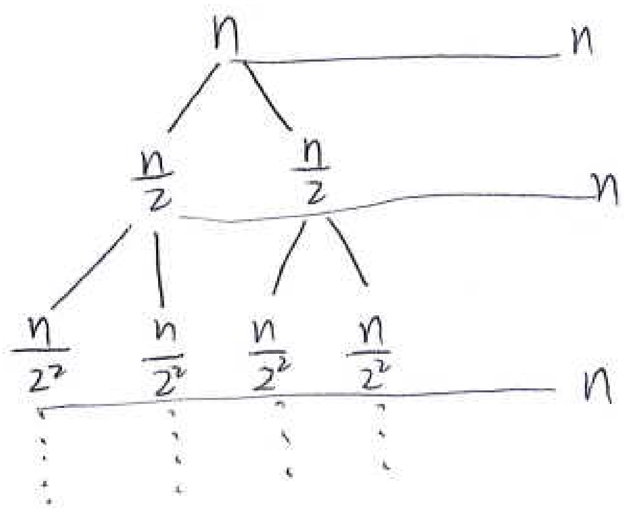
 Aug XX CS381

MergeSort

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + cn \qquad \boxed{n \log n}$$

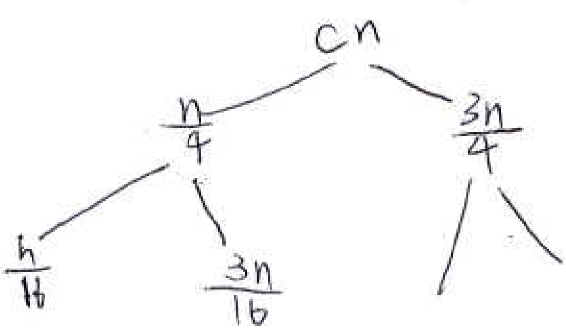
$$T(1) = c'$$

Recursion Tree Method



$$\frac{n}{2^i} = 1 \Rightarrow 2^i = n \Rightarrow i = \log_2 n \longrightarrow \text{depth}$$

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + cn^2 \qquad \boxed{cn} \qquad \boxed{cn^2}$$



