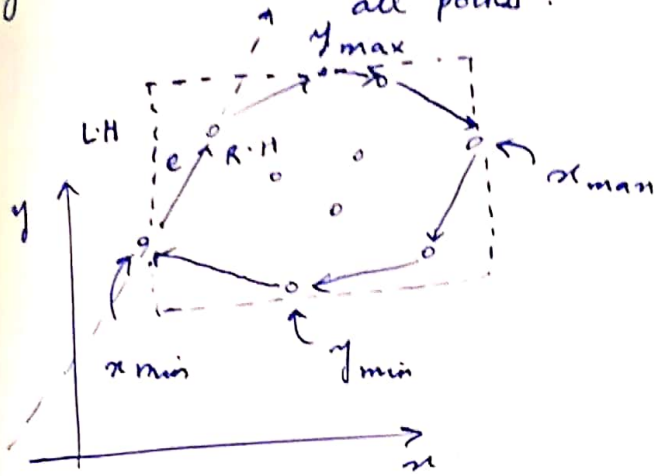
Polygon with minimum perimeter, containing all points :



1) $V(P) \subseteq S$

2) $S \cap LH(e) = \phi$

(left half)

$\forall\ e \in P$

$\Downarrow$

$p, q \in S$

$\overrightarrow{pq}$ is an edge of P

$\Rightarrow S \cap LH(\overrightarrow{pq}) = \phi$

S = input point set having n points

P = min. perimeter polygon containing all points of S

LH(e) = left half plane w.r.t e

RH(e) = right half plane w.r.t e

1st method : take every pair of points and check whether all points lie on one side (brute force).

$\binom{n}{2} = O(n^2)$ pairs of points in S

checking every pair needs $O(n-2) = O(n)$ time

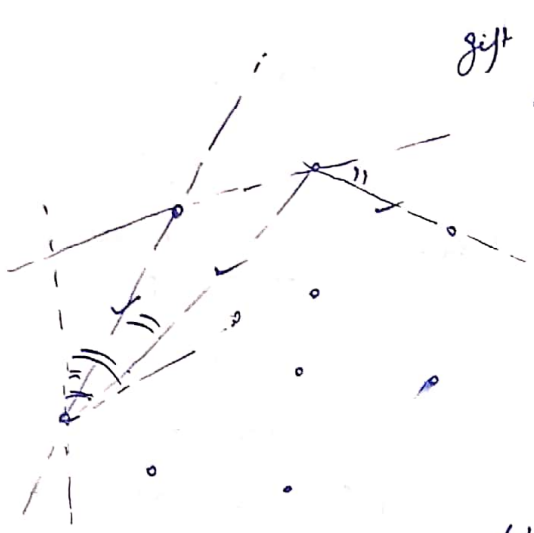
∴ Total time $= O(n^2) \cdot O(n) = O(n^3)$

Gift wrapping algorithm — $O(nh)$

(where h = # vertices of P)

$O(n^2)$ if $h = O(n)$

2nd method : start from leftmost point then check the polar angle of every other vertex w.r.t the y-axis, the min. polar angle vertex is another vertex of the polygon. Now find polar angle of all points w.r.t the new edge. min. polar angle point becomes a vertex. Proceed likewise.



Longest Common Subsequence (LCS)

Let $A[1 \ldots m]$ and $B[1 \ldots n]$ be two strings containing m & n characters respectively. $C[1 \ldots p]$ is said to be an LCS of $A[1 \ldots m]$ and $B[1 \ldots n]$ if:

1) $C[k] = A[i] = B[j]$ for $1 \le k \le p$
   for some $i \in [1, m]$ and some $j \in [1, n]$, and
   $C[1 \ldots k-1] = LCS(A[1 \ldots i-1], B[1 \ldots j-1])$

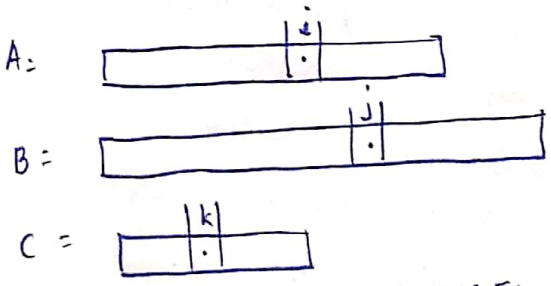
2) Value of p is maximum.

