Orion
PAGE:
DATE: 17/07/17

ALGORITHMS - II

* Sum of 2 n-bit binary numbers – $O(n)$, $\Omega(n)$, $\theta(n)$.
(Asymptotically tight upper & lower bound)

* Multiplication of 2 n-bit binary numbers – $O(n^2)$

Integer Multiplication

$(a+ib)(c+id) = (ac-bd)+i(ad+bc)$, $i = \sqrt{-1}$

4 multiplications → ac, bd, ad, bc.

Can we reduce multiplications? Yes.

$ad+bc = (a+b)(c+d) - (ac+bd)$

3 multiplications → $(a+b)(c+d)$, ac, bd.

Let a and b be two n-bit numbers
Let n be a power of 2, for simplicity.

Let $a =$ [a_L | a_R] and $b =$ [b_L | b_R]
(a_L, a_R, b_L, b_R : $n/2$ bits each)

Mathematically, $a = 2^{n/2} \cdot a_L + a_R$
$b = 2^{n/2} \cdot b_L + b_R$

$$ab = (2^{n/2}a_L + a_R)(2^{n/2}b_L + b_R)$$
$$= 2^n a_L b_L + 2^{n/2}(a_L b_R + a_R b_L) + a_R b_R$$

$2^n a_L b_L$: left shift, $O(n)$; $2^{n/2}(a_L b_R + a_R b_L)$: $O(n)$; $a_R b_R$: $O(n)$

Combining → $O(n)$ time.

Let $T(n)$ denote the time complexity of this divide and conquer algo

$$\Rightarrow T(n) = 4T\left(\frac{n}{2}\right) + O(n) \approx 4^{\log_2 n} + (\log_2 n)\,O(n) \times n$$
$$= O(n^2)$$

($4^{\log_2 n} = 2^{\log_2 n^2} = n^2$; $(\log_2 n)\,O(n) \times n = O(n^2)$)

ANUBHAV JAIN NOTES

Orion
PAGE:
DATE: 18/07/17

Karatsuba's Algorithm for Integer Multiplication

$ab = (2^{n/2}a_L + a_R)(2^{n/2}b_L + b_R) = 2^n a_L b_L + 2^{n/2}(a_L b_R + a_R b_L) + a_R b_R$

Obs:- $(a_L b_R + a_R b_L) = (a_L + a_R)(b_L + b_R) - a_L b_L - a_R b_R$

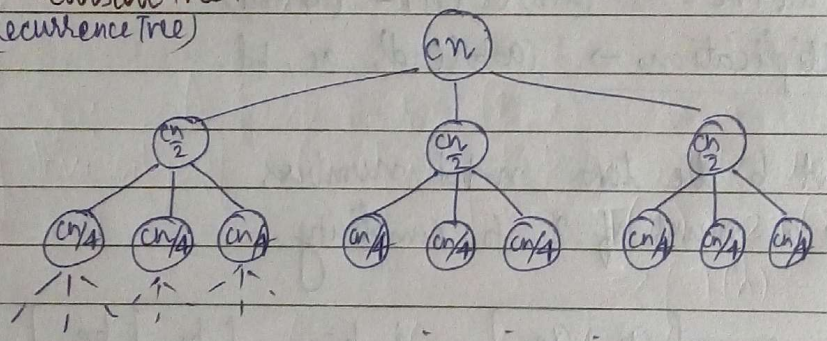
1, 2, 3 — Multiplications

$T(n)$ = time complexity of multiplication of two n-bit numbers.

$$T(n) = 3T(n/2) + O(n)$$

* $f(n) = O(n) \Rightarrow f(n) \le cn \quad \forall n \ge n_0$ for some +ve const.c

Recursion Tree:-
(Recurrence Tree)



$cn = \left(\frac{3}{2}\right)^0 cn$

$\frac{3cn}{2} = \left(\frac{3}{2}\right)^1 cn$

$\frac{3^2 cn}{2^2} = \left(\frac{3}{2}\right)^2 cn$

$\frac{3^3 cn}{2^3} = \left(\frac{3}{2}\right)^3 cn$

$\left(\frac{3}{2}\right)^{\log_2 n} cn$

$$\Rightarrow T(n) = cn\left[\left(\frac{3}{2}\right)^0 + \left(\frac{3}{2}\right)^1 + \dots + \left(\frac{3}{2}\right)^{\log_2 n}\right]$$

$$= cn \frac{\left(\left(\frac{3}{2}\right)^{\log_2 n + 1} - 1\right)}{(1/2)} = 2cn\left[\frac{3}{2}\left(\frac{3^{\log_2 n}}{n}\right) - 1\right]$$

$\{x^{\log y} = y^{\log x}\}$

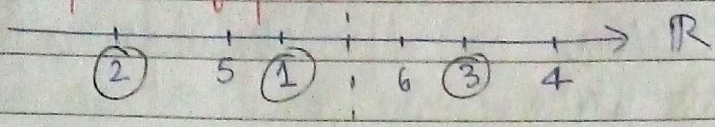
$$= 3c\, 3^{\log_2 n} - 2cn$$

$$= 3c\, n^{\log_2 3} - 2cn$$

$$T(n) = O(n^{\log_2 3}) = O(n^{1.585})$$

Orion
PAGE:
DATE: / /

Closest pair of points in 1D



$L = \{P_1, P_2, P_3\}$ → Recurse

$R = \{P_4, P_5, P_6\}$ → Recurse

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

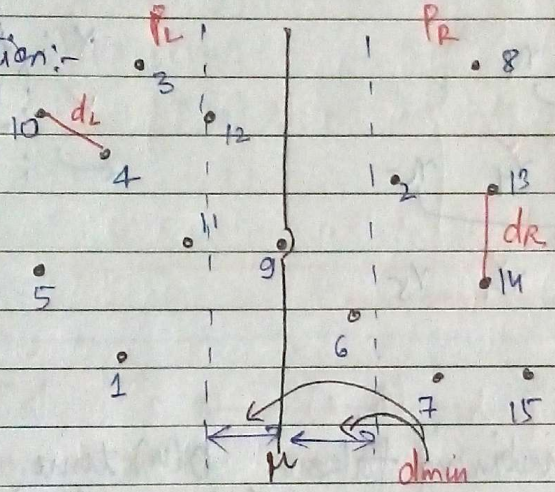
Combining Part.

Closest pair of points in 2D

Input → list of tuples → $P = (P_L \cup P_R)$

Divide / Partition in 2 parts based on x cordinates. by sorting list wrt x ($O(n \log n)$) and sort wrt y also.

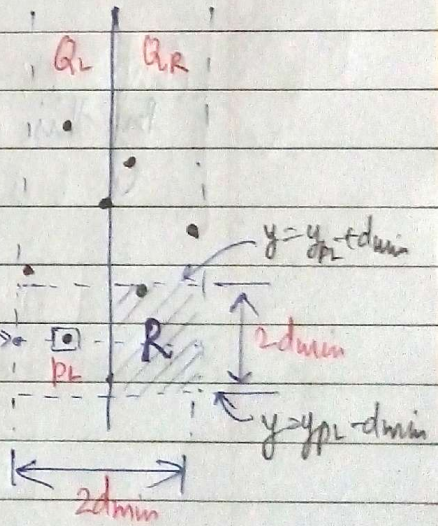
Visualization:-



$d_{min} = \min(d_L, d_R)$

$Q_L = \{p_L : p_L \in P_L \text{ and } x(p_L) \geq x_\mu - d_{min}\}$

$Q_R = \{p_R : p_R \in P_R \text{ and } x(p_R) \leq x_\mu + d_{min}\}$



In worst case, Q_L can have $O(n)$ points

Q_R ——— .

$\Rightarrow p_L \rightarrow O(n)$ points in worst case.

→ for each p_L, we need to find R. We will do binary search on Q_R. (find index for which $y \leq y_{p_L} + d_{min}$ & then for which $y \geq y_{p_L} - d_{min}$. All points between 2 indices lie in R).

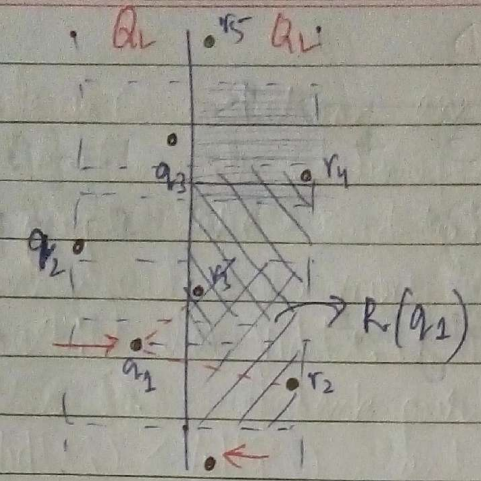
$\downarrow$

$O(\lg n)$.

$\Rightarrow$ Combining taken $O(n \lg n)$ time. $T(n) = 2T(n/2) + O(n \lg n)$

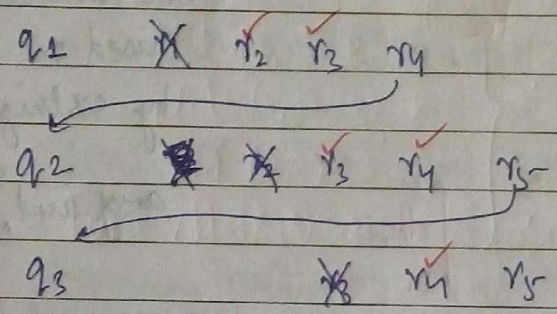
Orion

PAGE:
DATE: / /



r_1 → Out of R ⇒ abandon forever.

Q_L and Q_R → sorted wrt to y coordinates.



Discarded/
✗ : Abandoned forever
✓ : Comparison made (diff. found)
↩ : Out of R ⇒ Move to next point.

By this algo, Combining takes $O(n)$ time

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

$\Rightarrow \boxed{T(n) = O(n \lg n)}$

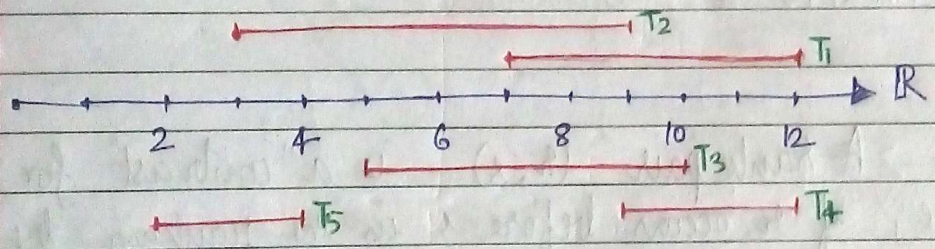
20/7/17

Orion
PAGE:
DATE: 20/7/19.

TUTORIAL-1

T1. (Task scheduler)
There are n tasks, each with a definite start time and a definite finish time. • Design an efficient algorithm for scheduling them to a minimum no. of machines. Each machine can process one task at a time. • Prove the correctness and • explain the time complexity of your algorithm. [3+4+3]

Ex. $T_1 = [7, 12]$
$T_2 = [3, 9]$ (We have sufficient no. of machines)
$T_3 = [5, 10]$
$T_4 = [9, 12]$
$T_5 = [2, 4]$.



T2. (Best Videoranker)
Let C be a community of n people who have membership of an online video parlor.

	V_1	V_2	V_3	V_4	V_5	
P_1 =	3	1	5	2	4	diff = ? = 3 (2 swaps)
P_2 =	4	2	5	1	3	diff = 1.
P_3 =	4	3	5	1	2	(1 swap)

The parlor has k videos, namely $V_1, V_2 \ldots V_k$. Each person sees and ranks the videos from 1 to k based on his/her liking. All these data are available as input. The problem is to find a the person in the community who holds "similarity" with the maximum no. of

people in C. • Design an efficient algorithm for this and explain its time complexity. [6+4]

→ Two people are said to have "Similarity" if their video rankings have no more than $\lfloor k/4 \rfloor$ contrasts.