cn^2

$\frac{10}{16}cn^2$

$\vdots$

$\left(\frac{10}{16}\right)^i cn^2$

$$n\left(\frac{3}{4}\right)^i \leq 20$$

$$i \sim \log_{\frac{4}{3}} n$$

Aug 28. CS381

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

$$T\left(\frac{n}{2}\right) = 2\ T\left(\frac{n}{2^2}\right) + \frac{cn}{2}$$

$$T(n) = 2^2\ T\left(\frac{n}{2^2}\right) + 2\left(\frac{cn}{2}\right) + cn$$

$$= 2^i\ T\left(\frac{n}{2^i}\right) + icn$$

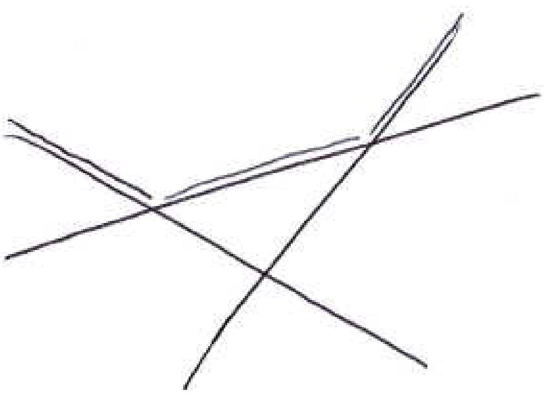
Stop when $2^i = n$

$$= n\ T(1) + c(\log_2 n) n$$

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + cn$$

$T(m_0)$

$m_0 \leq 20$



$f_i(x) = a_i x + e_i$

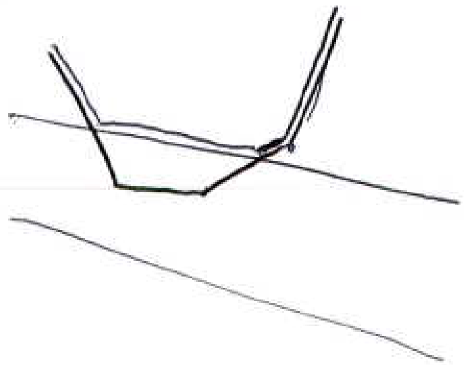
$i = 1, 2 \cdots n$

$h(x) = \max\{f_i(x)\}$

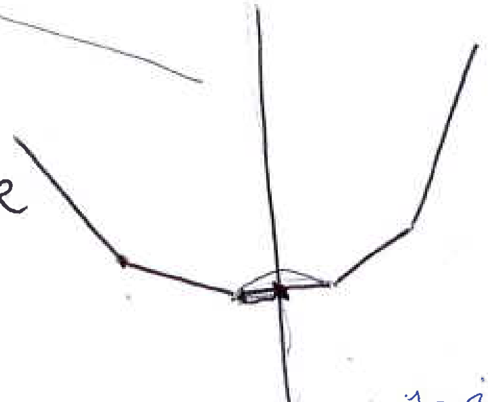
$i = 1, 2 \cdots n$

Compute Point

$h(x)$



presort by slope



$$\begin{cases} y_0 = a_1 x_0 + b_1 \\ y = a_2 x + b_2 \end{cases}$$

~~point (x, y_2)~~

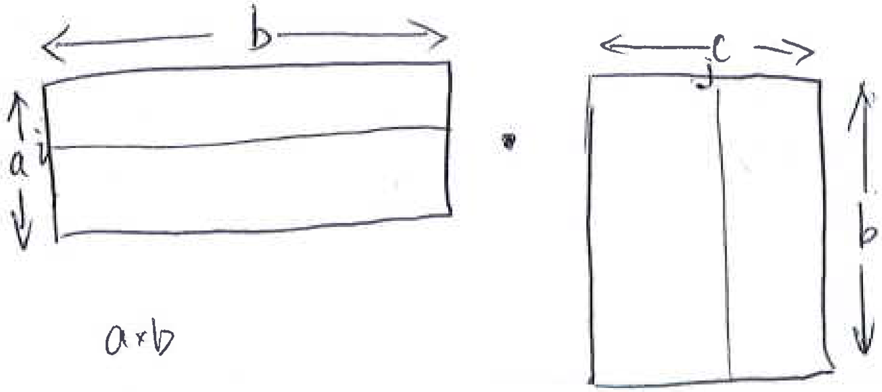
$$y - y_2 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_2)$$

$(x_1, y_1) - (x_2, y_2)$

$\cancel{y = a_1 x + b_1}$

$(x_3, y_3) - (x_4, y_4)$.

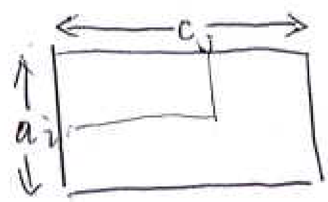
$$y - y_4 = \frac{y_4 - y_3}{x_4 - x_3}(x - y_4)$$



$a \times b$ $b \times c$.

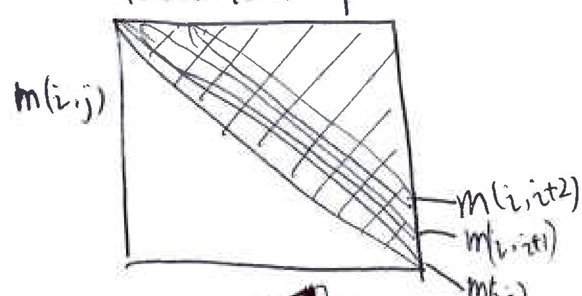
$a \times c$ matrix

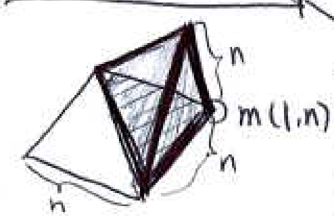
$=$



Total time : abc

table look up.





$\boxed{\text{Total work } \frac{n^3}{6}}$

$A_1 \cdots A_i A_{i+1} \cdots A_n$

$m(i,j)$ $A_i A_{i+1} \cdots A_j$. least cost of

$m(i,i+1)$ $A_i A_{i+1} = P_{i-1} P_i P_{i+1}$

$P_{i-1} \times P_i$ $P_i \times P_{i+1}$

$m(i,i+2)$ $A_i A_{i+1} A_{i+2}$

$$\begin{cases} (A_i A_{i+1}) A_{i+2} : P_{i-1} P_{i+1} P_{i+2} + m(i,i+1) \\ A_i (A_{i+1} A_{i+2}) = P_{i-1} P_i P_{i+2} + m(i+1,i+2) \end{cases}$$

$m(i,i+l)$.

$m(i,i+l+1)$

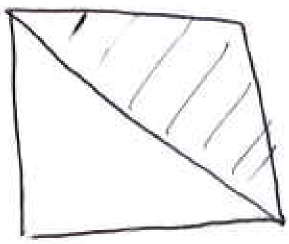
We want $m(1,n)$

$$l+1 \text{ choices} \begin{cases} m(i,i) + m(i+1,i+l+1) + P_{i-1} P_i P_{i+l+1} & A_i (A_{i+1} \cdots A_{i+l+1}) \\ m(i,i+1) + m(i+2,i+l+1) + P_i P_{i+1} P_{i+l+1} & (A_i A_{i+1})(A_{i+2} \cdots A_{i+l+1}) \\ \vdots & \vdots \\ m(i,i+l) + m(i+l+1,i+l+1) + P_{i-1} P_{i+l} P_{i+l+1} & (A_i \cdots A_{i+l}) A_{i+l+1} \end{cases}$$

$m(i,j)$ $\qquad$ $g(i,j)$ $\quad$ $(A_i A_{i+1} \cdots A_{g(i,j)}) \cdot (A_{g(i,j)+1} \cdots A_j)$

$g(i,j)$: the best choice when computing $m(i,j)$, the partition position

$g(i,j)$



compute $g(i,j)$ while computing $m(i,j)$.