Ex:

A = | D | R | A | M | A | (1 … m)

B = | G | R | A | M | M | A | R | (1 … n)

C: LCS(A,B) = | R | A | M | A | (1 … p)



$C[1 \ldots k-1] = LCS(A[1 \ldots i-1], B[1 \ldots j-1])$

if $C[k] = A[i] = B[j]$

Dynamic Programming (DP)

1) Optimal substructure

- $C[p] = A[m] = B[n]$ if $A[m] = B[n] \Rightarrow C[1..p-1] : LCS(A[1..m-1], B[1..n-1])$
- $A[m] \neq B[n] \Rightarrow$ one of the following is true :

a) $C[p] = A[m] \Rightarrow C : LCS(A[1...m], B[1...n-1])$

b) $C[p] = B[n] \Rightarrow C : LCS(A[1..m-1], B[1...n])$

c) $C[p] \notin \{A[m], B[n]\} \Rightarrow C : LCS(A[1...m-1], B[1...n-1])$

Overlapping sub-problems :

$LCS(A[1...i], B[1...j])$

$= LCS(A[1...i-1], B[1...j-1]) \cup \{A[i]\}$ if $A[i] = B[j]$

let $L[i][j] = |LCS(A[1...i], B[1...j])|$ = length

$$\Rightarrow L[i][j] = L[i-1][j-1] + 1 \quad \text{if } A[i] = B[j]$$

$$= \max\{L[i][j-1], L[i-1][j], L[i-1][j-1]\}$$

o.w.

Tutorial Problem - 1

Let $A[1...m]$ and $B[1...n]$ be two 1D arrays containing m and n integers respectively, where $m \leq n$.

We have to construct a sub-array $C[1...m]$ of B such that $\sum_{i=1}^{m} |A[i] - C[i]|$ is minimized.

Develop the DP recurrences needed for DP with clear arguments.

Design the algorithm & write the pseudocode.

Demonstrate your algorithm on a few input instances

Derive its time and space complexity.

Matrix chain multiplication

$A_1 \cdot A_2 \cdot A_3 = (A_1 A_2) A_3$ or $(A_1)(A_2 A_3)$

$A_1 = 20 \times 10$, $A_2 = 10 \times 5$ $\rightarrow$ $A_{12} = 20 \times 5 \times 10$ (operations)

$A_3 = 5 \times 30$ $\rightarrow$ $A_{123} = 5 \times 20 \times 30$ (operations)

$\therefore$ Total # of (scalar) multiplications $= 1000 + 3000 = 4000$

If other way is followed.

$A_1 (A_2 A_3)$ here $\boxed{n = 3,\ k = 2}$

Cost for this multiplication:

$10 \times 5 \times 30 + 20 \times 10 \times 30 = 6000 + 1500 = 7500$

$\Rightarrow$ Min. cost multiplication is for $(A_1 A_2) A_3$.

Now consider n matrices:

$\boxed{A_1, A_2, \ldots A_n}$

$(A_1 A_2 \ldots A_k)(A_{k+1}, \ldots, A_n)$

$\rightarrow (\times) \leftarrow$

Final product

To minimise total number of scalar multiplications,

$k = ?$ $\quad [1 \le k \le n-1]$

For 4 matrices: $A_1 A_2 A_3 A_4$

$k=1$: $A_1(A_2 A_3 A_4) \longrightarrow A_1((A_2 A_3) A_4)$ || $A_1(A_2(A_3 A_4))$

