do the same greedy!

1. Initialize 3 pointers to 3 adjacent vertices
2. While area increases, advance pointer 3
3. If area increases, advance pointer 2 once and go to #2
4. Advance pointer 1 once and go to #2
5. Stop when pointer 1 has gone full circle

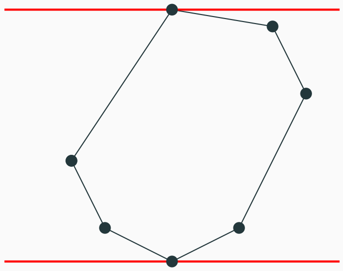
Time complexity: $O(n)$ again! (after getting convex hull)

In fact this method generalizes to any k-gon!

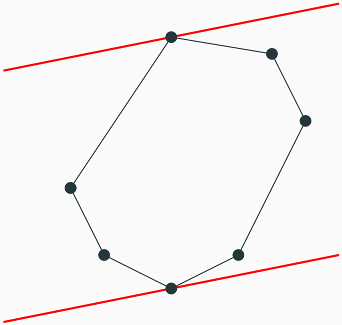
$\Rightarrow O(nk)$ time to find biggest area k-gon of a set of points, when $k \leq h =$ #points on convex hull; much faster than $O(n^3k)$ last class!

Unfortunately correctness quite difficult to prove. See reference.