Contrast: Let Vid(x, i) denote the rank of Video x by Person i. A rank pair (x, y) is a "contrast" if Vid(x, i) < Vid(y, i) (or >) and Vid(x, j) ≥ (or ≤) Vid(y, j) for two persons i and j.

Vid (2,2) = 2
Vid (5,2) = 3
Vid (2,3) = 3
Vid (5,3) = 2

} 1 Contrast.

Contrast: A rank pair (r, s) is a "contrast" for two persons i & j if r occurs before s in the ranking by Person i but s occurs before r in that by the person j.

	V_1	V_2	V_3	V_4	V_5
P_1 =	3	1	5	2	4
P_2 =	1	5	3	2	4

contrast = 2.

Orion
PAGE:
DATE: 31/7/19

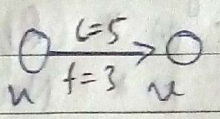
Network flow

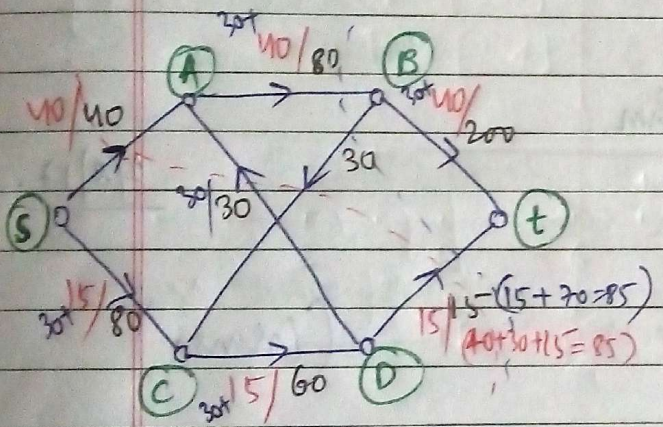
$G(V, E)$ directed the weighted capacities $c(u,v)$ for $(u,v) \in E$
flow function $f(u,v)$ for $(u,v) \in E$ $\quad u \longrightarrow v$

1. $f(u,v) = -f(v,u) \quad \forall (u,v) \in E$
2. $\Sigma f(u,v) = 0 \quad \forall u \in V - \{s,t\} \quad v \in V$
3. $f(u,v) \leq c(u,v) \quad \forall (u,v) \in E$

flow value $|f|$ for f is $\Sigma f(s,v)$

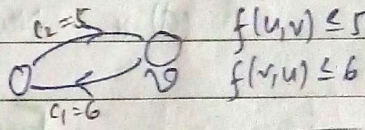
Defn:

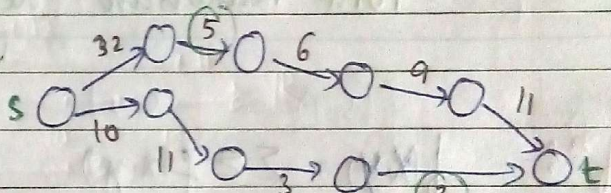




S-A-B-t : 40
S-C-D-t : 15
S-C-D-A-B-t : 30
85

Max flow $(u,v) = 5$
$\quad\quad (v,u) = 6$





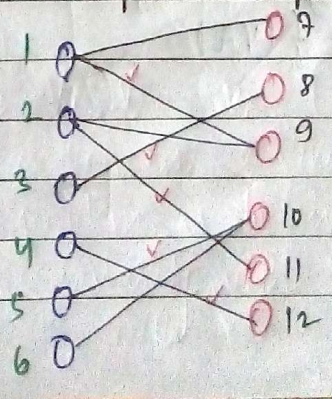
Max flow $(s,t) = 5+2 = 7$.
$\quad\quad (t,s) = 0$

Take any s-t cut, net flow will be same 85.

* Max. matching problem

NP Hard Problem

Vertex Cover :-
Set of points s.t every edge has it vertex in it



VC = {9, 3, 11, 4, 5}
{1, 7} {6, 10}.

VC = {1, 3, 2, 4, 10}

$n = 11 + 1 = 12 \quad m = 8$

$d = n - |VC| = 12 - 5 - 7$

ANUBHAV JAIN NOTES

Orion
PAGE:
DATE: / /

Red residual network.

Residual capacities → $c_f(u,v) = c(u,v) - f(u,v)$ (c(u,v) could be zero!)

Residual edges → $E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$

$E_f \not\subseteq E$ (in general)

$|E_f| \leq 2|E|$

Pg-652 (Ch-26 Max flow - CLRS 2nd Ed.)

Fig (a) * Make flow network G & flow f.
(b) * Residual Network G_f. → Get residual capacity from augmented path
(c) * New flow network
(d) * New Residual network.

• Ford-Fulkerson algorithm

1/8/17.

Def^n:- $f(X,Y) = \sum_{x \in X} \sum_{y \in Y} f(x,y)$ (Cormen)

Results:- $f(x,x) = 0 \qquad x, y \subseteq V$

$f(x,y) = -f(y,x)$

$x, y, z \subseteq V$ with $X \cap Y = \phi$

$f(x \cup y, z) = f(x,z) + f(y,z)$

$|f| = f(s,V)$ (definition)

$= f(V,V) - f(V-s, V)$

$= f(V, V-s)$

$= f(V,t) + f(V, V-s-t)$

$= f(V,t)$

} Results above.

(flow conservation!)

Orion

PAGE:
DATE: / /

Lemma: Graphs G, G_f — flow functions f, f' ⇒ G, $f+f'$

(f on G, f' on G_f)

Proof *

$$(f+f')(u,v) = f(u,v) + f'(u,v)$$
$$= -f(v,u) - f'(v,u)$$
$$= -(f(v,u)+f'(v,u))$$
$$= -(f+f')(v,u)$$

(Skew Symmetry)

*

$$(f+f')(u,v) = f(u,v) + f'(u,v)$$
$$\leq f(u,v) + [c(u,v) - f(u,v)]$$
$$= c_f(u,v)$$
$$\leq c(u,v)$$

(Capacity constraint)

*

$$\sum_{v \in V} (f+f')(u,v) \longrightarrow = \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) \quad u \in V - \{s,t\}$$
$$= 0 + 0 = 0.$$

(Flow conservation)

*

$$|f+f'| = \sum_{v \in V} (f+f')(s,v) = \sum_{v \in V} (f(s,v) + f'(s,v))$$
$$= \sum_{v \in V} (f(s,v)) + \sum_{v \in V} f'(s,v)$$
$$= |f| + |f'|$$

Augmented flow.

$$c_f(p) = \min \{ c_f(u,v) \mid (u,v) \text{ is on } p \}$$ (p → PATH.)

Lemma: G, f — p a path in G_f (augmenting)

Define $f_p : V \times V \to R$

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p \\ -c_f(p) & \text{if } (v,u) \text{ is on } p \\ \phi & \text{otherwise} \end{cases}$$

$|f_p| = c_f(p) > 0$

Orion
PAGE:
DATE: / /

Lemma. G, f, G_f, f_p augmenting flow on path p $\Rightarrow$ $f' = f + f_p$, $|f'| = |f| + |f_p|$

$G \rightarrow f'$

CUT (S, T)
$(S, V-S) = T$ $\quad s \in S, t \in V \setminus S$

Capacity of cut $c(u,v)$
$c(S,T) = \sum\sum_{u \in S, v \in T} c(u,v)$

[If $(u,v) \notin E$, $u \in S$, $v \in T$ then $c(u,v) = \phi$].

Lemma. $f(S,T) = |f|$

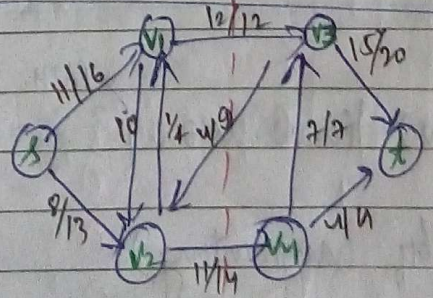
Proof:

$f(S,T) = f(S,V) - f(S,S)$
$= f(S,V)$ — source
$= f(s,V) + f(S-s,V)$
$= f(s,V)$
$= |f|$.

[$f(S,S) = 0$ { flow conversation, skew sym.

Corollary: $|f| \leq c(S,T)$ for any cut (S,T)

Proof: $|f| = f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) \leq \sum\sum_{u \in S, v \in T} c(u,v) = c(S,T)$



$S = \{s, v_1, v_2\}$
$T = \{t, v_3, v_4\}$.

$f(S,T) = 19$ $\quad c(S,T) = 26$.

$f(S,T) \triangleq \sum\sum f(u,v)$ → can be +ve
$c(S,T) \triangleq \sum\sum c(u,v)$ → always +ve.

Orion
PAGE:
DATE: / /

Main Theorem

(G, f, s, t) 1. f is maximum
2. G_f has no augmenting path
3. $|f| = c(S,T)$ for some (not all) cut (S,T) of G.

Proof:-

- (1) ⇒ (2) ¬(2) ⇒ $|f_p| > 0$ in G_f resulting in $G \xrightarrow{f'}$ s.t. $|f'| = |f| + |f_p| > |f|$ □.

- (2) ⇒ (3) Defn: $S \xrightarrow{\text{Reachability set from source}} \{ v \in V \mid \exists$ a path from s to v in $G_f \}$
  & $T = V \setminus S$
  Clearly, (S,T) is a cut $s \in S$ (Defn) $t \in T$
  Now for each $(u,v) \in E$, $u \in S$ $v \in T$
  $f(u,v) = c(u,v)$ because, else, $(u,v) \in E_f$ & $v \in S$ □.
  So, $|f| = f(S,T) = c(S,T)$.

- (3) ⇒ (1) $|f| \le c(S,T)$ as we know $\forall (S,T)$ cuts.
  $\therefore |f| = c(S,T) \Rightarrow |f|$ is maximum
  END OF PROOF.

Generic flow algorithm

1. $f \leftarrow \phi$ $[f(u,v) = f(v,u) = 0]$
2. while ∃ an augmenting path in G_f { (Any path ⇐ Ford Fulkerson)
3. do { $c_f(p) \leftarrow \min \{ c_f(u,v) \mid (u,v)$ is in $p \}$
4. for each (u,v) in p
5. do { $f(u,v) \leftarrow f(u,v) + c_f(p)$
   $f(v,u) \leftarrow -f(u,v)$ }
}

ANUBHAV JAIN NOTES

Complexity $O(E|f^*|)$, f^* is the max flow.
* EDMONDS-KARP = Augmentation along a shortest path from s to t.

Orion
PAGE:
DATE: / /

Longest path problem- NP Hard.

CPM- critical path method
operations research.

Lemma 1. ED-KARP, $G=(V,E)$, s, t,
then $\forall v \in V-\{s,t\}$, $\delta_f(s,v)$ in G_f never decreases with each flow augmentation.
($\delta_f(s,v)$ = shortest distance of v from s in augmented graph)

Thm 1- ED-KARP does $O(EV)$ flow augmentation

Cor 1- ED-KARP runs in $O(E^2V)$ time

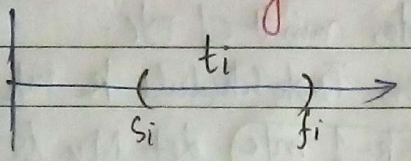
$\Rightarrow$

Orion

PAGE:
DATE: 03/08/17

Tutorial-1 Problems

T1. Task scheduling



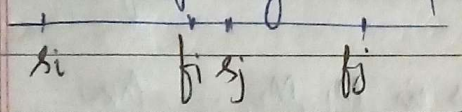
Tasks $T = \{t_1, t_2, \ldots, t_n\}$

t_i : s_i → start time

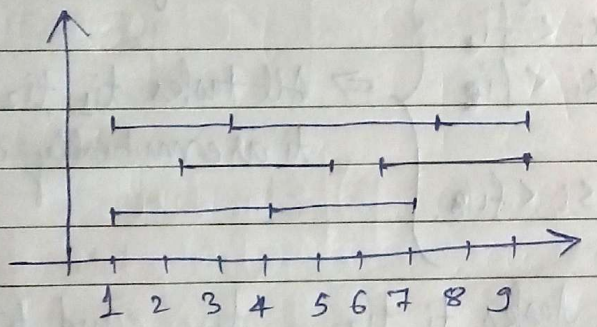
f_i → finish time

Objective: schedule the tasks to min # of m/c so that no m/c gets conflicting task.