$k=2$: $(A_1 A_2)(A_3 A_4)$

$k=3$: $(A_1 A_2 A_3)(A_4) \longrightarrow (A_1(A_2 A_3)) A_4$ || $((A_1 A_2) A_3) A_4$

Define:

$m(i,j)$ = min no. of multiplications for $A_i A_{i+1} \ldots A_{j-1} A_j$

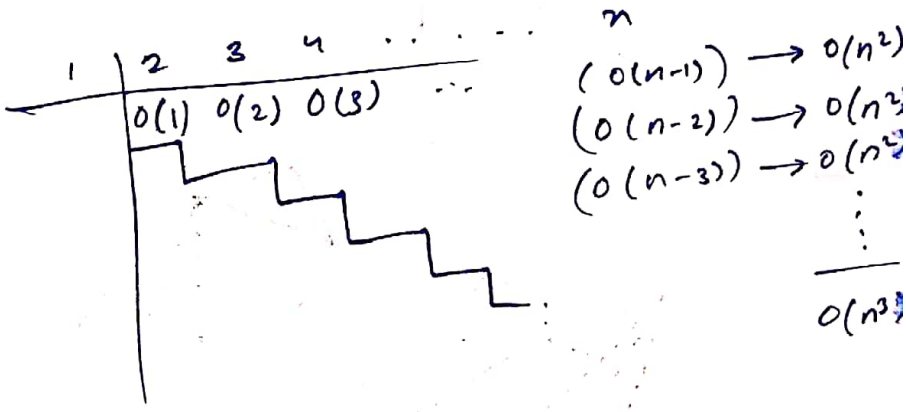
$$m(i,j) = \min_{i \le k \le j-1} \{ m(i,k) + m(k+1,j) + r_{i-1} \times r_k \times r_j \}$$
$$= 0 \quad \} \; i \le j$$

example :

$A_1 = 5 \times 10$
$A_2 = 10 \times 5$
$A_3 = 5 \times 5$
$A_4 = 5 \times 20$
$A_5 = 20 \times 10$

	1	2	3	4	5
1	0	250			
2		0	250		
3			0	500	
4				0	1000
5					0

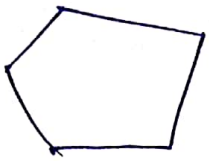
Time complexity analysis :



Minimum ink triangulation of a convex polygon. (P)

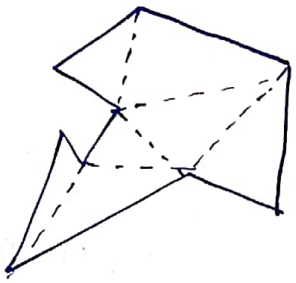
Def: A polygon is said to be CONVEX if the line segment joining any two vertices of the polygon lies inside the polygon.

Ex:

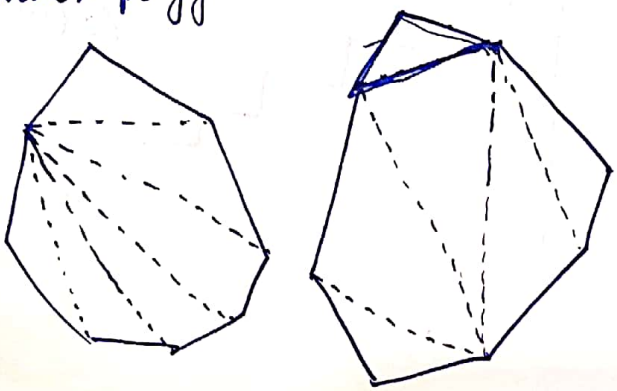


Triangulation: of a polygon is the collection of triangles obtained by adding diagonals such that the intersection between the interiors of any two triangles is empty.

e.g.



for a convex polygon:



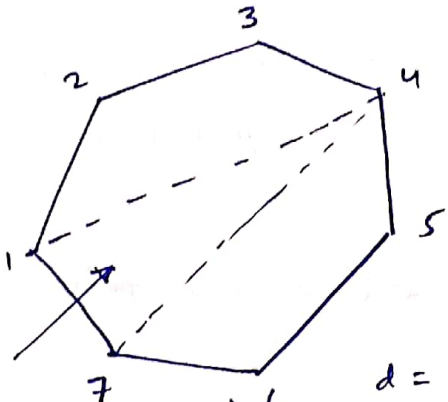
Minimum ink: Sum of lengths of the diagonals is minimum.