then use ~~g(1,n)~~ **g** to recursive

$g(1,n)$

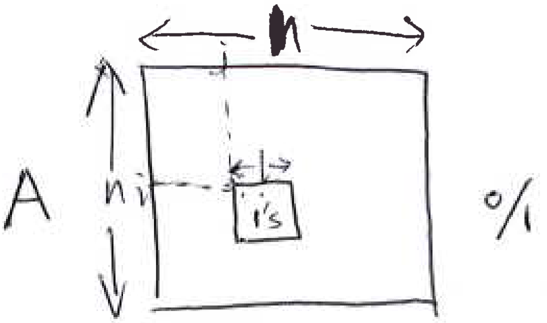
$(A_1 \cdots A_{g(1,n)})(A_{g(1,n)+1} \cdots A_n)$

$g(1, g(1,n))$ $\qquad$ $g(g(1,n)+1, n)$

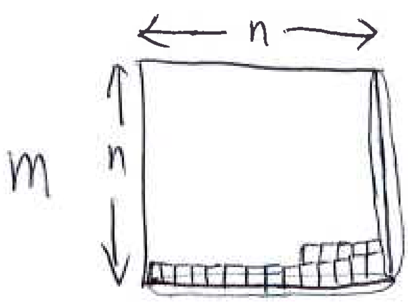
$(\quad)(\quad)$ $\qquad$ $(\quad)(\quad)$

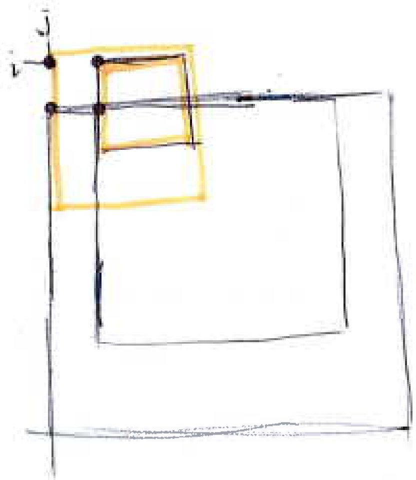
Sep 6 CS381

Find the largest square that contains all 1's



m(i,j): largest l the square could have starting from (i,j). [top left corner of square]





To determine value of m(i,j).

if A(i,j) = 0.

then m(i,j) = 0.

else

the largest l for (i,j) is

1 + the minimum of $\underbrace{m(i, j+1), m(i+1, j), m(i+1, j+1)}_{\text{three neighbors}}$

time: $\boxed{O(n^2)}$

space $\boxed{O(n^2)}$

space $\boxed{O(n)}$ → store only one row at a time

For Array A. $\overline{\quad i \quad j \quad}$

Find i, j where S_{ij} is Max when $S_{ij} = A_i + \cdots + A_j$.

① $\boxed{O(n^3)}$. Brute force.

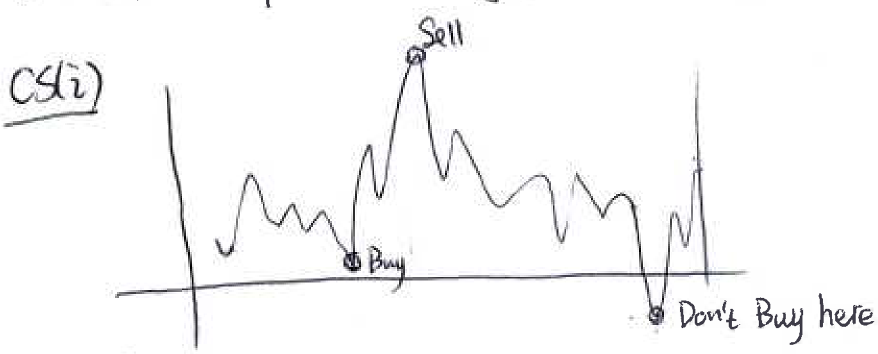
② $\boxed{O(n^2)}$.

$CS(i) = A(0) + A(1) + \cdots + A(i)$. $CS(i+1) = CS(i) + A(i+1)$ ———— $O(n)$.

$S(i,j) = CS(j) - CS(i-1)$

Find the Max of $S(i,j)$ ———— $O(n^2)$

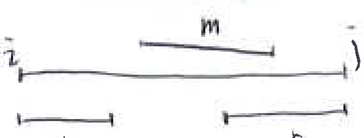
③ $\boxed{O(n)}$ Treat $A(i)$ as price change on the i'th day, Then $CS(i)$ is the final price on the i's day.



$\underline{low(i)} = Min\{CS(j)\}$ $j < i$ the lowest Buy point on the left of i.