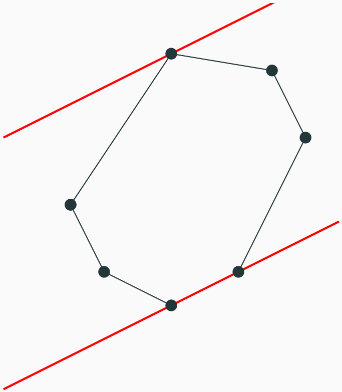
Rotating Calipers



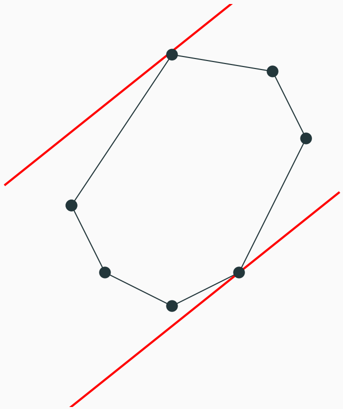
Rotating Calipers



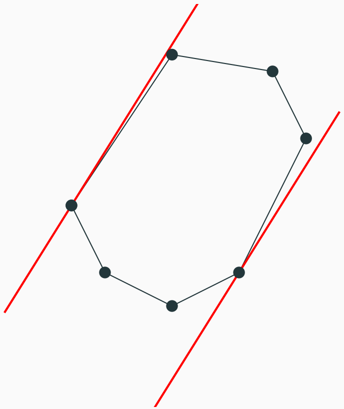
Rotating Calipers



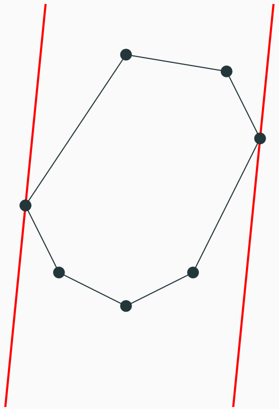
Rotating Calipers



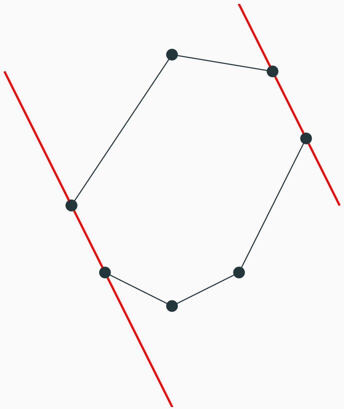
Rotating Calipers



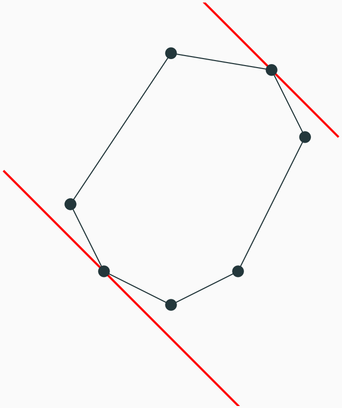
Rotating Calipers



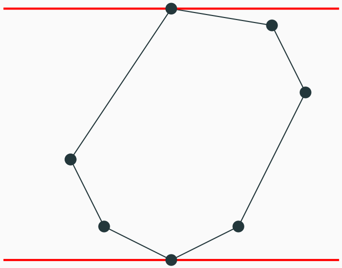
Rotating Calipers



Rotating Calipers



Rotating Calipers



Problem 3: Bounding Box

Find the smallest bounding box of these points. Rectangle could be rotated.

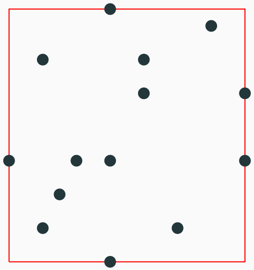


Figure 3: A possible (rectangular) bounding box

Problem 3: Solution

Observation: box must line up with one edge on the hull.

Why? Suppose not, then we can wiggle the rectangle without changing which vertices it touch. Now, how does the area change?

Notice rectangle vertex moves on a semi-circle $\Rightarrow$ "extra" area is concave-down function $\Rightarrow$ optimum is at extrema of movement.

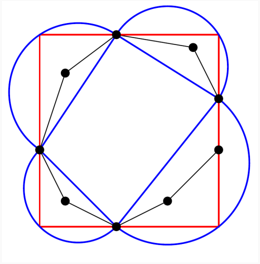


Figure 4: Rectangle vertices move along semicircles of convex hull chords

Problem 3: Solution

How to iterate through all rectangles quickly? Two pairs of calipers!

Algorithm: initialize 4 perpendicular calipers, then rotate them, events are when any caliper hits an edge. Compute rectangle area at every event and take the max.

Time complexity: every caliper has at most n events, so $O(n)$ after convex hull.

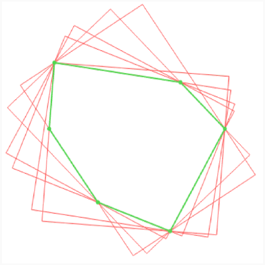


Figure 5: Iterating through bounding boxes

Problem 4: Merging Convex Hulls – Take Two

We learned how to merge two convex hull separated by dividing line. What about the general case?

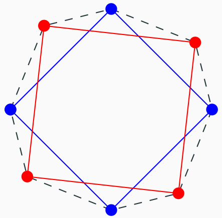


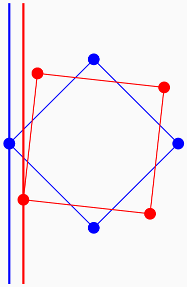
Figure 6: Merging two convex hulls

Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$

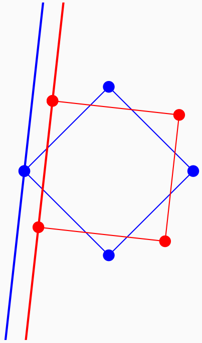


Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$

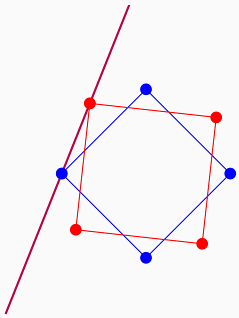


Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$

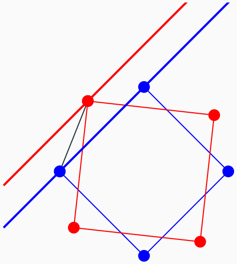


Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$

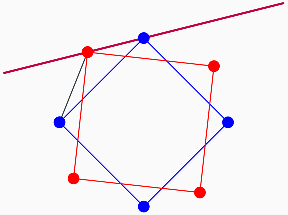


Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$

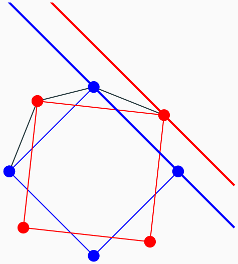


Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$

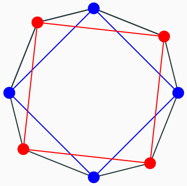


Problem 4: Solution

Initialize 2 calipers pointing up at left most point of each hull. Events when any caliper hits edge. Add edge when

- Calipers switch relative order
- Outer caliper hits an edge

Time complexity: $O(n)$



Problem 5: Non-parallel Calipers

Which point should we add to the convex hull to make it biggest?