Non-conflicting or compatible



$(s_i, f_i) \cap (s_j, f_j) \neq \phi$ ← Optimization problem.



Greedy Selection:
Select task i having earliest start time and schedule to a m/c that can handle it & there is no conflict with the tasks scheduled to the same m/c.

$M_1, M_2, \ldots$

Algorithm: TaskSchedule(T)

Input: $T = \{t_i : (s_i, f_i) \; ; i = 1..n\}$

Output: A non-conflicting schedule on min # of m/c.

```
1.  m ← 0                                   // # of m/c          O(1)
2.  while T ≠ φ                                                 O(n)
3.      remove a task ti from T with min si                     * O(1)
4.      if ∃ a m/c Mj with no conflicting task with ti,
5.          schedule ti to Mj                                   O(log n)
                                                                (using priority queue)
6.      else                                                    O(1)
7.          Allocate another m/c Mm+1
8.          Schedule ti to Mm+1
9.          m ← m+1.
10. return.
```

* : Preprocessing has to be done.

Orion
PAGE:
DATE: / /

Proof of correctness:

Line #4 ensures that algorithm TaskSchedule allocate non-conflicting tasks to the m/c.

Claim: Algorithm TaskSchedule provides min # of m/c.

Proof by contradiction: Let Algorithm TaskSchedule k m/c (If possible, there is a schedule with k-1 m/c) Let M_k be the last m/c allocated by Algorithm TaskSchedule & t_i be the first task allocated to M_k.

(line #4) $\Rightarrow$ t_i conflicts with tasks in $M_1, M_2, \ldots M_{k-1}$.

$\Rightarrow \exists\ t_{i_1} \in M_1, t_{i_2} \in M_2, \ldots, t_{i_{k-1}} \in M_{k-1}$

such that t_i conflicts with each of the tasks $t_{i_1} \ldots t_{i_{k-1}}$

$s_{i_1} < s_i < f_{i_1}$

$s_{i_2} < s_i < f_{i_2}$

$\vdots$

$s_{i_{k-1}} < s_i < f_{i_{k-1}}$

$\Rightarrow$ All tasks $t_{i_1}, t_{i_2}, \ldots t_{i_{k-1}}$ and t_i are mutually conflicting.

So, the k conflicting tasks $t_{i_1}, t_{i_2}, \ldots t_{i_{k-1}}$ and t_i cannot be allocated to k-1 m/c.

$\Rightarrow$ Contradiction

Hence, the claim is true.

Time Complexity:

Line #3: if the tasks are sorted in increasing order of start time, then it takes O(1).

$\Rightarrow$ Preprocessing (sorting) takes O(nlgn) time

Line #4: Keep allocated m/c $M_1 \ldots M_m$ in a min-priority queue

M_1 M_2 $\ldots$ M_m

$\{s_{i_1}\}$ $\{s_{i_2}\}$ $\{s_{i_p}\}$

with priority k_j = Start time of the last job scheduled to M_j.

$\Rightarrow$ O(log n)

Find M_j with last added task t_j having min start time

$M_j \rightarrow \{t_{j_1}, t_{j_2}, \ldots, t_{j_p}\}$

(k_j = earliest finish time)

Rest of the line takes O(1)

Total = O(nlogn)

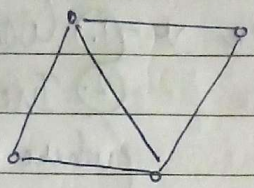
Orion

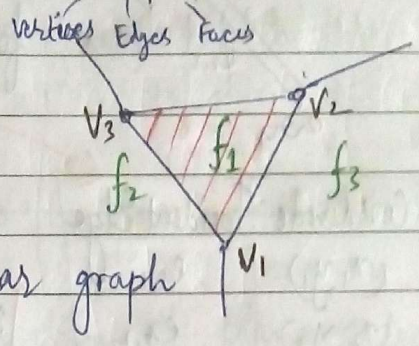
PAGE:
DATE: 07 / 08 / 17

$G = (V, E)$

Planar graph : if G can be represented on a plane.

$G = (V, E, F)$





Problem is how to represent a planar graph for performing queries efficiently.

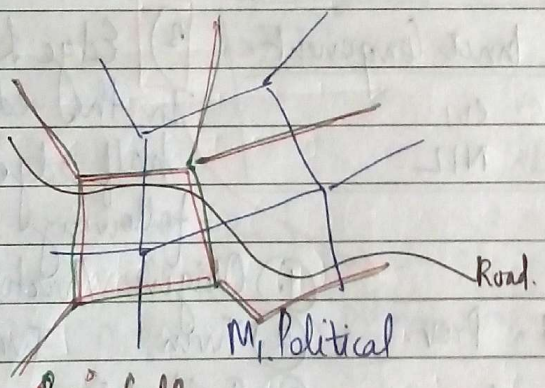
$V = \{v_1, v_2, \ldots, v_m\}$

$F = \{f_1, f_2, \ldots f_m\}$

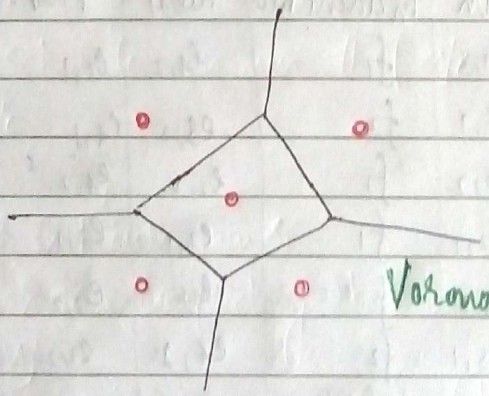
$f_i = \{v_{i_1}, v_{i_2}, \ldots v_{i_k}\}$ polygon.

Example.

Map Overlay Problem



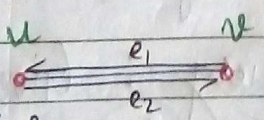
Subdivision Point Location / Faaity location Proble



Voronoi Diagram.

DCEL

Defⁿ:



half-edge $e_{u,v}$, $e_{v,u}$ ← Notations

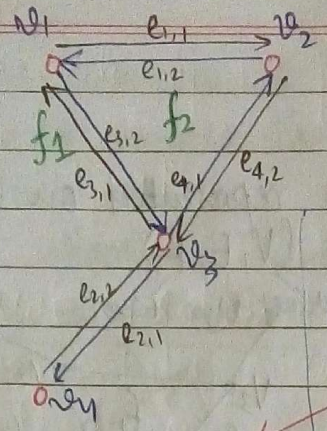
mutual twin

$Twin(e_{u,v}) = e_{v,u}$

$Twin(e_{v,u}) = e_{u,v}$

Orion

PAGE:
DATE: / /



Doubly Connected Edge List (DCEL)

It maintain following list / records.

1) Vertex List.

for each vertex v — 1. Coordinates (x_v, y_v)
2. Incident edge $\vec{e}_v$ where $\vec{e}_v$ originate at vertex v [choose $\vec{e}_v$ randomly]

VID	Coordinate	Incident Edge
v_1	(x_1, y_1)	$e_{1,1}$
v_2	(x_2, y_2)	$e_{4,2}$
v_3	(x_3, y_3)	$e_{2,1}$
v_4	(x_4, y_4)	$e_{2,2}$

2) Face List.

1. Outer Component (f) → An arbitrary half edge
2. Inner Component (f)

	FID	Outer Component	Inner Component
(Infinite face)	f_1	NIL	$e_{1,1}$
(Bounded face)	f_2	$e_{4,1}$	NIL

3) Edge List.

In the edge table, for each half edge $\vec{e}$ we have following info:

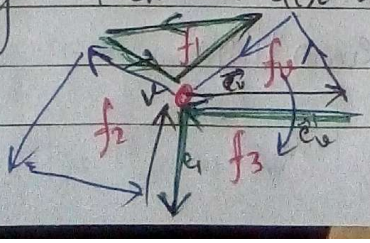
1. Origin // vertex at which $\vec{e}$ originates
2. Twin — Twin $(\vec{e}_{i,j}) = \vec{e}_{j,i}$
3. Incident Faces // a ptr. to an arbitrary edge of the incident face
4. Next // next half edge — Next $(e_{1,1}) = e_{4,2}$
5. Previous // ptr. to the previous half-edge — Prev $(e_{1,1}) = e_{3,2}$.

EID	Origin	Twin	Incident Face	Next	Prev.
$e_{1,1}$	v_1	$e_{1,2}$	f_1	$e_{4,2}$	$e_{3,1}$
$e_{1,2}$	v_2	$e_{1,1}$	f_2	$e_{3,2}$	$e_{4,1}$
$e_{2,1}$	v_3	$e_{2,2}$	f_1	$e_{2,2}$	$e_{4,2}$
$e_{2,2}$	v_4	$e_{2,1}$	f_1	$e_{3,1}$	$e_{2,1}$
$e_{3,1}$	v_3	$e_{3,2}$	f_1	$e_{1,1}$	$e_{2,2}$
$e_{3,2}$	v_1	$e_{3,1}$	f_2	$e_{4,2}$	$e_{1,2}$
$e_{4,1}$	v_2	$e_{4,2}$	f_2	$e_{1,2}$	$e_{3,2}$
$e_{4,2}$	v_3	$e_{4,1}$	f_1	$e_{2,1}$	$e_{1,1}$

Query: Given f_1 find its boundary

$e_{1,1}, e_{4,2} = \text{Next}(e_{1,1}) \dots$

Query: Given a vertex v, find out all the half edges originating at v



Let $\vec{e}_v$:= Incident Edge (v).

$\vec{e}'_v = \text{twin}(\vec{e}_v)$

Orion
PAGE:
DATE: / /

f_1 = Incident face $(\vec{ev})$
$\vec{e_1}$ = Next $(\vec{ev})$

Query: Given a line l and a face f such that L divide fs. Find all the faces cut by L.

08/08/17

DCEL is an elegant representation of planar subdivision or planar graph. $G = (V, E, F)$

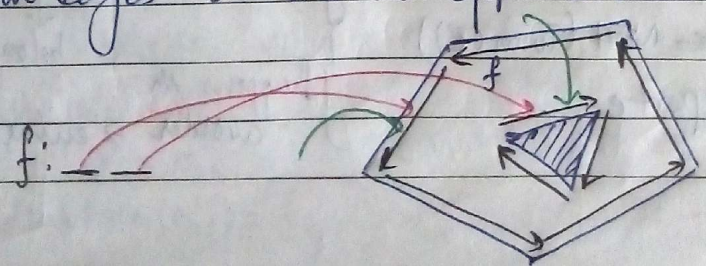
1) Vertex List
2) Face List
3) Edge List

Advantages:-
1) efficient traversal & queries
2) Linear Storage $O(|V|) + O(|E|) + O(|F|) = O(|V| + |E| + |F|)$
3) Allow dynamic updates of the data structures. addition/deletion of information.

Connected Graph
$|V| + |F| - |E| = 2$
$|F| = O(|V|)$
$|E| = O(|V|)$
} Property of planar graph

Applications:
1) Planar Straight Line Graph (PSLG)
2) Planar Subdivision ⟶ Map Overlay.
3) Voronoi Diagram

Remarks:- 1) Half edges are along the boundary of a face in anti-clockwise order. ⇒ face is on the left of an half edge.
2) Twin edges directed to opposite direction.



