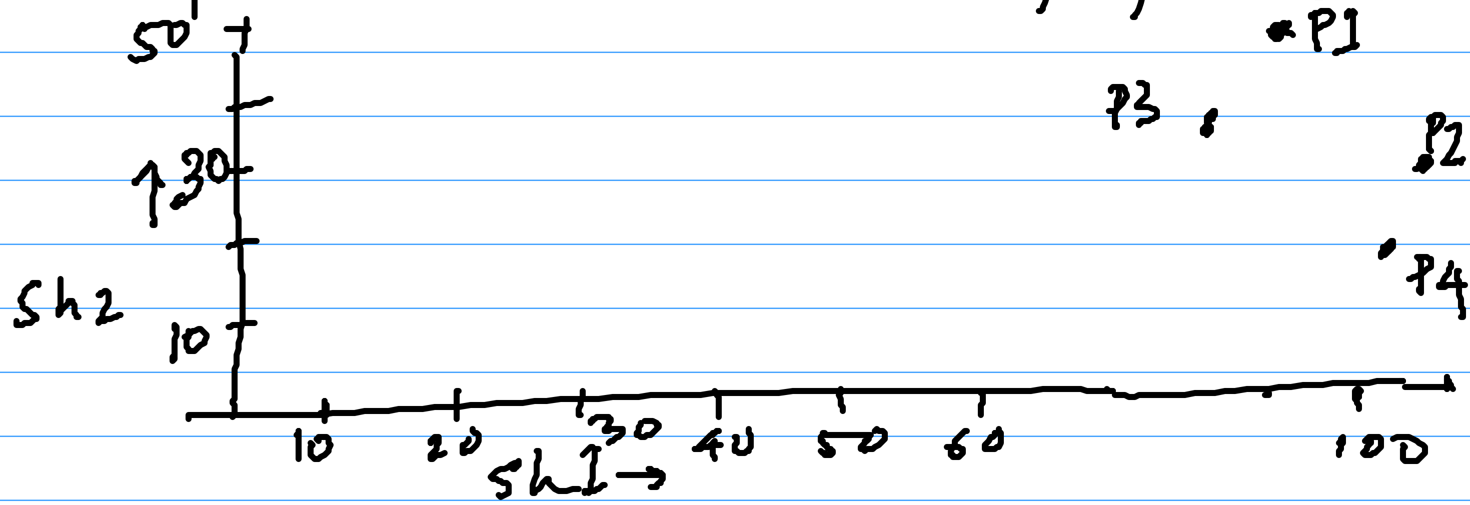
CSL 852 Computational Geometry

Lecture 3

	P1 ✓	P2 ✓	P3	P4
Sh1	100	120	90	115
Sh2	50	30	40	20

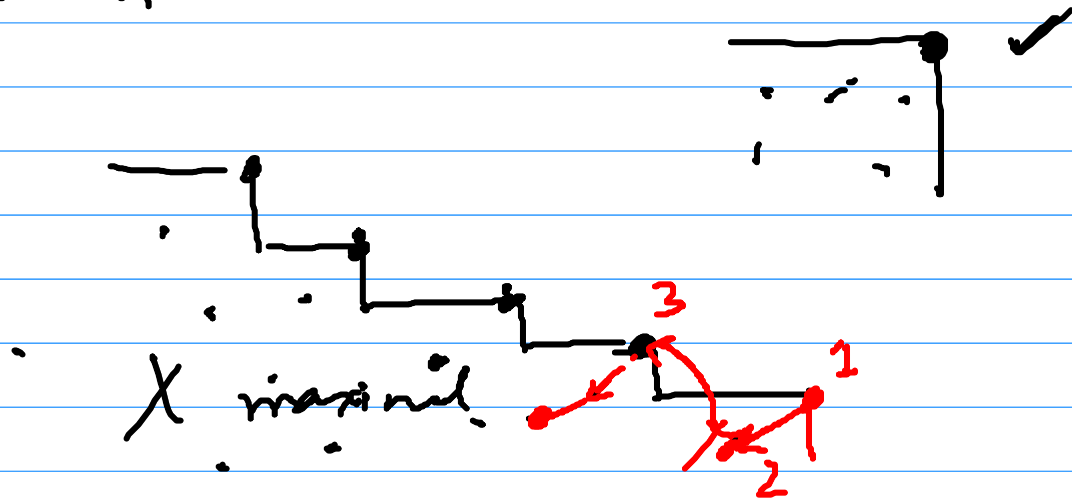
Value of shares : not known for future



P_i dominates P_j iff $x_i > x_j$ and $y_i > y_j$

Given a set S of n points in d dimensions we want to identify $S' \subset S$, such-that $\forall p \in S'$, there does not exist any point $q \in S$ such that q dominates p. S' is called the **maximal** points of S.