$\underline{BestUpswing} = Max\left(CS(i) - low(i)\ \&\ \text{Old BestUpSwing}\right)$

④ $\boxed{O(n)}$ All done in ONE sweep recursion



每段keep三个值:
两段小的算好后，算大的

$MaxSum(0, n-1)$ $T(1) = C_1$
$T(n) = 2T(\frac{n}{2}) + C_2$

$m = max\{m', m'', r' + l''\}$.
$l = max\{l', w' + l''\}$
$w = w' + w''$

Two files: <DP>

f_1: $l_1', \cdots l_m'$ $\qquad X = x_1 \cdots x_m$

f_2: $l_1'', \cdots l_n''$ $\qquad y = y_1 \cdots y_n$

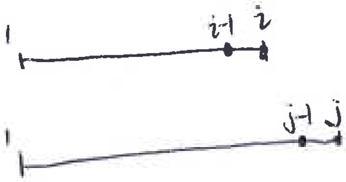
diff f_1 f_2

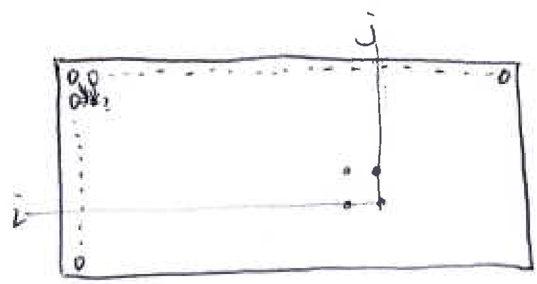
longest common subsequence problem
(不连续的子片段)

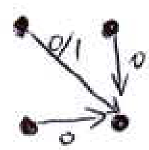
Q: Find the size of longest common subsequence.
LCS

C_{ij} = length of LCS $\qquad$ index 0 means empty string.
of $x_1 \cdots x_i$
of $y_1 \cdots y_j$

we want C_{mn}



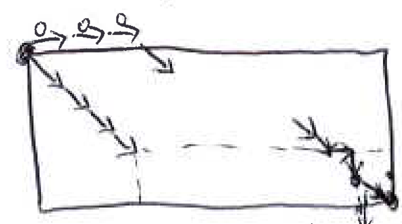




$$C_{ij} \longleftarrow \begin{cases} C_{i-1,j-1} \\ C_{i,j-1} \\ C_{i-1,j} \end{cases}$$

$$C_{ij} = \max \begin{cases} C_{i-1,j-1} + 1 & \text{if } x_i = y_j \;;\; C_{i-1,j-1} + 0 \text{ if } x_i \neq y_j \\ C_{i,j-1} + 0 \\ C_{i-1,j} \end{cases}$$

max of three candidates.

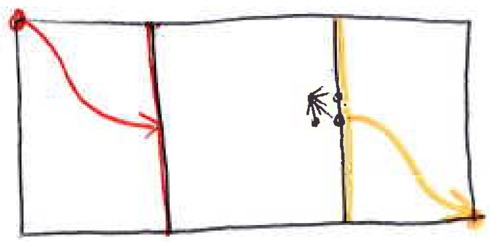


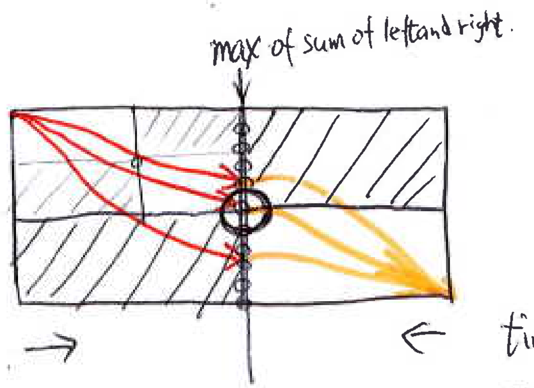
backtrack

To find the path.
back track in matrix.
Use $O(mn)$ space.

~~To find the path in O(n) space.~~

two vectors sweeping.





recursive.

time: $mn + \frac{mn}{2} + \frac{mn}{4} \cdots \cdots = 2mn$

~~space: 2n~~

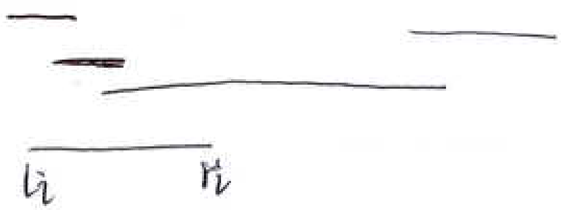
Inorder.

left ......

pop out point.

right ....

[Greedy]

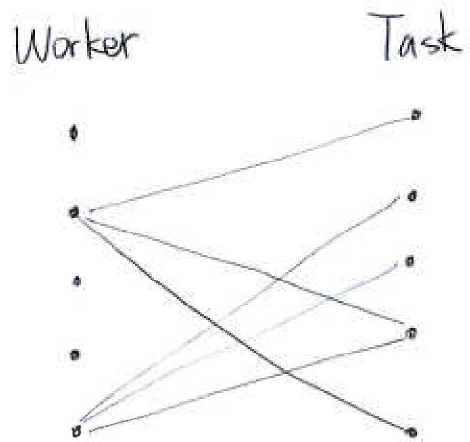


① only 1 resource. maximum number of events
sort events by finish time.
sweep from left to right

② multipul resources fulfill all events.
△ determine number of resources:
maintain a counter for number of overlaping events. ⇒ max count is the number of resources needed.
while sweeping from left to right.
if ~~encounter~~ bump into a l_i, counter++
else if bump into a r_i, counter--.