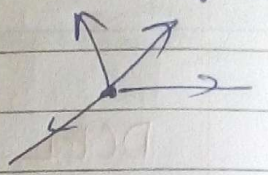
Orion
PAGE:
DATE: / /

3) For a hole, if the faces are in counter clockwise order, then the hole lies to the right of the boundary
4) Isolated vertex.

Traversal operations on DCEL

1) Walking on the boundary of a face
2) Access a face from an adjacent one.
3) Visit all the edges originating at a vertex.

ccw → finite face
cw → infinite face

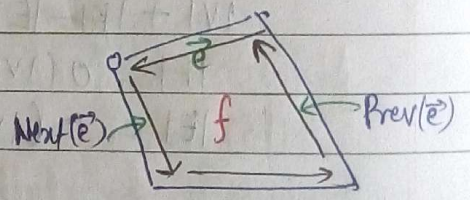


Complex Queries:

Given a line L & an half edge $\vec{e}$ s.t. L intersect e.
Find all the edges & faces intersected by L.

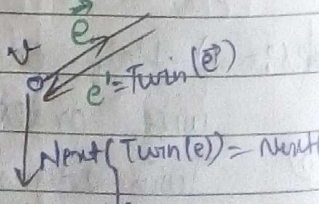
I. Traversing face (Trace the boundary)

Input f:
Let $\vec{e}$ ← Incident Edge ($\vec{f}$)
start_edge ← $\vec{e}$
while (Next(e) ≠ start_edge)
{
report $\vec{e}$
$\vec{e}$ ← Next($\vec{e}$)
}



II. Traversing incident edges of a vertex.

e ← Incident Edge (v)
start_edge ← $\vec{e}$
while (Next(Twin($\vec{e}$)) ≠ start_edge)
{
$\vec{e}$ ← Next(Twin($\vec{e}$))
report e
}



happens in ccw order ⇒ ensures no repetition

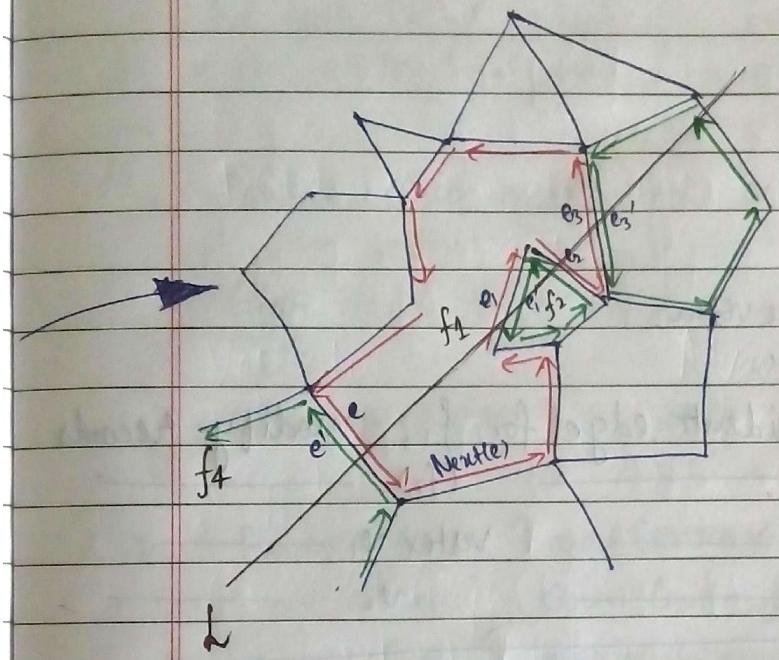
Orion

PAGE:
DATE: / /

Cond^{n} for intersection of line segment PQ with line L.

$(ax_p + by_p + c)(ax_q + by_q + c) \leq 0$

$ax + by + c = 0$



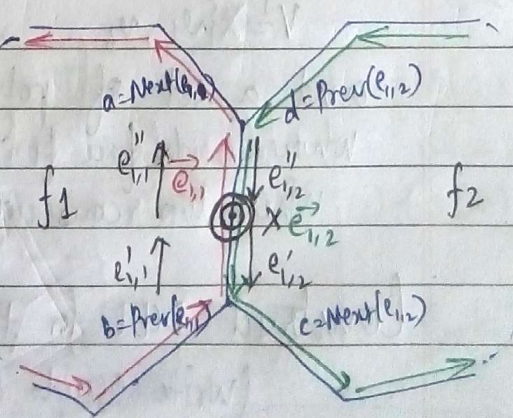
Report all the edges intersected by L by using Breadth First Search (BFS)

Let Q be a queue of edges intersected by L

1. Initialize $Q \leftarrow \phi$ (empty)
2. Enqueue $(Q, \vec{e})$ // $\vec{e}$ is given as intersecting edge
   Enqueue $(Q, Twin(\vec{e}))$
   Mark $\vec{e}$ & $Twin(\vec{e})$ as visited
3. while $(Q \neq \phi)$
   - $u \leftarrow$ Dequeue (Q) // u is an edge at the front of Q.
   - if L intersect u then
     - start_edge $\leftarrow u$.
     - $v \leftarrow$ Next (u)
     - while (Next(v) $\neq$ start_edge)
       - if v is not visited & $v \cap L \neq \phi$
         - report v, incident face(v)
         - Enqueue (Q, v), Enqueue(Q, Twin(v))

Adding a vertex on an edge

1. New vertex : x
2. New edges : $e'_{1,2}$, $e''_{1,2}$
3. Incident Edge(x) = $e_{1,2}$
4. Origin($e'_{1,2}$) = x
5. Prev($e'_{1,2}$) = $e''_{1,2}$
6. Incident face($e'_{1,2}$) = f_2



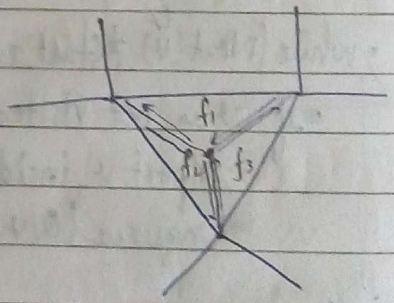
Orion
PAGE:
DATE: / /

7. Origin ($e''_{1,2}$) = Origin ($e_{1,2}$)
8. Next ($e''_{1,2}$) = $e'_{1,2}$
9. Prev ($e''_{1,2}$) = Prev ($e_{1,2}$)
10. Incident ~~edge~~ face ($e''_{1,2}$) = f_2
11. Next (Prev ($e_{1,2}$)) = $e''_{1,2}$
12. Prev (Next ($e_{1,2}$)) = $e'_{1,2}$
13. Delete $e_{1,2}$

: } for $e_{1,1}$
:

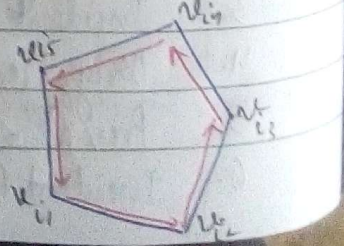
Twin ($e''_{1,1}$) = $e''_{1,2}$ & vice versa
Twin ($e'_{1,1}$) = $e'_{1,2}$ "
If $\vec{e}_{1,1}$ was set as incident edge for f_1, modify records.
— ($\vec{e}_{1,2}$) — (f_2) —
— $\vec{e}_{1,1}$ — vertex v_1 —
— $\vec{e}_{1,2}$ — v_2 —

* Delete Vertex :- Delete all twin edges, faces. (v_1 & v_2) (f_1, f_2, f_3)



* Delete Edge :- —

Ex Let a planar subdivision is given as a set of vertices
$V = \{v_1, v_2, \ldots, v_m\}$
and a set of polygonal faces $F = \{f_1, f_2, \ldots, f_n\}$
where each face f_i is an ordered set of its vertices $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ CCW order
where two consecutive vertex makes an edge.



Construct DCEL for this subdivision.
(Write pseudo code).

Orion
PAGE:
DATE: / /

* Process data face by face to extract edges

HW Prob 2. 10/08/17

Naive Algorithm inefficient.

for each (i, j), check if $i < j$ & $a_i > a_j$ $\quad 1 \le i \le j \le L$

$K_{C_2} = \frac{K(K-1)}{2} = O(K^2)$

Contrast is max when both ranks in opp. order.

Algo

1. Divide L into 2 parts A and B by middle element of L. (just like merge sort).
2. Find C_A = Count Contrast (A) $\quad T(K/2)$
   C_B = Count Contrast (B) $\quad T(K/2)$
   C_{AB} = merge & count (AB) $\quad O(k)$

1	5	4	8	10	2	6	9	3	7.

1	5	4	8	10

2	6	9	3	7.

(5,4) $C_A = 1$ (6,3) (9,3) (9,7) $C_B = 2$

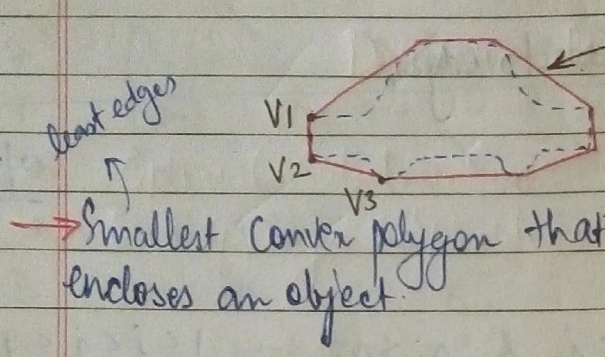
C_{AB} = (5,2) (5,3) (4,2) (4,5) (8,2) (8,6) (8,3) (8,7) (10,2) (10,6) (10,9) (10,3) (10,7)

$C = 17$.

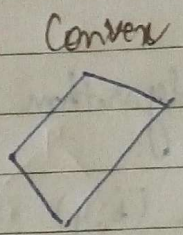
```
for (i = 1 .. n1)
    j = 1 .. n2
        if ai > bj
            ca += n1 - i + 1
            c[k] = bj  i++
```

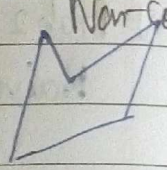
→ Time Complexity of Algorithm
Count contrast $14 = k$
$T(K) = 2T(K/2) + O(k)$
$T(k) = O(k \log k)$.