Observations:

(i) # triangles = $n-2$, where n is the no. of vertices of P

(ii) # diagonals = $n-3$

(iii) Each edge of P will be an edge of some triangle in the triangulation (T), whether T is min-ink or not.



$w(1,7) = d(1,4) + d(7,4)$ $\quad d$ = Euclidean distance.

(iv) Every diagonal of T is incident on exactly 2 triangles of T.

vertex 'i' and vertex 'n',

→ $E(1,n)$

A triangle in T with edge $(1,n)$ has the third vertex as k, where $2 \le k \le n-1$

Define: $f(1,n)$ = sum of lengths of the diagonals in T.

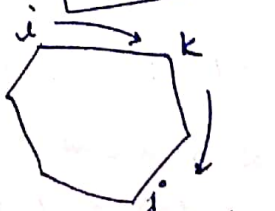
$$f(1,n) = \min_{2 \le k \le n-1} \{ f(1,k) + f(k,n) + w(t(1,k,n)) \}$$

(min-ink)

$t(1,k,n)$ → triangle with vertices 1, k, n

optimal substructure of DP

↳ overlapping subproblems.



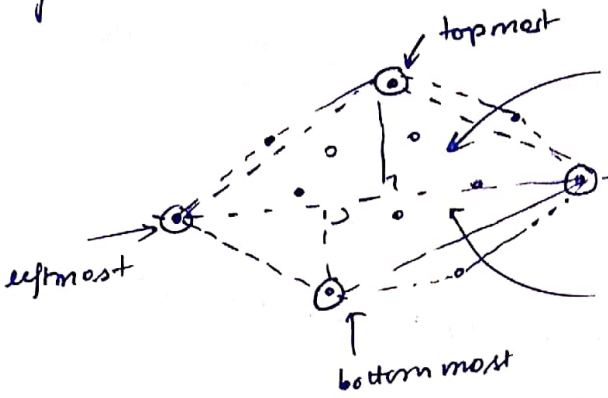
$$f(i,j) = \begin{cases} 0 & \text{if } j - i \le 1 \\ \min\limits_{j+1 \le k \le j-1} \{ f(i,k) + f(k,j) + w(t(i,k,j)) \} \end{cases}$$

NOTE: This problem is very similar to matrix chain multiplication.

Convex Hull :

Let P be a set of 2-dimensional points. The convex hull $CH(P)$, P is the smallest perimeter polygon that contains all points of P.

The vertices of CH(P) are described in a specific order, say clockwise.



Quick Hull

note, after drawing the first horizontal line (passing through leftmost & rightmost) : those points which are at the farthest distance from the line must belong to the convex Hull. Proceed recursively

Balanced partition : lies within some bounds say, $\frac{1}{10}$ to $\frac{10}{1}$

Time complexity :

$$T(n) = T(\alpha n) + T((1-\alpha)n) + O(n)$$

α is a constant.

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

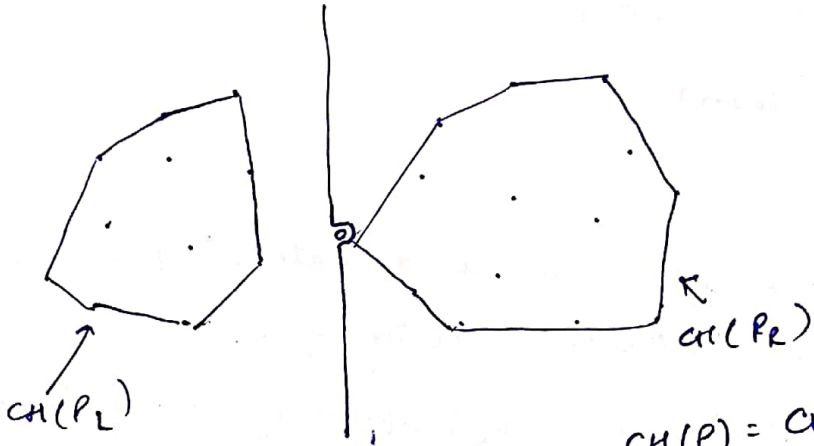
Worst Case :