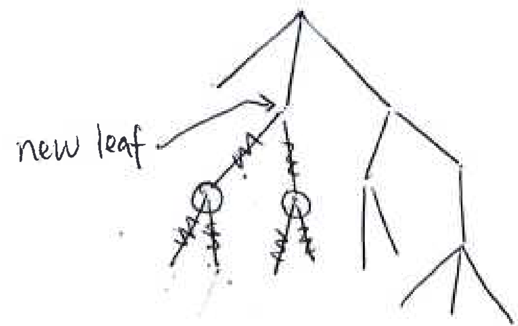
△ assigning resources:
event list : sorted by start time
available resource list :
boolean isAvailable[maxcount]
while sweeping from left to right.
if bump into a l_i, assign i to some available resource and mark it false
if bump into a r_i, mark the corresponding resource true.

[Matching]



① // initialize
isAvailableWorker[i] = true.
isAvailableTask[i] = true.
// start assigning
mark corresponding booleans false.
- - - -

② $|E|\sqrt{|V|}$



Q: Minimum the number of guards.

- maintain a list
  CurrentLeaves:

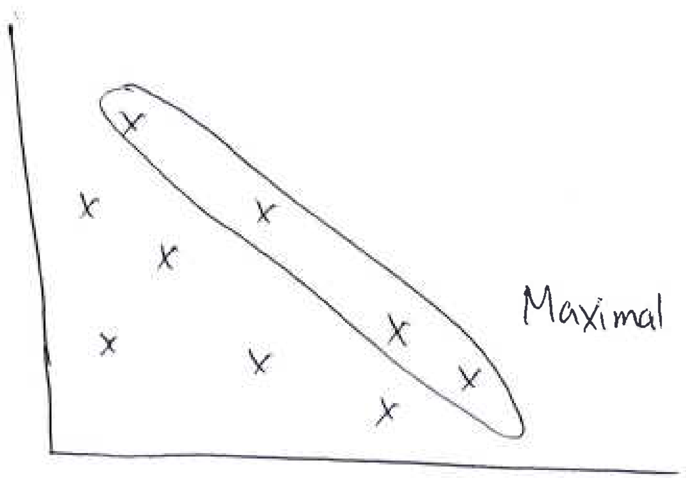
- For each node, maintain a reference counter
  if ref counter is 1, add it to list
- Idea: Grep a leaf from list, put guard at it's parent, "delete" corresponding edges, update ref counters.
  Grep another leaf from list....
  until list is empty.

- Linear.

Input: n points (x_i, y_i)

Output: points that are not dominated by others.

if $x_i > x_j$ and $y_i > y_j$,
then (x_i, y_i) dominate (x_j, y_j)



Maximal element.

① $O(n^2)$.

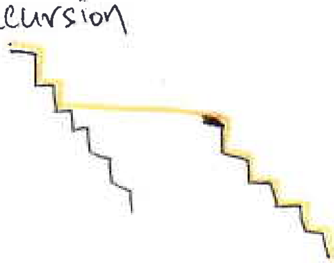
② $O(n \log n)$

sort points by x coordinate.

sweep from right to left.

remember the highest Y encountered so far

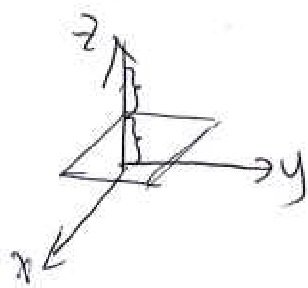
③ Recursion



recursive on left and right. then merge.....

④ Binary search

3D version



x above
x below

recursion ......

⑤ $n \log h$ h is output size

Maximum Increasing Subsequence.

Input: $a_1, \cdots a_n$

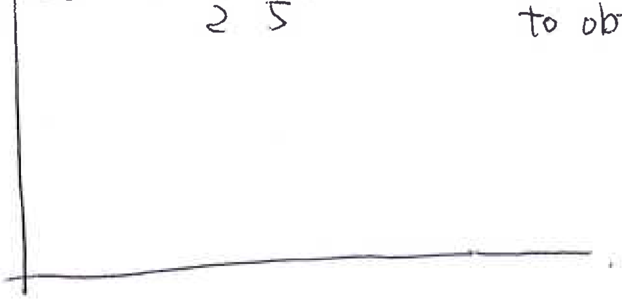
Output. $i_1, i_2, \cdots i_k$ — k

where $i_1 < i_2 < \cdots < i_k$

	1.	3.	9.	5.	8.	2.
label	1	2	3	3	4	2

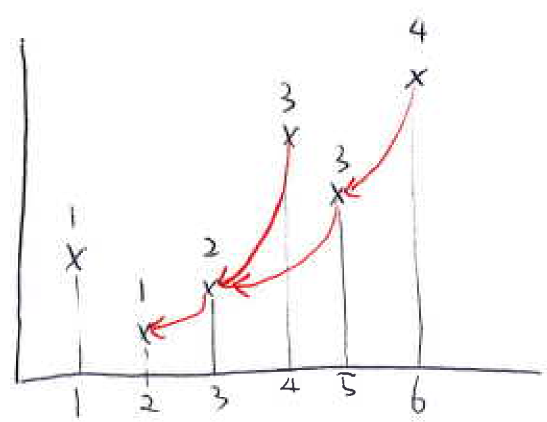
k. label array: 1, ~~3~~, ~~9~~, 8
2 5

log k
binary search label array, ~~[illegible]~~ ask left neighbor to obtain its label and +1, then replace the right neigh[illegible]