Orion
PAGE:
DATE: 14/08/17



→ Smallest convex polygon that encloses an object





*Consider any 2 points inside or on boundary of polygon, if line segment joining the points lies entirely in the polygon, it is convex polygon.
(Internal angles < 180°)

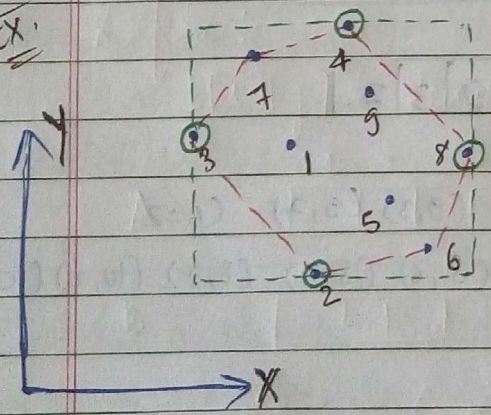
Input: A set of points in 2D: $P = \{p_1, p_2, \ldots, p_n\}$

Output: Vertices of the convex hull of P, taken in some order

Let CH(P) denotes the convex hull of P

$CH(P) \subseteq P$

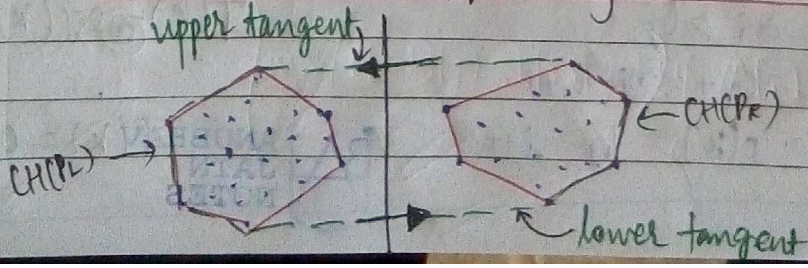
Ex:



Input → List of tuples
each tuple contains 2 entities

CH(P) = {3, 2, 6, 8, 4, 7}
in order
(cw or ccw)

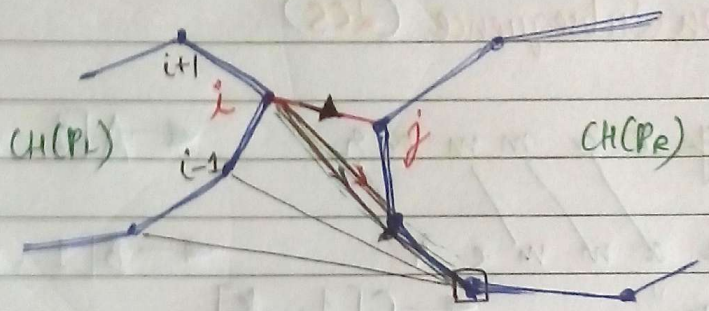
Divide and Conquer for Convex Hull



Orion
PAGE:
DATE: / /

Let P_L and P_R be the left and the right subsets of P obtained by partitioning.
That means $P_L \cup P_R = P$,
$P_L \cap P_R = \phi$, $-1 \leq |P_L| - |P_R| \leq 1$

Obs: $CH(P) = CH(P_L \cup P_R)$
$= CH(CH(P_L) \cup CH(P_R))$



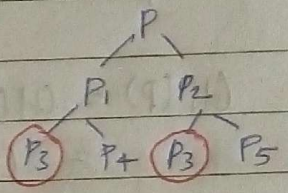
Time: $T(n) = 2T(n/2) + O(n)$

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

Orion
PAGE:
DATE: 17/8/17

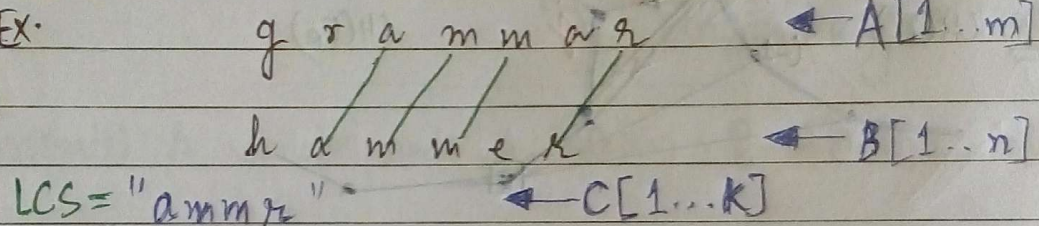
DYNAMIC PROGRAMMING

- Optimal Substructure
- Overlapping Subproblems



Longest Common Subsequence (LCS)

Ex.



1) $A[i] = B[j] \Rightarrow LCS(A[1..i], B[1..j])$ will contain $A[i]$ as its last character.

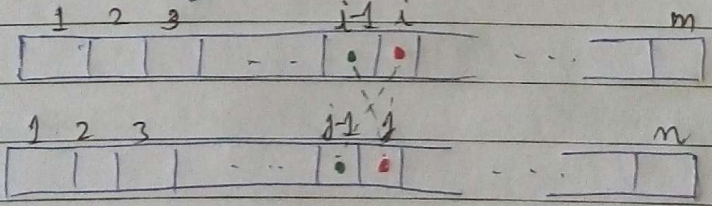
$$L[i][j] = L[i-1][j-1] + 1$$

2) $A[i] \neq B[j]$

$$L[i][j] = \max \begin{cases} L[i-1][j] \\ L[i][j-1] \end{cases}$$

Define :-

$L[i][j]$ = Length of LCS $(A[1..i], B[1..j])$



$$L[i][j] = \begin{cases} 0 & , \text{if } (i=0) \vee (j=0) \\ L[i-1][j-1] + 1 & , \text{if } (A[i] = B[j]) \wedge (i \neq 0) \wedge (j \neq 0) \\ \max(L[i-1][j], L[i][j-1]) & , \text{if } (A[i] \neq B[j]) \wedge (i \neq 0) \wedge (j \neq 0) \end{cases}$$

Orion
PAGE:
DATE: / /

	g	r	a	m	m	a	r
h	0	0	0	0	0	0	0
a	0	0	1	1	1	1	1
m	0	0	1	2	2	2	2
m	0	0	1	2	3	3	3
a	0	0	1	2	3	3	3
r	0	1	1	2	3	3	4

L =

All-pair Shortest Paths

21/08/14.

Input: A weighted graph $G(V, E, w)$ having no cycle with negative weight.

Output: d_{ij} = length of shortest path from vertex i to vertex j, where $1 \le i, j \le n$, n = number of vertices in V

Π_{ij} = shortest path from i to j.

Floyd and Warshall Algorithm

Observation – Define $V = \{1, 2, \ldots, n\}$

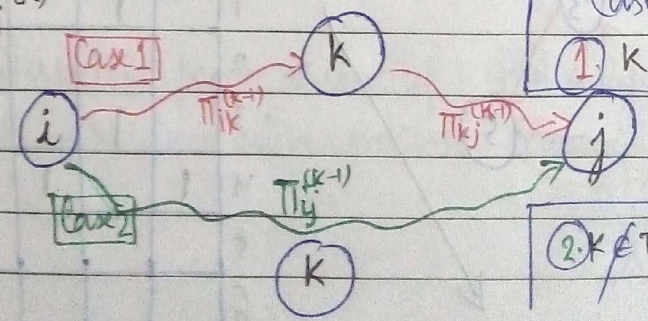
Define $V_k = \{1, 2, \ldots, k\} \subseteq V$

Consider any two vertices i & j from V

Consider all possible shortest paths from i to j whose intermediate vertices are only from V_k. Let this set be $P_{ij}^{(k)}$

Cases:-

1) $\Pi_{ij} \in P_{ij}^{(k)}$
2) $\Pi_{ij} \notin P_{ij}^{(k)}$



Cases:-

1. $k \in \Pi_{ij}^{(k)} \Rightarrow d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

2. $k \notin \Pi_{ij}^{(k)} \Rightarrow d_{ij}^{(k)} = d_{ij}^{(k-1)}$

min

ANUBHAV JAIN NOTES

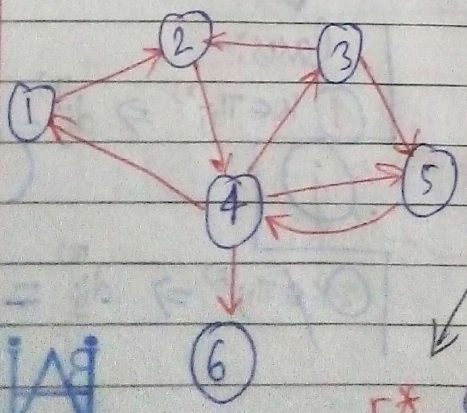
(Base case) $k=0 \Rightarrow d_{ij}^{(0)} = w_{ij}$ if $(i,j) \in E$
$= \infty$ otherwise

Algo

```
1) dij^(0) ← W (weight matrix of G)        // D[0][i][j]
2) for k ← 1 to n
3)     for j ← 1 to n
4)         for i ← 1 to n
5)             if dij^(k-1) > dik^(k-1) + dkj^(k-1)
6)                 dij^(k) ← dik^(k-1) + dkj^(k-1)        D[k][i][j]
7)             else dij^(k) = dij^(k-1)
8) return dij^(n)
```

```
1) D ← W
2) for k ← 1 to n
3)     for i ← 1 to
4)         for j ← 1 to
5)             if D[i][j] > D[i][k] + D[k][j]
6)                 D[i][j] ← D[i][k] + D[k][j]
7) return D
```

Complexity :-
Space - $O(n^2)$
Time - $O(n^3)$



E

	1	2	3	4	5	6
1		1				
2				1		
3		1			1	
4	1		1		1	1
5				1		
6						

$E^* = \{(i,j) : \text{there is a path from } i \text{ to } j \text{ in } G\}$

Orion

PAGE:
DATE: / /

$G(V,E) \longrightarrow G^*(V,E^*)$

G^* ↑ transitive closure of G → Floyd Warshall.

$G^{(2)} = (V, E^{(2)})$

$E^{(2)} = \{(i,j) : \text{there is a path of length at most 2}\}$.

22/08/17

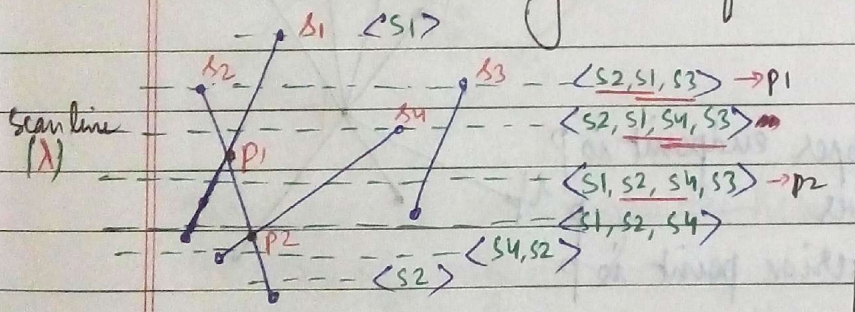
GIS (Geographical Information System)

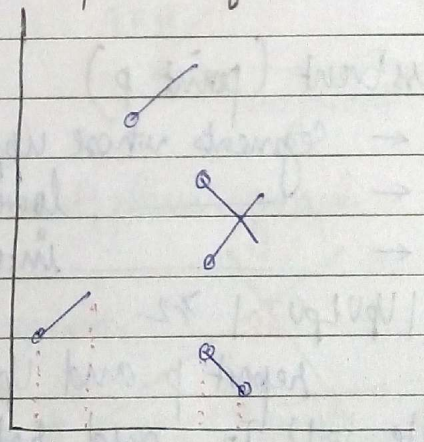
- Forest Map
- Road Map.

} overlay problem.

Output Sensitivity Algorithm → Time Complexity is a function of output size.

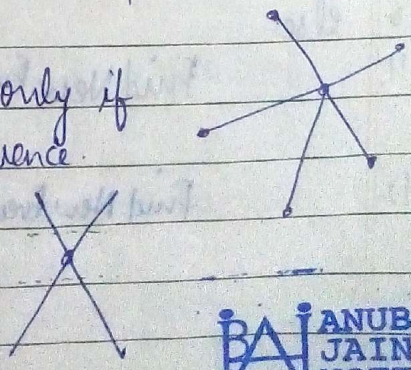
Given n line segments, find the point of intersections.





Obs 1: As the scan line moves down, the sequence of line segments intersected by the scanlines changes if and only if it encounters an (upper or lower) endpoint or an intersection point.

Obs 2: An intersection point arises if and only if two segments are consecutive in some sequence.



Orion

PAGE:
DATE: / /

Data Structures:

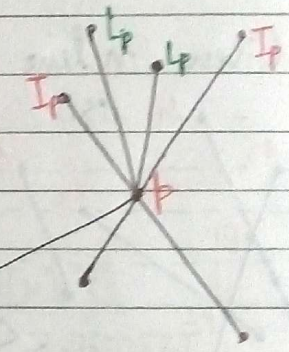
1) Event points – Upper endpoints, lower endpoints, intersection points.

