

Computational Geometry

Algorithms for manipulation of
geometric objects

We will concentrate on 2-D geometry

Numerically robust - try to avoid
division

- Round-off error
- Divide-by-0 checks

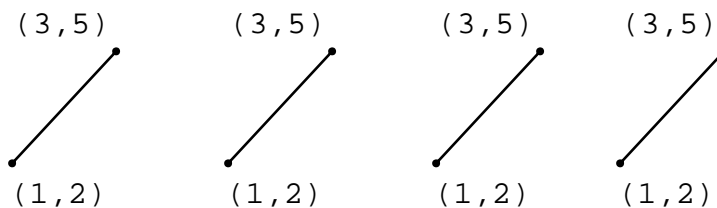
Techniques

Basics

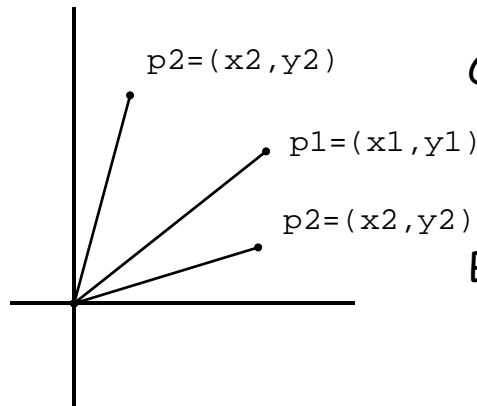
Point $p =$

Line segment p_1-p_2

Can represent different things



Relative Orientation of Segments



How do we know which case it is?

Could compare slopes

But that has division problems

Cross Product

Can compare slopes without division

$$y1/x1 < y2/x2$$

$$x2 \cdot y1 < x1 \cdot y2$$

Algorithm Design & Analysis

Examples

$(3, 2) \times (2, 4) =$

$(3, 2) \times (4, 1) =$

$(3, 2) \times (2, -2) =$

$(3, 2) \times (-3, 1) =$

$(3, 2) \times (-6, -4) =$

$(3, 2) \times (2, 4) =$

$(3, 2) \times (4, 1) =$

$(3, 2) \times (2, -2) =$

$(3, 2) \times (-3, 1) =$

$(3, 2) \times (-6, -4) =$

Lecture Notes 8
David Maier
5

Algorithm Design & Analysis

Left Turn, Right Turn

How do we know if a sequence of two line segments turns to left or right?

Lecture Notes 8
David Maier
6

Algorithm Design & Analysis

Use Cross Product

Translate p_2 to origin, along with other points. Look at relative position of p_1, p_3

$p_1 = (3, 6)$

$p_2 = (4, 3)$

$p_3 = (6, 1)$

$p_1' = (3-4, 6-3)$

$p_2' = (4-4, 3-3)$

$p_3' = (6-4, 1-3)$

Lecture Notes 8 David Maier **7**

Algorithm Design & Analysis

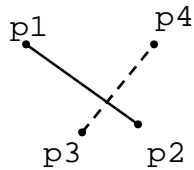
Intersection: Cases

Main conditions
 p_1, p_2 on different sides of p_3, p_4 on different sides of

Lecture Notes 8 David Maier **8**

How Do You Tell Different Sides?

One is a left turn and one is a right turn



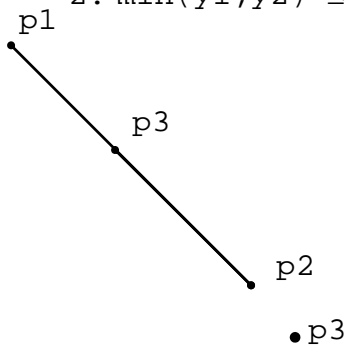
If 0 Change in Direction

Then the point is on the p1—p2 line somewhere

OnSegment(p1, p2, p3)