$$T(n) = T(1) + T(n-2) + O(n)$$

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

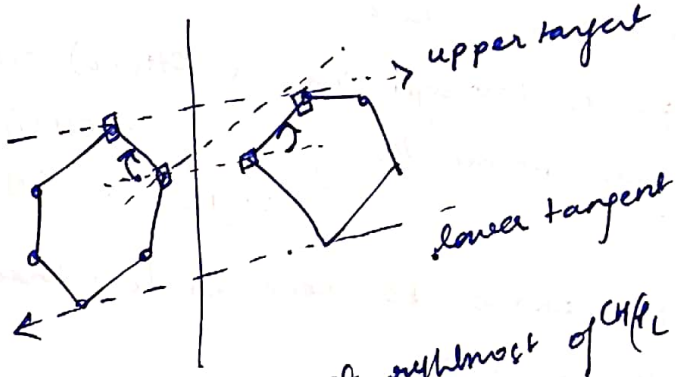
$$\Rightarrow T(n) = O(n^2)$$

CH(P) by divide and conquer:



$CH(P) = CH(P_L \cup P_R)$
$\subseteq CH(P_L) \cup CH(P_R)$

ensures
$abs(|P_L| - |P_R|) \leq 1$

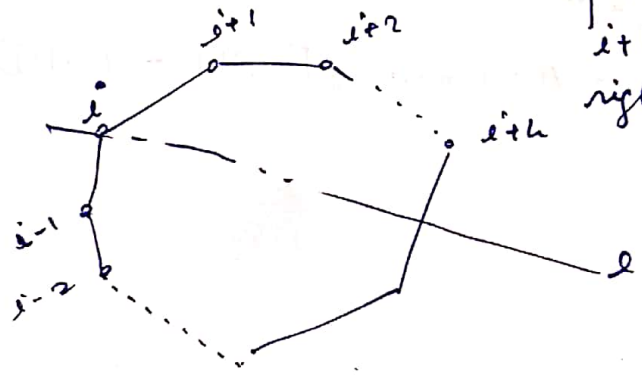


first connect rightmost of $CH(P_L)$ with leftmost of $CH(P_R)$
check if all points of P lie on one side!, if not.
then proceed clockwise on $CH(P_R)$ and anticlockwise on $CH(P_L)$ until you find upper tangent.

but still worst case $O(n^2)$

Very important observation:

No vertex lies to the left of l, if and only if the vertex $i+1$ lies to the right of 'l'.



Time complexity: (conquer, divide & combine)

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

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

Steps:

1) Find the point whose x-coordinate is the median of the x-coordinates of all points of P.
2) Partition P into (P_L, P_R) s.t. x of each point of P_L is less than x of each point of P_R
3) Recursively compute $CH(P_L)$ & $CH(P_R)$
4) Compute the upper and lower tangent of $CH(P_L) \cup CH(P_R)$
5) Report the vertices $i_{min}, i_{min}+1, \ldots, i_{max}, j_{max}, j_{max+1}, \ldots, j_{min}$ in sequence, where