2) Sequence of line seg. intersected by l. → AVL Tree: T

Event Queue – Height Balanced Binary Search Tree ✓ (AVL Tree) : Q
Binary Max Heap ✗

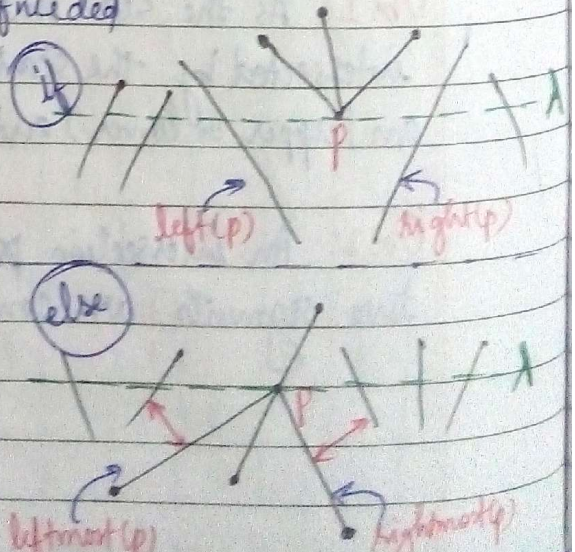
Find Intersections (Line set S)

1. Build Q using the endpoints of segments in S (priority = max y)
2. Initialize T as empty.
3. while Q is not empty.
4. p ← dequeue (Q) //delete the point with max y in Q.
5. Rebalance Q, if needed
6. Process Event (p)



Process Event (point p)

1. U_p ← segments whose upper endpoint is p
2. L_p ← _____ lower _____ p
3. I_p ← _____ interior point is p
4. if $|U_p \cup L_p \cup I_p| \geq 2$
5. report p and U_p, L_p, I_p
6. delete $L_p \cup I_p$ in T and rebalance T if needed
7. insert $U_p \cup I_p$ in T and rebalance T, if needed
8. if $(U_p \cup I_p = \phi)$
9. Find New Event (left(p), right(p), p)
10. else
11. Find New Event (left(leftMost(p)), leftMost(p), p)
12. Find New Event (right(rightMost(p)), rightMost(p), p)



Orion
PAGE:
DATE: / /

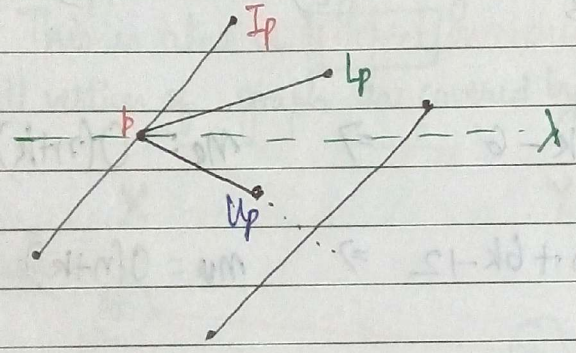
FindNewEvent (s_1, s_2, p)

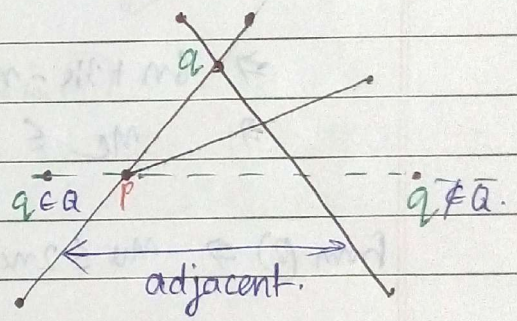
1. q ← intersection between s_1 & s_2
2. if q lies right of p or above p & $q \notin Q$
3. insert q in Q

24/08/17

Plane Sweep Algorithm
for Line Segment Intersection

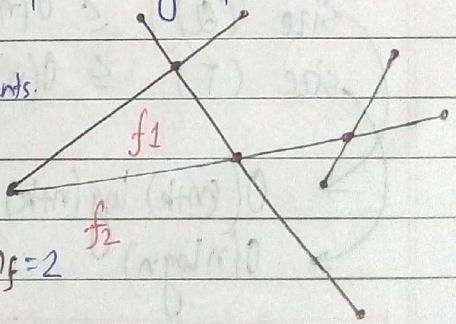
Input: n line segments.





n_v = no. of vertices in the resultant planar graph
n_e = no. of edges
n_f = no. of faces.

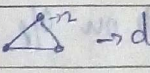
n = no. of line segments. = 4
K = #intersections = 4
$n_v = 10$, $n_e = 10$, $n_f = 2$



Maximum no. of vertices possible = K + 2n

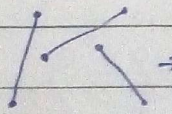
$\Rightarrow n_v \leq 2n + k$. —— (1)

~~Max~~ $\sum_v$ degree(v) $= 2n_e$



→ degree = 2×3 = 6 = $2n_e$.

Min $\sum_v$ degree(v) $= n_v$.



→ degree = 6 = n_v.

$\Rightarrow n_v \leq 2n_e$ —— (2)

Orion
PAGE:
DATE: / /

degree of a face = # edges defining it.

$n \geq 3$

$\deg(f) \geq 3 \Rightarrow \sum_f \deg(f) \geq 3n_f$

Every edge belongs to atmost 2 faces. $\Rightarrow \sum_f \deg(f) \leq 2n_e$.

$3n_f \leq 2n_e$ ——— (3)

Euler's formula for a planar graph :-

$n_v - n_e + n_f \geq 2$ → connected graph.

$2n + k - n_e + \frac{2n_e}{3} \geq 2$. [from (1) and (3)]

$\Rightarrow 6n + 3k - n_e \geq 6$

$\Rightarrow n_e \leq 6n + 3k - 6 \Rightarrow n_e = O(n+k)$

From (2) $\Rightarrow n_v \leq 2n_e \leq 12n + 6k - 12 \Rightarrow n_v = O(n+k)$

Time Complexity :

Size (Q) $\leq O(n+k)$

Size (T) $\leq O(n)$

$O((n+k)\log(n+k)) = O((n+k)\log n)$

$O(n \log n)$

Total = $O((n+k)\log n)$

↓

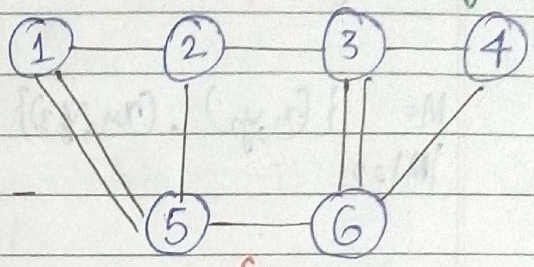
depends on o/p size ⇒ o/p sensitive Algo.

Orion
PAGE:
DATE: 28/08/17

Matching

Given an undirected graph $G(V, E)$, a matching M is a subset of E such that no two edges in M have a common vertex. A matching is, therefore, also called an independent edge set

$M = \{(1,5), (3,6)\}$ is a maximal matching
→ (cannot add any edge from E-M to M)

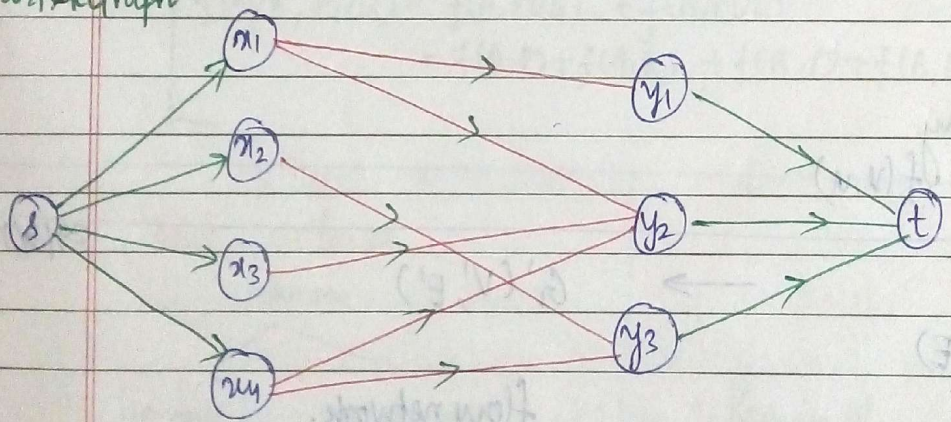


G

$M = \{(1,2), (3,4), (5,6)\}$ is a maximum matching (hence also maximal)
→ (max. possible cardinality among all maximal matchings)
This is also a perfect matching
→ (all vertices of graph are covered by edges in M.)

X Y

Bipartite Graph



$G = (V = (X \cup Y), E = \{(x,y) : (x \in X), (y \in Y)\})$

→ $G' = (V', E')$, $V' = V \cup \{s, t\}$, $E' = E \cup \{(s,x) : x \in X\} \cup \{(y,t) : y \in Y\}$,
$c(u,v) = 1 \ \forall \ (u,v) \in E'$

Lemma: M is a matching in G if and only if f is a flow in G' such that $|f| = |M|$
(|f| → value of flow.)

Orion

PAGE:
DATE: / /

$\Rightarrow \max |f| = \max |M|$

maximum

Integrality Theorem $\Rightarrow |f|_{max}$ = an integer (as all $c(u,v) = 1$ int.)

$M = \{(x_1, y_2), (x_4, y_3)\}$

$|M| = 2$

	Generalized
$f(s,x_1) = f(s,x_4) = 1$	$f(s,x) = 1$ if $(x,y) \in M$
$f(s,x_2) = f(s,x_3) = 0$	$f(s,x) = 0$ if $(x,y) \notin M$
$f(y_2,t) = f(y_3,t) = 1$	$f(y,t) = 1$ if $(x,y) \in M$
$f(y_1,t) = f(y_4,t) = 0$	$f(y,t) = 0$, if $(x,y) \notin M$
$f(x,y) = 1 \ \forall (x,y) \in M$	
$f(x,y) = 0 \ \forall (x,y) \in E \setminus M$	

Capacity constraint : $f(u,v) \leq c(u,v) \quad \forall (u,v) \in V^2$

Each $(u,v) \in E'$ has a flow 0 or 1 and hence obey this property

Skew symmetry

$f(u,v) = -f(v,u)$

29/8/17.

$G_1 \longrightarrow G_1'(V', E')$

(V, E)

$V = X \cup Y$

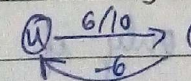
bipartite graph

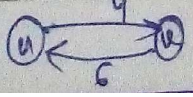
flow network.

M $\Leftrightarrow$ f such that $|M| = |f|$.

1st ed.

$f(v,u) = -f(u,v)$

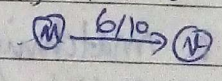


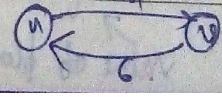


Residual = (same)

New ed.

$f(v,u) = 0$





Orion
PAGE
DATE / /

Flow-cut Lemma

$f(S,T) = |f|$ for any cut (S,T) with $s \in S$ and $t \in T$.