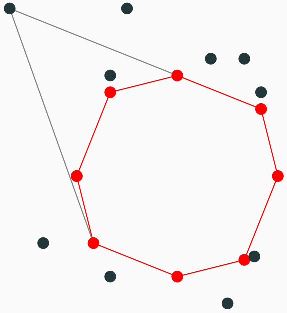


Figure 7: Picking a point to enlarge convex hull

Problem 5: Solution

Observation: point must be on “big hull” of all points.

- Pick arbitrary starting point on big hull, draw two tangent lines to small hull as initial calipers
- Rotate calipers around such that their intersection sequentially go through all points on the big hull.

Time complexity: $O(n)$ after convex hull

Problem 6: Minkowski Sum

How do we navigate a (convex polygon shaped) robot around convex polygon obstacles?

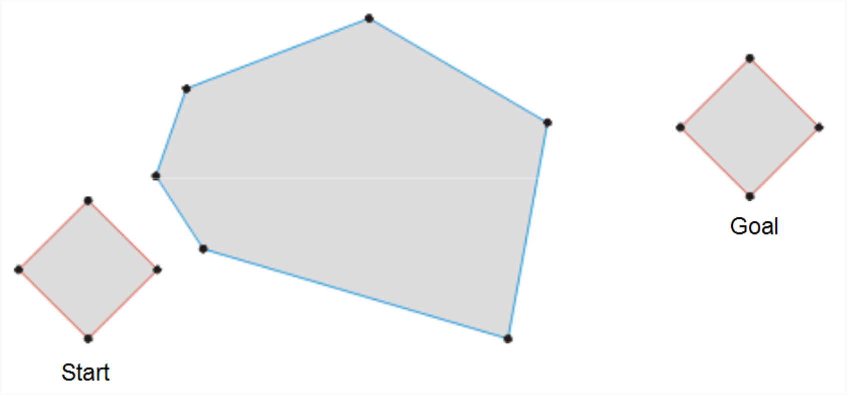


Figure 8: Navigating a convex robot around convex obstacle

Problem 6: Minkowski Sum – Solution

The problem would be a lot easier if we enlarge our obstacles and shrink the robot to a single point. How? Just “add” the robot and the obstacle together!

Actually very easy – just merge all the edges and sort them by angle!

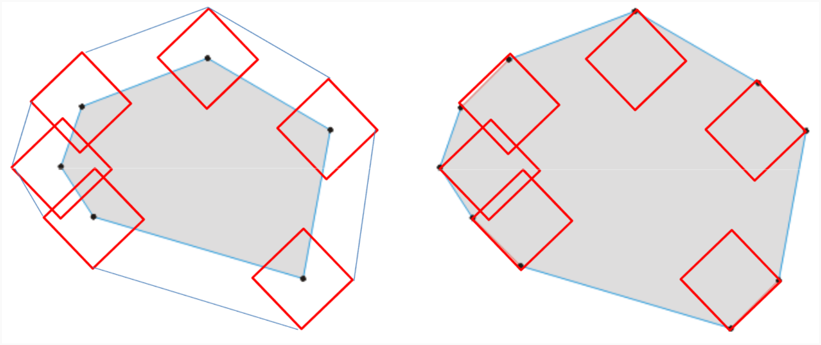


Figure 9: Computing the Minkowski Sum of two convex polygons

References

Minimum area k-gon (generalized version)

- Aggarwal, Alok, et al. ”Geometric applications of a matrix searching algorithm.” Proceedings of the second annual symposium on Computational geometry. ACM, 1986.

Next Class

Line Sweep