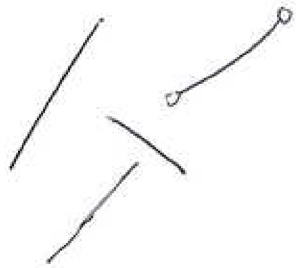
→ sweep.

$\boxed{n \log k}$



aux array ~~a_1~~ a_2, a_3, ~~a_4~~ a_5, a_6

Input



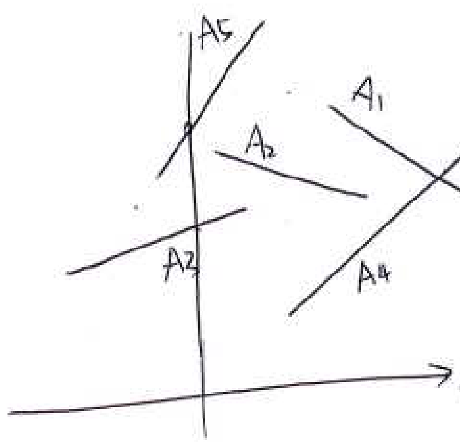
n segments. do any of them intersect?

intersect detection.

Output: Yes/No.

① $O(n^2)$. for every pair of segments...

② $O(n)$



current tree: A_3. A_5.

event list: sorted endpoints by x coordinate.

use binary search tree to store current segments..

compare. 上面的比下面的大.

insertion. deletion from tree.

check neighbor in tree

△ The intersection detected is not the left most intersection

Find all intersections

△ keep on going. modification needed. event list: endpoints and intersection already detected.

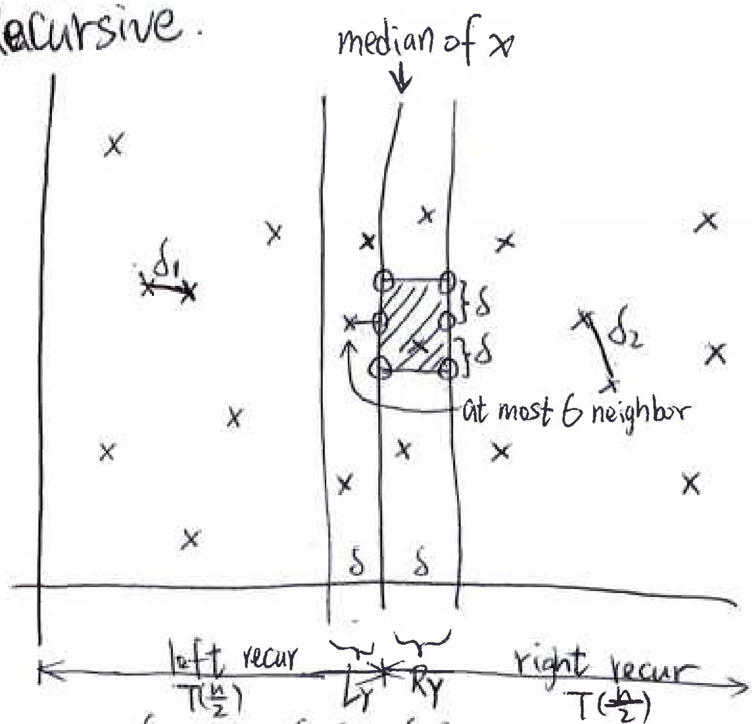
$n\log n + T\log n$

Input: points (x_i, y_i).

Output: the closest two points that have the smallest distance between them

① Recursive.

Exam Question



$\delta = \min\{\delta_1, \delta_2\}$.

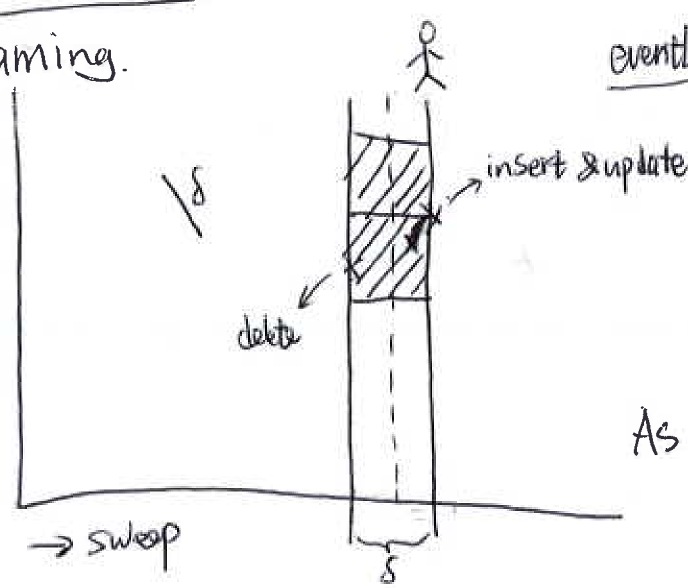
L_Y: in left river of width δ. sort points according to Y coordinate and store in list.