1. $\min(x_1, x_2) \leq x_3 \leq \max(x_1, x_2)$

2. $\min(y_1, y_2) \leq y_3 \leq \max(y_1, y_2)$



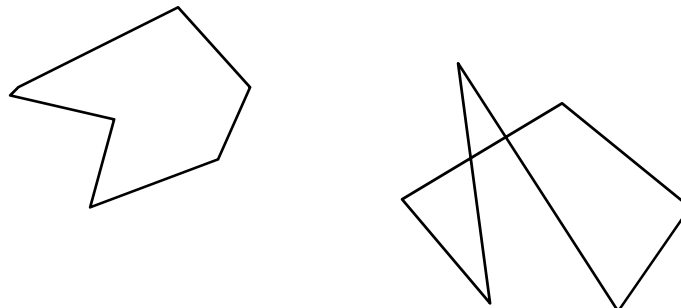
Beware of Boundary Conditions

- Co-linear points, points on line
- Multiple intersections

- Vertical, horizontal lines
- p_1-p_2 versus p_2-p_1
- Degenerate objects
What about intersection with the degenerate line segment p_3-p_3

Polygon

Polygon - closed sequence of sides
Simple polygon if edges don't cross



Exercise 33.1-7

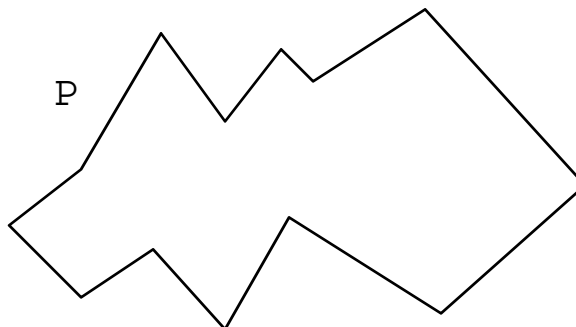
Find out if point p is in simple polygon P
 P has n points p_1, p_2, \dots, p_n , want $O(n)$

1. Assume no 3 consecutive points on the same line
2. Make sure p doesn't lie on any edge of P
3. Construct a segment $p-q$ where q is for sure
4. Count number of segments of P that intersect $p-q$

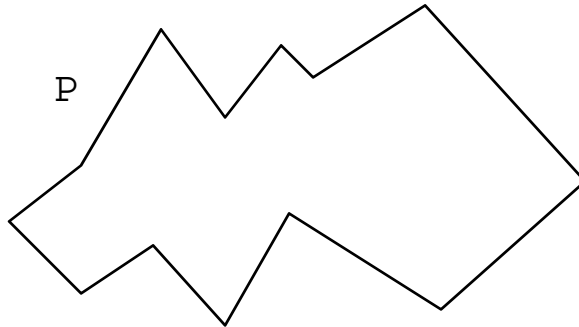
How to Find a Point Outside of P

Let $x' = \max x_i$

If $p = (x, y)$, let $q =$

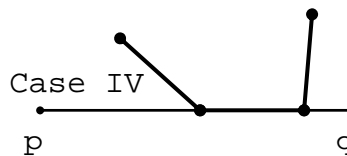
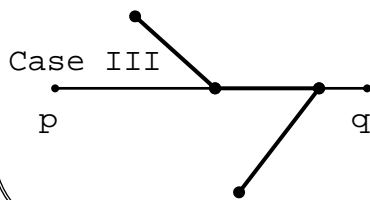
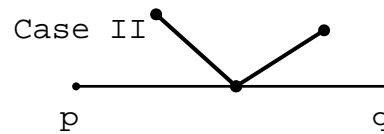
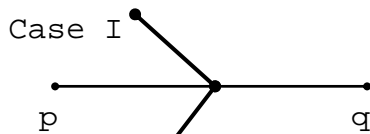


Cases to Watch out For
 $p-q$ passes through a vertex



What's the Difference in These Cases

Do previous and next points lie on the same side of $p-q$ or not?