Problem Find the set of maximal points among a given set S of n points.



1. Sort the points along x axis
2. Starting from the point the largest x coordinate, we move left (in decreasing values of x) $O(n \log n)$

and maintain the value of the largest y coordinate till now, say Y_{max}

If for the current point p

$y_p < Y_{max}$ then p is <u>not</u> max

else p is maximal

$(Y_{max} \leftarrow y_p)$

For each point additional $O(1) \Rightarrow$ total $O(n)$

Can we do better?

Lower bound argument

Idea of $\Omega(n \log n)$ lower bound for maximal points

Construct an input $(x_i, y_i)_{1 \le i \le n}$

Such that $x_i + y_i = C$.

How many maximal points?

For any pair p_i, p_j

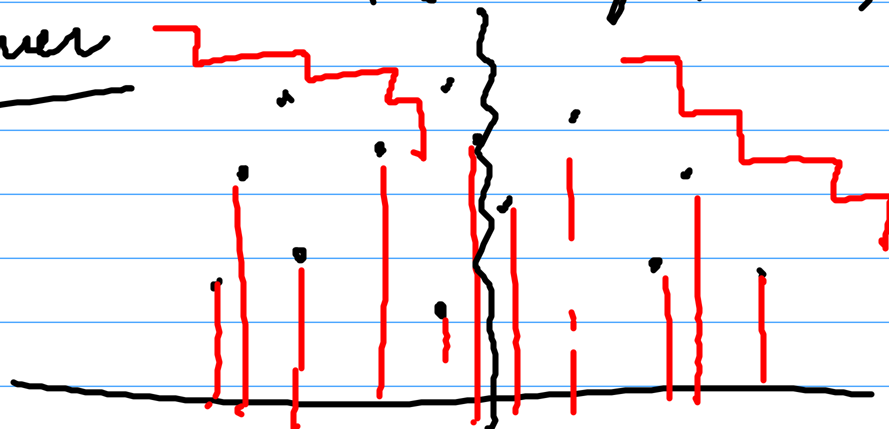
either (i) $x_i > x_j$ and $y_i < y_j$

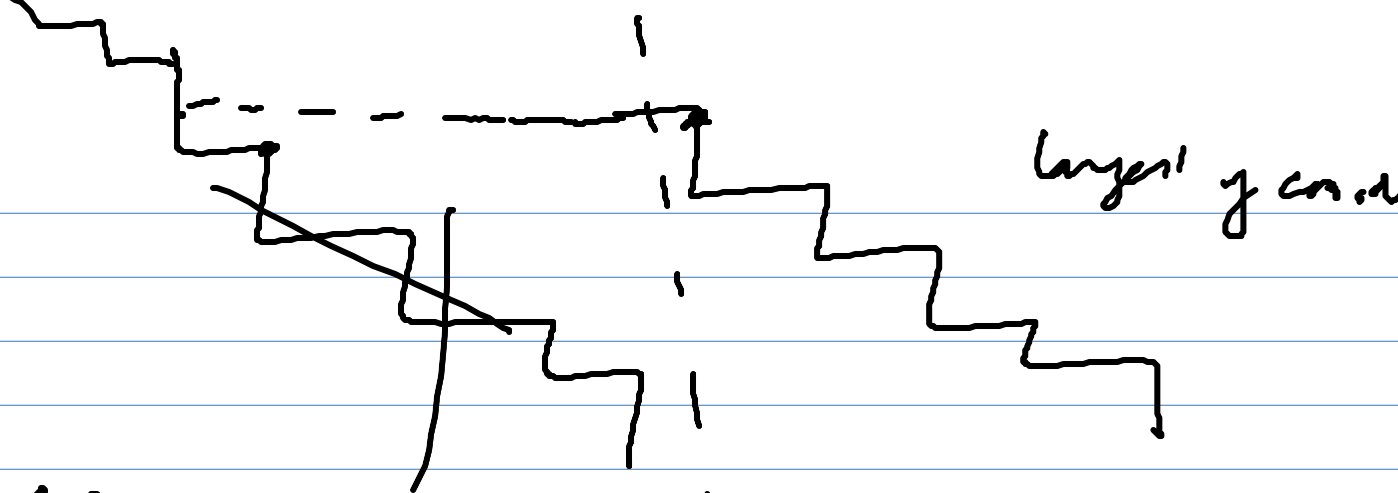
(ii) or $x_i < x_j$ and $y_i > y_j$

At the end of any algorithm for maximal, there is information about the comparison between every pair of points, i.e. we can sort (without any further comparisons)