R_Y: in right.

walk along in one river, buttom to up. check with corresponding shaded area in the other river. —— Linear. $(< 6n)$

$\boxed{O(n\log n)}$ $T(n) = 2T(\frac{n}{2}) + cn$

② Streaming.



eventlist: points in the interval (maintain)

Sweep from left to right,

when encounter a new point, insert and update δ (narrow interval, delete...)

when a point is out of bound of interval delete it in eventlist

As we sweep, the interval shrinks.

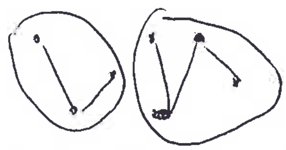
$O(n\log n)$

Lower bound

Finding the min

lower bound $n-1$

prove:



$n-1$ edges

Finding the min and max at the same time.

lower bound $\frac{3}{2}n-2$

algorithm pair up, $\frac{n}{2}$ pairs

	Paid	Gain
① Compare each pair and throw into 2 bags	$\frac{n}{2}$	n
② find min in smaller bag	$\frac{n}{2}-1$	$\frac{n}{2}-1$
③ find max in bigger bag	$\frac{n}{2}-1$	$\frac{n}{2}-1$
	$\frac{3n}{2}-2$	$2n-2$

prove: min max

(1, 1) ··· n

from $2n$ 1's down to 2

$(1,1) \geq (1,1)$ → 0, 0 -2

$(1,0) \geq (1,0)$ → 0 -1

$(0,1) \geq (0,1)$ → 0 -1

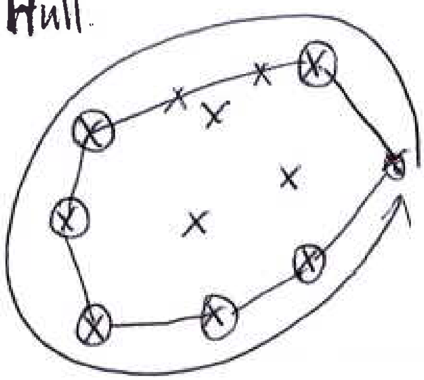
$(1,0) < (0,1)$

$(0,1) > (1,0)$

$\frac{3n}{2}-2 = \frac{n}{2}$ n

$n-2$ $n-2$

Convex Hull



lower bound : $n \log n$

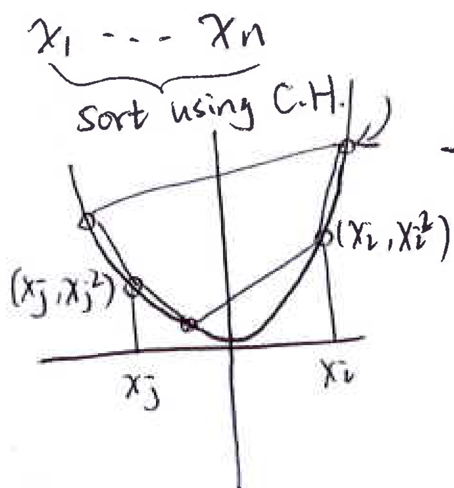
prove:

~~Convex hull with~~

C.H.

Convex hull with less than nlogn algorithm $\Rightarrow$ Sorting with less than nlogn algorithm

假设 C·H 时间复杂度 < nlogn.
那么 sorting 可以 < nlogn.
矛盾. 所以 C·H ≥ nlogn

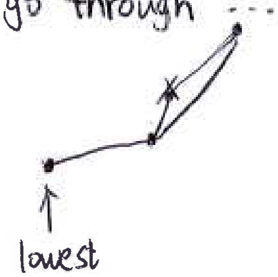


$f(x) = x^2$

Actual Algorithm

① Sort by polar angle. — $n \log n$

② go through ... — n



if left turn : push.

if right turn : pop until no right turn

if C.H complexity < $n \log n$
then sorting can be done
by using C.H algorithm on (x_j, x_j^2) ...

complexity of sorting : $\underset{\text{C.H}}{n \log n} + \underset{\text{preprocess each point to } (x_j, x_j^2)}{cn}$

Contradict.

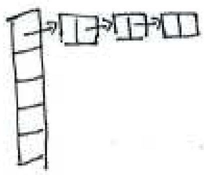
$\Omega(n \log n)$ lower bound.