In Cases III & IV, need to test if p_{i-1}
and p_{i+2} are same side of $p-q$ or not

In $O(n)$ time, can head around polygon,
counting number of intersections
with $p-q$, taking into account

Pairwise Intersection

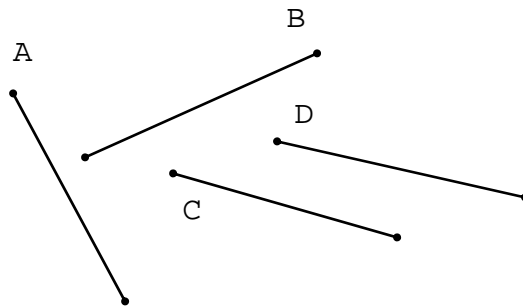
Have a collection of n line segments,
want to know if there is a pair that
intersect

Perhaps they represent traces on a circuit
layer

Sweep = A vertical line moving left to
right over the segments
(conceptually)

Interested in Intersections with Sweep Line

Want the intersection points in top to bottom order

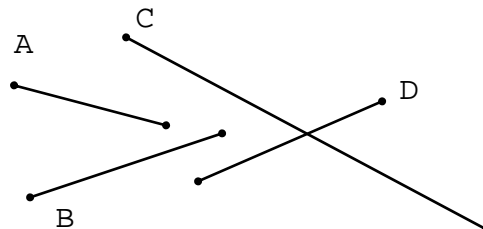


Intersecting Segments

Intersecting segments adjacent in some sweep list

Assumptions:

- No 3 segments intersect
- No vertical segments

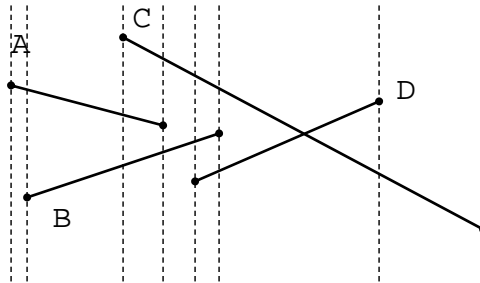


Where Does Sweep List Change?

Sweep list changes at segment endpoints.

Update sweep list "just after" encountering each endpoint

- after it enters for
- after it leaves for



Maintaining Sweep List

Keep sweep list as a structure that supports $O(\log n)$ operations

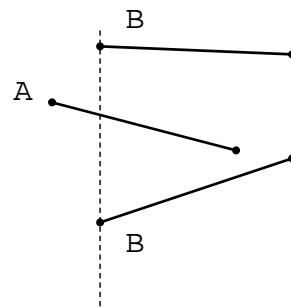
Operations:

- Insert segment
- Delete segment
- Above:
- Below:

Some kind of balanced tree, ordered on

How to Order New Segment on Sweep Line

Need to know where new segment B is relative to each segment A already in sweep line.



Order Endpoint List

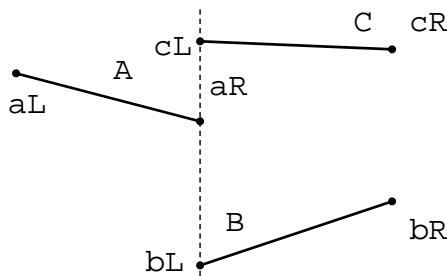
Need to have the endpoints in the order that the sweep line encounters them

Sort on x-coordinate

What about ties?

Left endpoints win over

Lower y-coordinate wins over



Algorithm Design & Analysis

Algorithm

EP (endpoints):
SL (sweep list)

for each p in EP in order

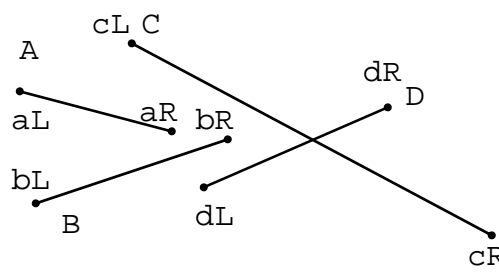
- insert s into SL
- check if s intersects lines above and below

- check if segments above and below s intersect each other
- delete s from SL

Lecture Notes 8David Maier

Algorithm Design & Analysis

Example

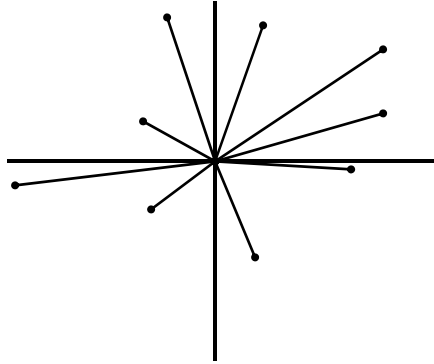


| EP | SL |
|----|----|
| aL | |
| bL | |
| cL | |
| aR | |
| dL | |
| bR | |
| dR | |
| aR | |

Lecture Notes 8David Maier

Polar Angle

Next couple algorithms, need to order points by polar angle.