Let $S = \{s\} \cup X$ and $T = \{t\} \cup Y$.

$$f(x,y) = \begin{cases} 1 & \text{if } (x,y) \in M \\ 0 & \text{if } (x,y) \notin M \end{cases}$$

So,
total flow is $|f| = |M| \times 1 = |M|$

$|M| = f(X,Y)$
$= f(\{s\} \cup X, \{t\} \cup Y)$
$- f(s,t) - f(s,Y) - f(X,t)$
$= f(S,T) = |f|.$

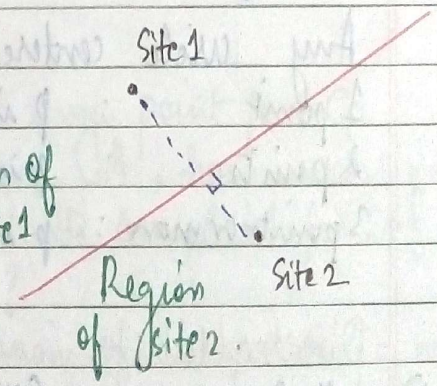
$f(A \cup B, C) = f(A,C) + f(B,C)$
$f(A \cup B, P \cup Q) = f(A, P \cup Q) + f(B, P \cup Q)$
$= f(A,P) + f(A,Q) + f(B,P) + f(B,Q)$

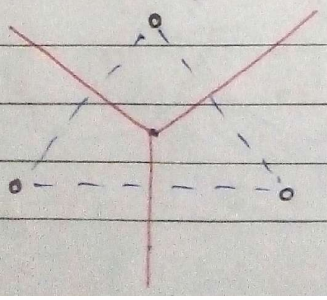
Edit distance

hammr

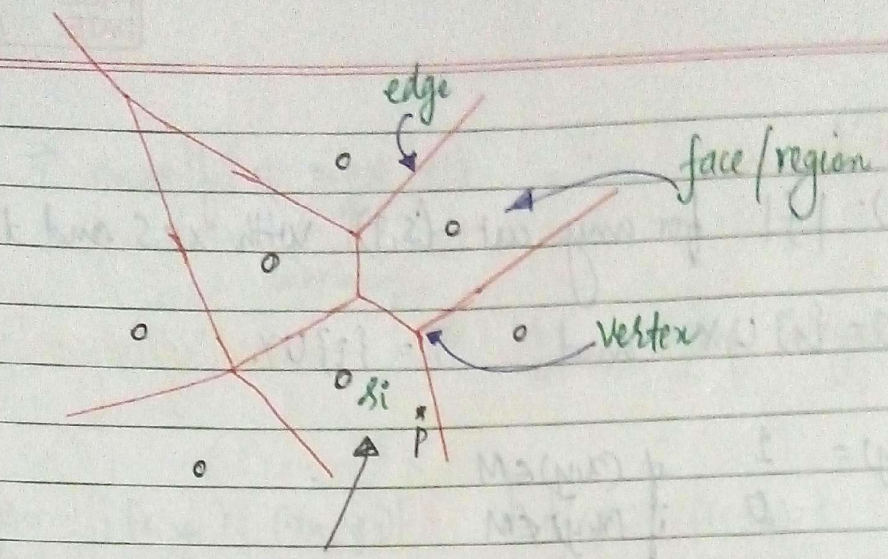
hammer grammar

hammar





Voronoi Diagram



$$R(s_i) = \{ p \in \mathbb{R}^2 : dist(p, s_i) \leq dist(p, s_j) \; \forall j \}.$$

Site corresponds to one face.
$O(n)$ faces. $\Rightarrow$ vertices & edges $= O(n)$.

$n_v = O(n)$
where n = # sites.

Observation:-
Largest empty circle centered at a point p.

Any circle centered at p, if circle passes through
1 point : p is in the region of a site
2 points : p is on the edge
3 points or more : p is the vertex.

Observations:-
1) # faces = n (# sites, i.e. input size)
2) # vertices of VD, $n_v = O(n)$
3) # edges of VD, $n_e = O(n)$

Orion
PAGE:
DATE: / /

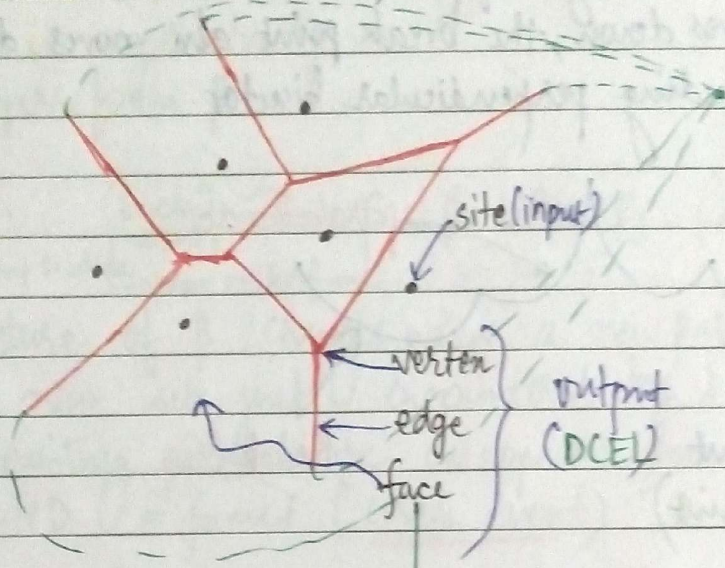
Proof:-
After extending and joining the open edges with V_a (a vertex added sufficiently far from VD), the resultant graph becomes a connected planar graph. So, we can use Euler's formula

$$(n_v+1) - n_e + n = 2 \quad \text{— (1)}$$

$$\deg(v) \geq 3 \ \forall v \Rightarrow \sum_v \deg(v) \geq 3(n_v+1)$$

$$\Rightarrow 2n_e \geq 3(n_v+1) \quad \text{— (2)}$$

$$\Rightarrow n_v \leq 2n-5, \ n_e \leq 3n-6 \Rightarrow n_v = O(n), \ n_e = O(n)$$

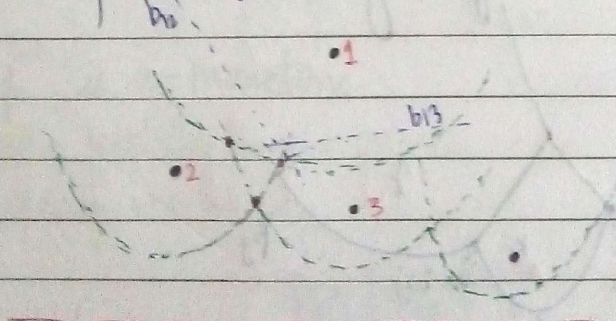


also called Voronoi cell

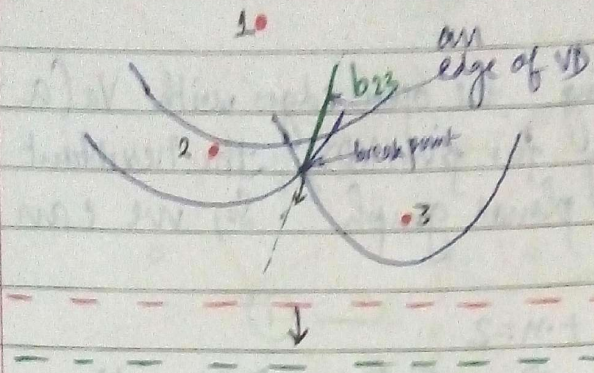
Observations:-

4) Largest empty circle centered at any point p : C_p

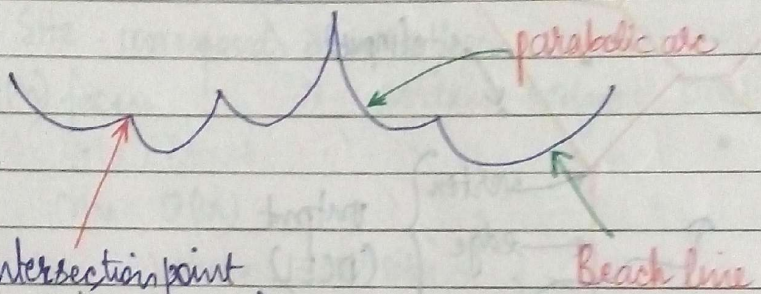
5) Question- For a given sweep line (λ), how much portion of VD lying above λ has the VD completed for it (irrespective of the sites below λ)?

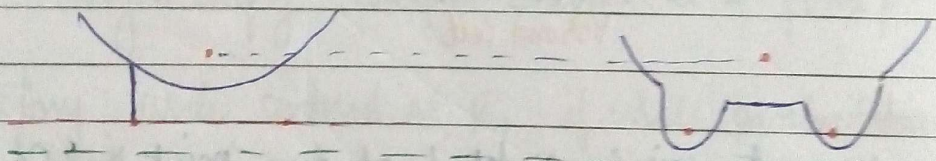


Union of the parabolic regions. Each parabola is defined by a site (above λ) as the focus and λ as the directrix

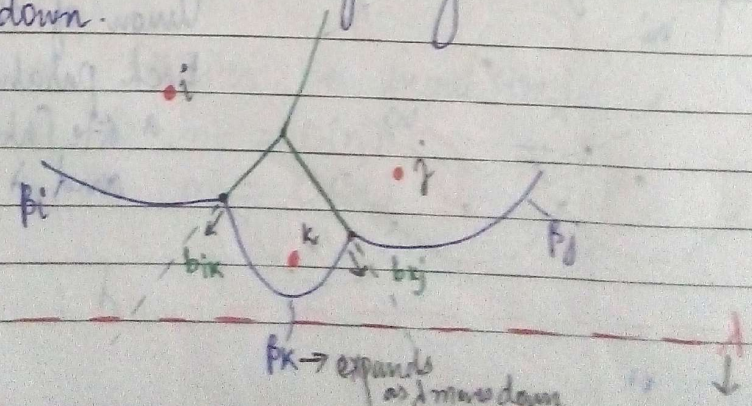


6) What does an intersection point of two parabolas signify?
It traces an edge of VD
As λ moves down, the break point also moves downward along the corresponding perpendicular bisector



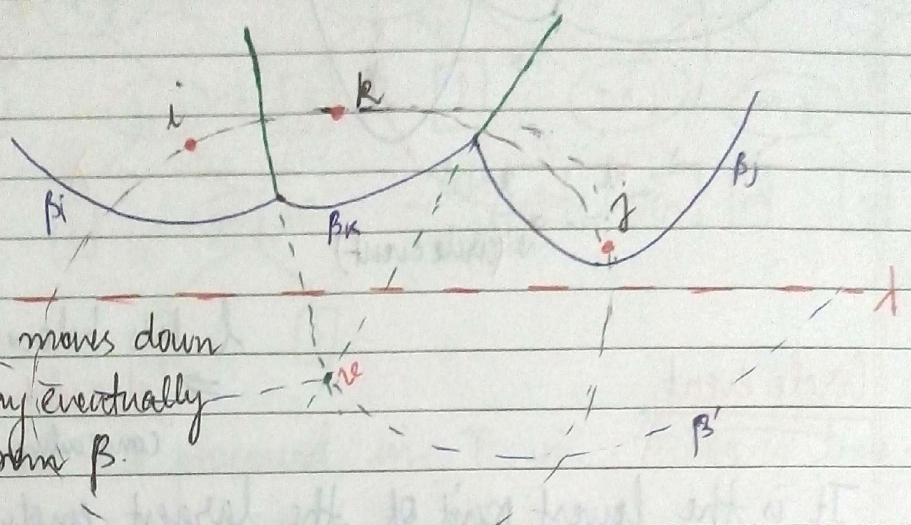


β = Beach line → boundary of the Voronoi region computed so far and it will not change anymore as the sweep line λ moves down.



Orion
PAGE:
DATE: / /

As sweep line l moves down,
Break points move along perpendicular bisector and the parabolas opens up.

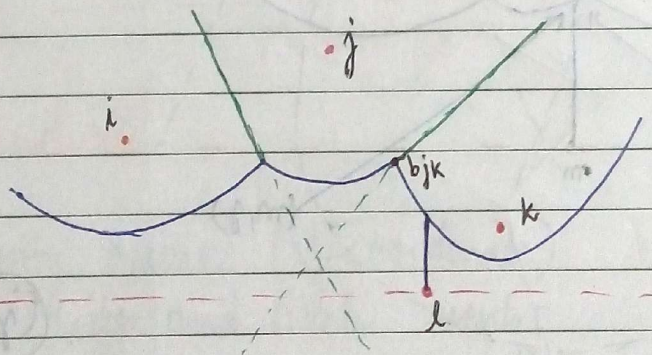


β_k shrinks as l moves down and so it may eventually disappear from β.

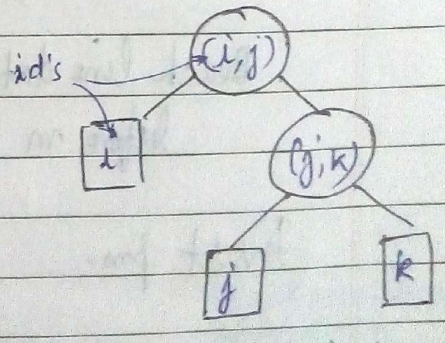
{Circle event} ⟺ $\beta = \beta_i \cup \beta_k \cup \beta_j$ (in left-to-right order)
↓ tangent to circle → new vertex found.