So it cannot be faster than sorting, i.e. the lower bound of sorting applies

<u>Divide and conquer</u>

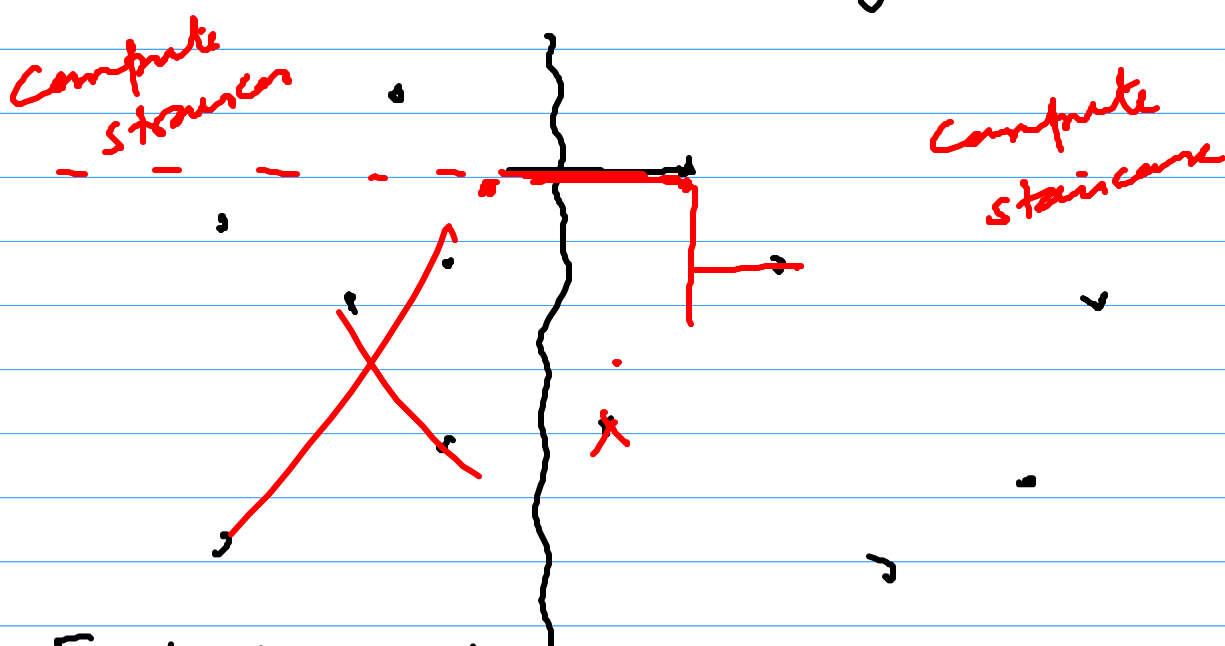




$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \qquad T(1) = O(1)$$

$$\Rightarrow T(n) = O(n \log n)$$ — median + merging

- Faster algorithm when there is only one maximal point
- output size may be crucial



1. Find the median
2. Find the intersection of the staircase with median and eliminate all points below it (and to its left)
3. Compute staircase of left, comp. right staircase

$$T(n) = O(n) + T(n_\ell) + T(n_r)$$

$$n_\ell, n_r \leq \frac{n}{2}$$

Suppose h is the size of final output (h is not known)

$$T(n, h) = O(n) + T(n_\ell, h_\ell) + T(n_r, h_r)$$

time for input size n and output size h

$$n_\ell, n_r \leq \frac{n}{2}$$

$$h_\ell + h_r = h - 1$$

$$T(n, h) = O(n \log h)$$

(Can be shown that this is optimal w.r.t input & output sizes