Discussion Problems

Have $Q = \{p_1, p_2, \dots, p_n\}$

1. Given p , find point p_i in Q with smallest positive polar angle from p .
2. Sort points in Q by increasing polar angle from x axis.

Try to solve these problems using only cross product.

Convex Hull

Given $Q = \{p_1, p_2, \dots, p_n\}$, what is smallest convex polygon P such that

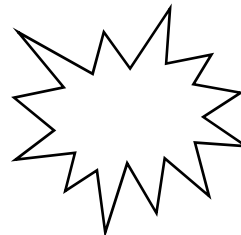
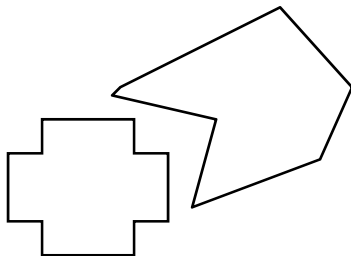
Vertices of P will be in Q

Consider Q with 100 points.

- What is the most that can be on convex hull?
- What is the fewest that can be on convex hull?

Uses of Convex Hull

Approximate more complicated shape by convex hull, for intersection "pre-check"

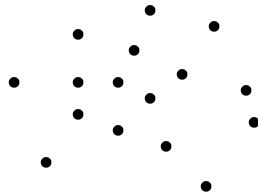


Multiple Convex Hull Algorithms

Some algorithms are good when the number of points on the hull is low. Others perform same on any number of points.

Can often use heuristics to speed up either.

Make any convex polygon R using points in Q .
Then points on interior of R

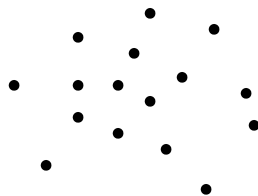


One Way to Choose R

Let p_a, p_b, p_c, p_d be the points with
 $\min x, \max x, \min y, \max y$.

Let R be polygon $p_a-p_d-p_b-p_c-p_a$

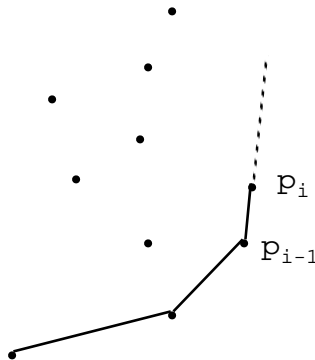
Will this always remove points?



Jarvis's March

Package wrapping:

Of points not on hull so far, which is at least angle with the last point on the hull



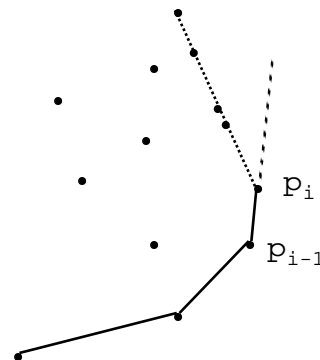
Issues

Co-linear points: What if several points on same edge of convex hull?

Want most points:

Want least points:

How to find distance?



Issues, Continued

Where to start:

What line to use through first point:

Complexity

At each point, do an $O(n)$ process to find point with minimum positive polar angle.

How many times do we need to do that search?

