Lecture 2 | 6.046J

Divide & conquer

Convex hull

Median finding

Paradigm:

Given a n-sized problem,
we divide it into 'a' sub problems
of size $\frac{n}{b}$, $a \geq 1$, $b > 1$

Solve each subproblem recursively.

For most problems: Combine solutions of those sub-problems.

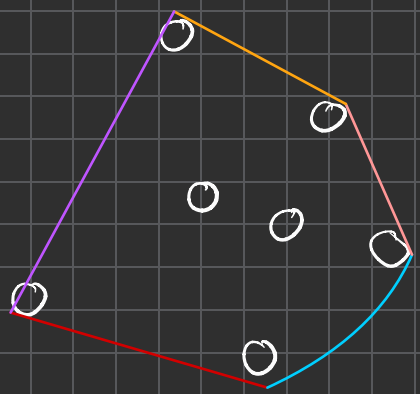
$$T(n) = a\,T\left(\frac{n}{b}\right) + [\text{Work for combine}]$$

Convex hull

Given n points on the plane

$S = \{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$.

Assume no two points in S have the same x coordinates or y coordinates (for convinience), and no three points are co-tansent (in a line).



Convex hull is the smallest convex polygon containing S denoted $CH(S)$.

We represent the hull as a sequence of points on the boundary of the hull in clockwise order, as a doubly linked list.

Disregarding efficiency

If we had points S

e.g.:

Bad

Good

A brute-force approach is to look at all pairs of points in S and determine whether all other points in S fall to one side of the line. If so x_0, x_1 start the sequence. Then look at all pairs (x_1, x_i) and do the same procedure.

This approach yields, for n points, $O(n^2)$ segments and $O(n)$ tests per segment hence $O(n^3)$ which is not optimal for large input sets.

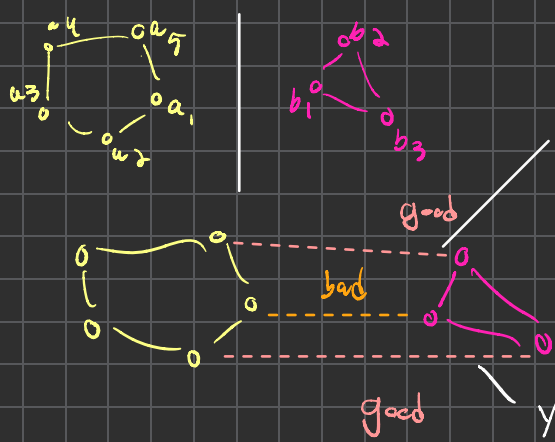
Sort points by x-coordinates once and for all.

For input set S divide into left half A and right half B by x-coordinates.

Then we compute $CH(A)$ & $CH(B)$ recursively and combine.

How, then, do we merge?

Suppose we have two hulls:



We look at tangents

The obvious merge is looking at all pairs of points $\Theta(n^2)$

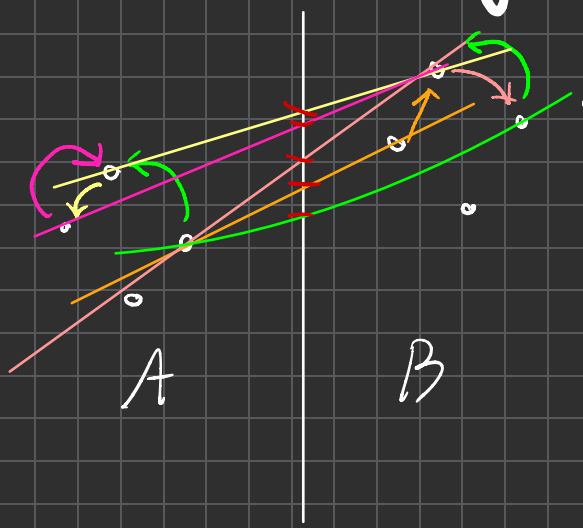
but we'd like taking max y-values and min y-values

However,

does NOT work!

Constant time is no good, but can we do better than $\Theta(n^2)$?

Two Finger Algorithm



How to find highest tangent

We start with the rightmost point in A and leftmost point in B s.t $|P_a - P_b|$ is minimized.

Then we note the height of the tangent between A & B

We move counter-clockwise in B and clockwise in A undoing moves which do not yield a new max Y-value

If we do both moves without improving, max has been reached.

Then we do the same for lower tangent swapping counter-clockwise and clockwise and minimizing Y

Complexity: $O(n)$ for moving $T(n) = T(\frac{n}{2}) + \Theta(n) \Rightarrow \Theta(n \log n)$

Very good!

When we find top and bottom tangents how do we represent the new hull. (T_a, T_b), (B_a, B_b)

First link: (T_a, T_b)

Go down b-list til B_b is found

Then link to B_a and go in A-list until T_a is found again.

Which clearly is $\boxed{O(n)}$

Median finding

We want to find median in unsorted list in less than $O(n \log n)$ time

Given a set of n numbers, define rank(x) as the numbers y in the set, $y \leq x$

i.e $\text{rank}(x) = |\{y \in S \mid y \leq x\}|$

Find element x s.t $\text{rank}(x) = \lfloor \frac{n+1}{2} \rfloor$

is the lower median, while the upper median is $\text{rank} = \lceil \frac{n+1}{2} \rceil$

Select routine Select(S, i) ← rank.

- Pick $x \in S$
- Compute $k = \text{rank}(x)$

$B = \{y \in S \mid y < x\}$

$C = \{y \in S \mid y > x\}$

if $k = i$: return x

if $k > i$: return select(B, i)

if $k < i$: return Select(C, $i-k$)

$\Theta(n^2)$ for a BAD selection

We need to deterministically and cleverly pick x.

Arrange S into columns of size 5. ($\lceil \frac{n}{5} \rceil$ columns)

Sort each column (big elements on top) linearly.

Find "median of medians"