2) Structure of β changes when a new parabola participates due to a new site just encountered by l (site event) or an existing parabolic arc disappears from β and new vertex of VD is formed (circle event)

Site Event

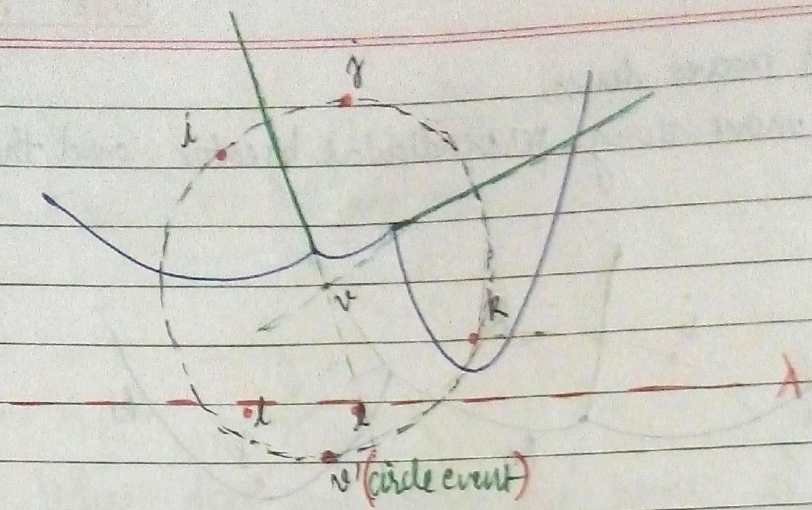


3) β is x-monotone.
(every vertical line intersects it in exactly one point).



Height-balanced BST storing β

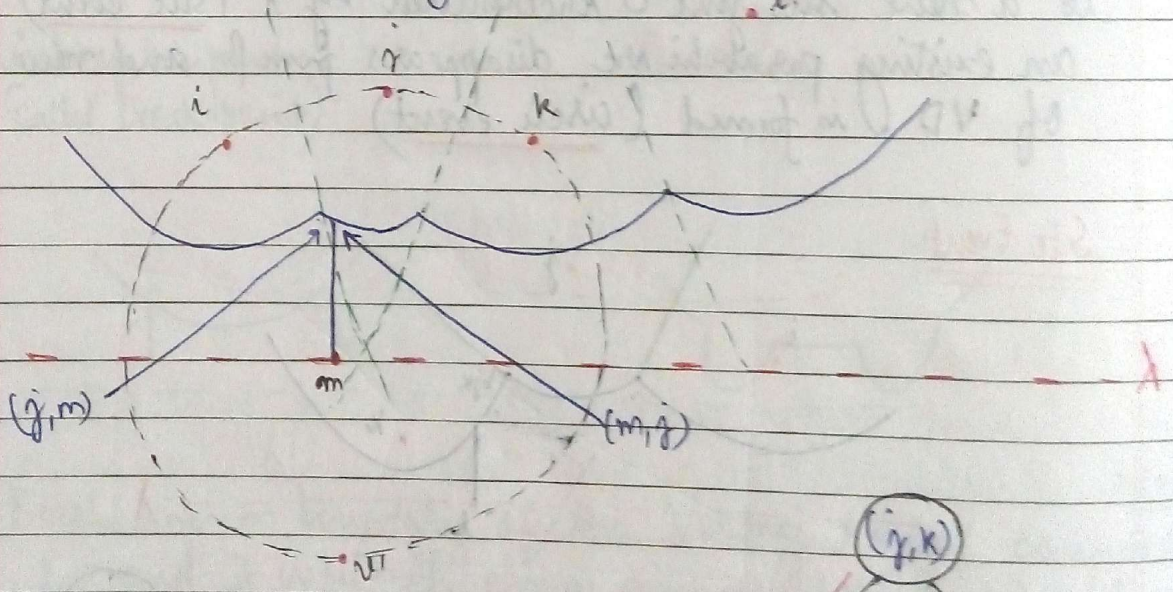
ANUBHAV JAIN NOTES



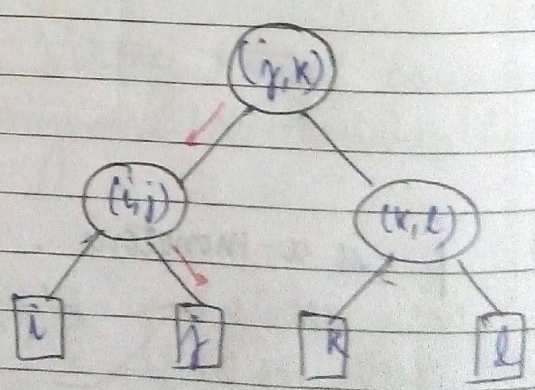
l lies below l but above v'.
⇒ false alarm (if βi, βj, βk are not consecutive by the time l almost reaches v')

Circle event

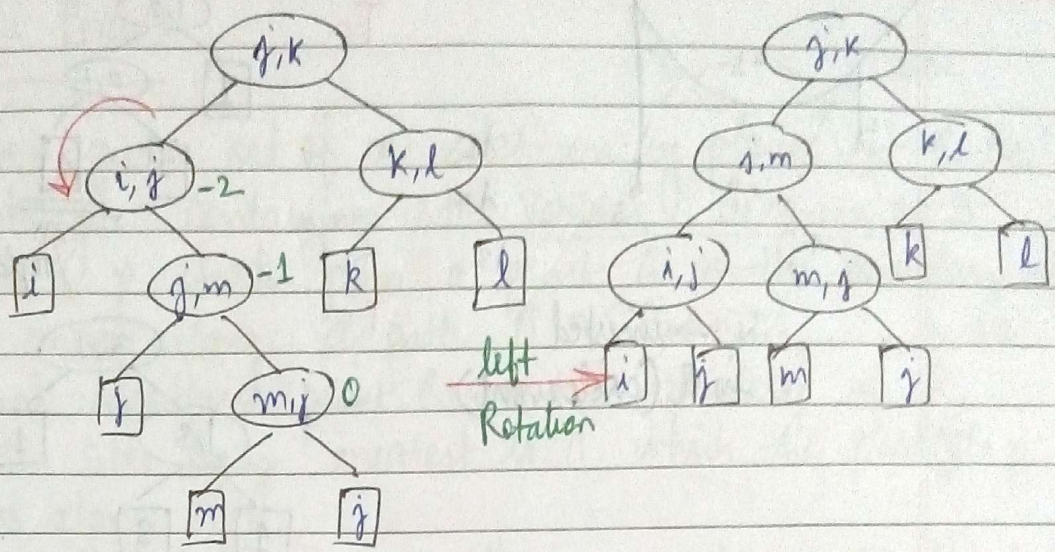
It is the lowest point of the largest empty circle passing through three sites (above l) whose corresponding parabolic arcs are consecutive in the beach line.
A circle event always lies below l



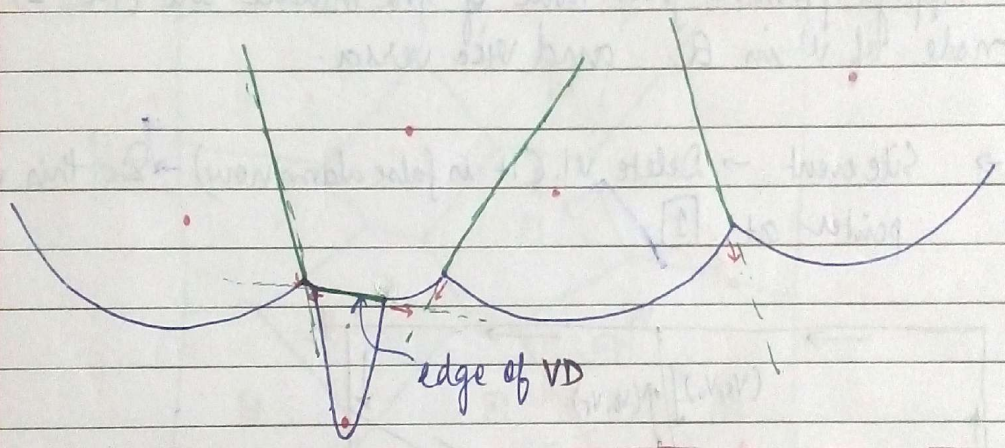
Beach line data structure (T)
before m is processed.



Insert pm.



⇒ Site event is processed in T in O(log n) time.



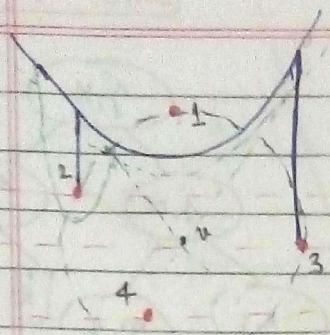
{→ Indicates future movement of break points}

Data Structures:

1) Q - priority queue (y-coordinate)
stores all sites and circle events

2) T - beach line

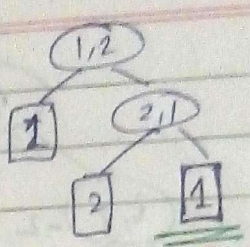
3) DCEL - output VD



λ_1
λ_2
λ_3

'v' → inserted in Q (circle event)

T=

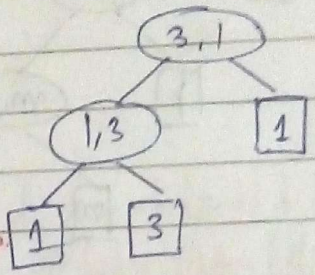


DCEL

V = ϕ
E = { (1,2)(2,1) } ← break point, endpts of edge
F = ϕ

replaced by

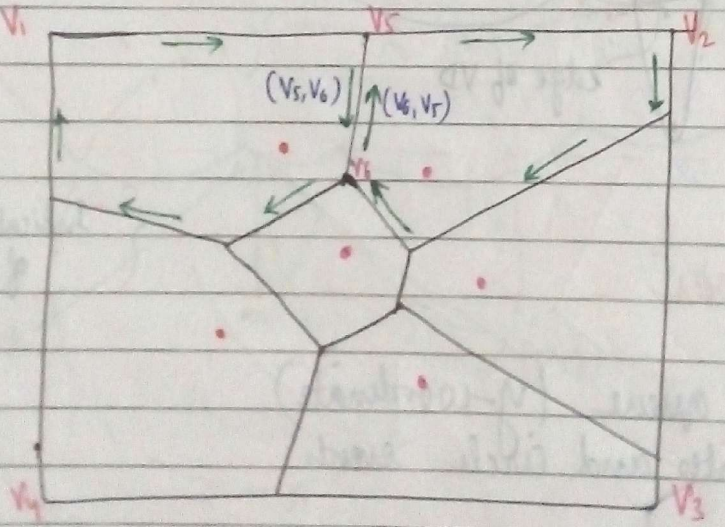
T=



(2,1,3) → triplet = circle event v1 in Q.

⇒ Apply pointers from node of the middle arc (i.e, 1) to the node of v1 in Q and vice versa.

λ_3 → Site event → Delete v1. (it is false alarm now) → Do this using pointer at 1.



Orion
PAGE:
DATE: 07/09/17

Tutorial - Prob 3

DEEL

Let S be a given set of n sites on xy plane. Let R be the rectangle containing the Voronoi diagram of S. A particle p starts from a point A on the boundary of R and moves along a path $\Pi = \langle A, A_1, A_2, \ldots, A_m, B \rangle$ to reach another point B on the boundary of R. Find the sites lying nearest to Π which the particle p moves along Π.

Assume that between every two consecutive points (A_i, A_{i+1}) in Π, the path is linear.