So, $O(n \cdot h)$ where $h =$

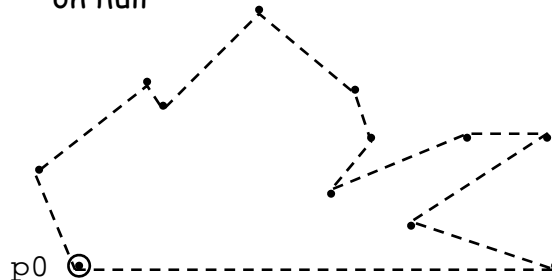
Graham Scan

Start with "angle sweep" and remove points

Find point p_0 with minimum y-coordinate

Sort points by polar angle from horizontal line through p

Always $O(n \log n)$, no matter how many points on hull



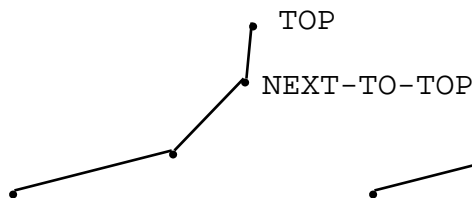
Candidate Hull on Stack

Push points on stack in order

Before you push, check angle with top two points on stack

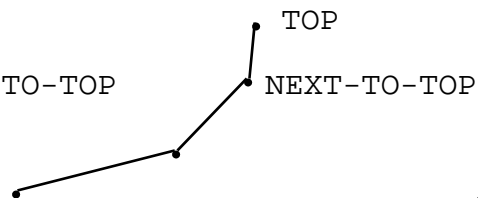
Left turn:

• P_i



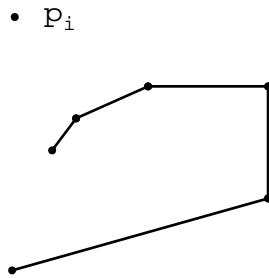
Right turn:

• P_i



May be Several Pops in a Row

Can have several points popped because of p_i



Graham Scan: Start-up

Input:

1. Let $p[0]$ have
2. sort $p[1] \dots p[n]$ by angle with
3. Have stack S
4. Push()
Push()
Push()

Graham Scan: Body

```
for i = 3 to N
  while NEXT-TO-TOP-TOP-p[i] is
    POP
  PUSH( )
S holds the convex hull when done
```

Complexity

$O(n \log n)$ to sort points

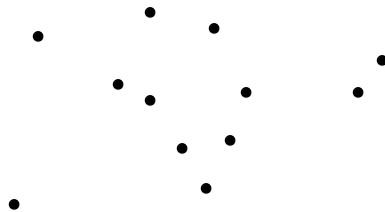
$O(n)$ for rest

Each point gets pushed once

Can do multiple pops in while-loop, but each is
of a previously pushed point

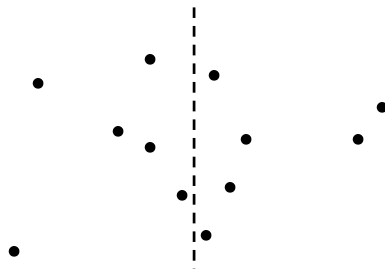
Planar Minimal Spanning Tree

Suppose I give you points in the plane
to represent nodes in a graph
All edges exist, weight is distance between
end points.
What is the first edge to add?



Closest Pair

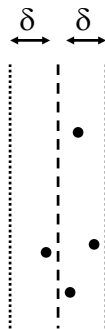
Use divide and conquer
Split points in half by x-coordinate
Find closest pair on each side
Let $\delta =$



Closer Pair

Can there be a pair of points closer than δ ?

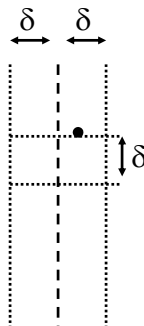
Yes! Near the boundary
So check



Avoid Too Many Comparisons

Don't want to compare all pairs of points near the boundary, otherwise get

Only compare each point near boundary to next 7 in order of y-coordinate



Complexity

For n points, have two sub-problems of size $n/2$

How much other work outside of the recursive calls?

Might seem like $O(n \log n)$ because of the sorting

That would give complexity

However, make two lists of the points at the very beginning, one sorted on x and the other on y

Can construct ordered lists for sub-problems in linear time.

$$T(n) = 2T(n/2) + O(n)$$