1. Sorting
2. Element Uniqueness
3. Set Equality

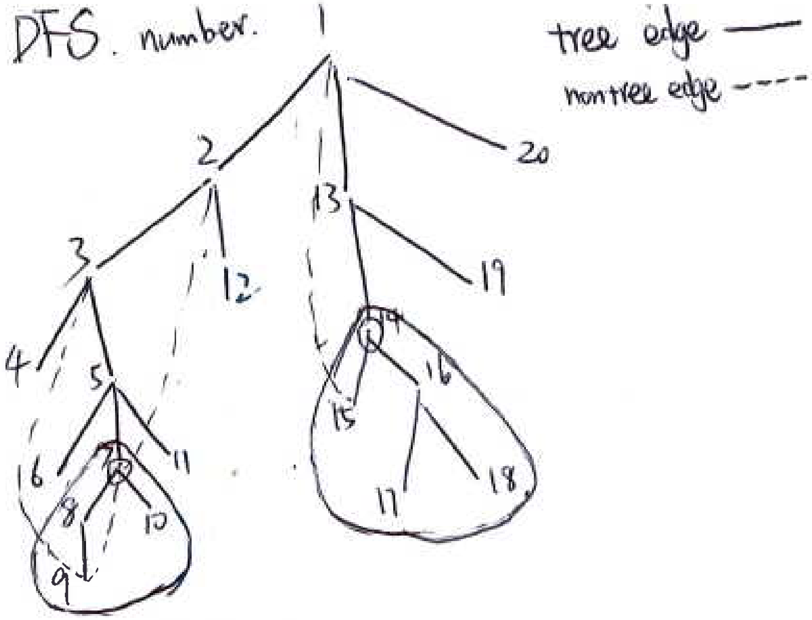
Graph Algoriths.

DFS

BFS.

Adjacency lists 

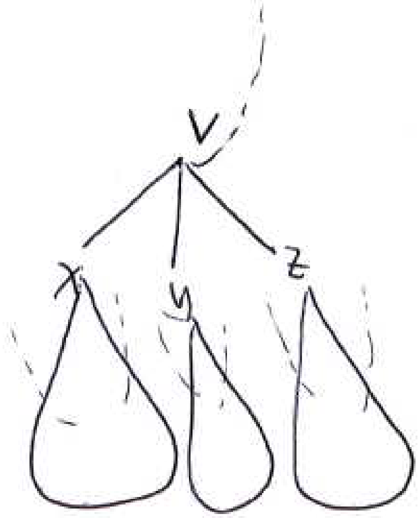
DFS. number.



$low(14) = 1$

$low(7) = 2$

low(v): in subtree of v.
the smallest DFS number point it could jump to



when done with low(x),
update parent of x: v
$low(v) = \min(low(v), low(x))$

$$\sum_{v \in V} |L(v)| = 2|E| \quad \text{Linear}$$

check. even. odd.

△ Find the bridge

if $low(x) \leq \cancel{low(v)}\ v$

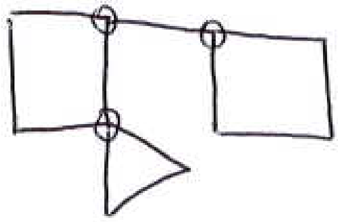
$\Updownarrow$

(v, x) is not a bridge.

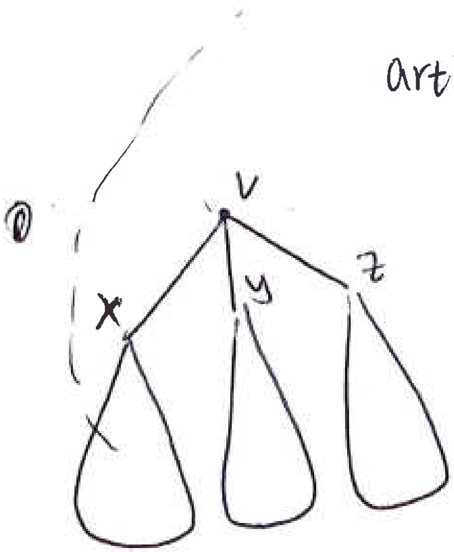
if $low(x) > \cancel{low(v)}\ v$

$\Updownarrow$

(v, x) is a bridge.



articulation vertices



if low(x) < v && low(y) < v && low(z) < v

$\Updownarrow$

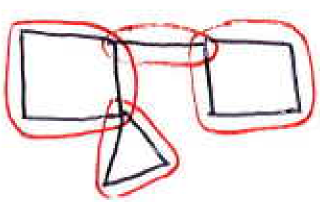
v is not a articulation point

Biconnected component

push edges to stack

if low(x) $\geq$ v.

pop edges (subgraph of x) → one biconnected component.



Question. (Exam):

△ if (a,b) is a bridge.

a, b are articulation points

$\boxed{\text{False}}$

△ if (a,b) is a bridge.

and graph has more than 2 points

at least one of a and b

is articulation point.

$\boxed{\text{True}}$

△ if graph has more than 2 points

there are at least <u>two</u>

articulation points.

$\boxed{\text{True}}$

Leaves of DFS tree are

articulation points.

If a tree has only one leaf,

then the node is an articulation point.