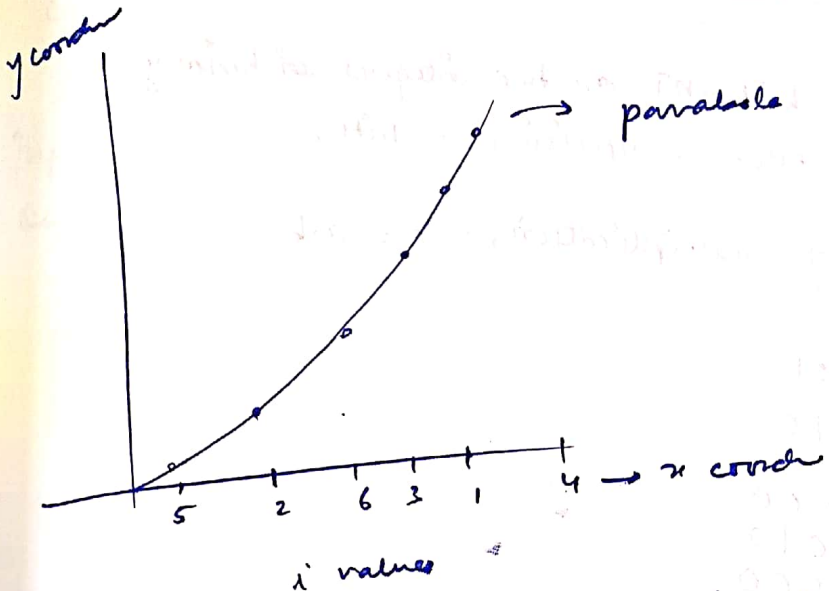
upper tangent passes through $i_{max} \in CH(P_L)$ and $j_{max} \in CH($
and lower tangent passes through $i_{min} \in CH(P_L)$
and $j_{min} \in CH(P_R)$

Sorting of n numbers cannot be done in less than $O(n \log n)$ time [in the worst case]

Convex hull of n points cannot be computed in $O(n \log n)$ in worst case:

Why?

Let $A[1 \ldots n]$ be an array of n numbers.
Define $P[1 \ldots n]$ be a set of n 2D points in which
$x(P[i]) = A[i]$ and $y[P[i]] = (A[i])^2$.



clearly this parabola is a convex hull for the points and contains all the points. Since the ~~vectors~~ vertices of a convex hull are reported back in a given order say clockwise, we will get back, the elements of array A in ascending order. Hence, if convex hull algorithm can run in worst case less than $O(n \log n)$, it would imply that we can sort any n element array in less than worst case $O(n \log n)$, which is not possible.

Tutorial - T.2 :

Input - P = a set of n 2D points.

Output - Q = a polygon whose vertex set is P in an order s.t.

1) upper vertex chain ... (see from moodle)

n- bit integer multiplication

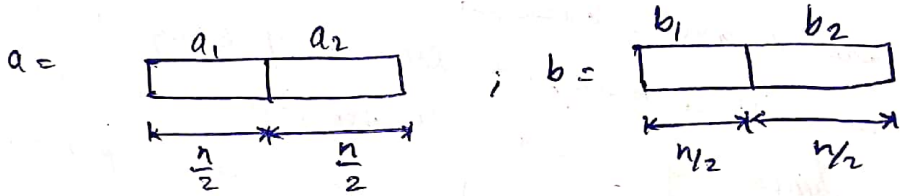
Input: a[1...n] and b[1...n] are two integers in binary number system each comprising n-bits.

Output: The product of multiplication, $c = a \cdot b$

Ex:

```
    a =   1101
    b =   1010
  ---------------
        10000
       11010
      000000
     1101000
  ---------------
    10000010
```

$T(n) = O(n^2)$



Assume, n is a power of 2.

$$a = 2^{n/2} a_1 + a_2$$
$$b = 2^{n/2} b_1 + b_2$$

$$ab = (2^{n/2} a_1 + a_2)(2^{n/2} b_1 + b_2)$$
$$= 2^n a_1 b_1 + 2^{n/2}(a_1 b_2 + a_2 b_1) + a_2 b_2$$

Let T(n) be the time complexity for multiplication of two n-bit numbers.

Then $T(n) = 4T(\frac{n}{2}) + O(n)$

$\Rightarrow T(n) = O(n^2)$ — we need to reduce this

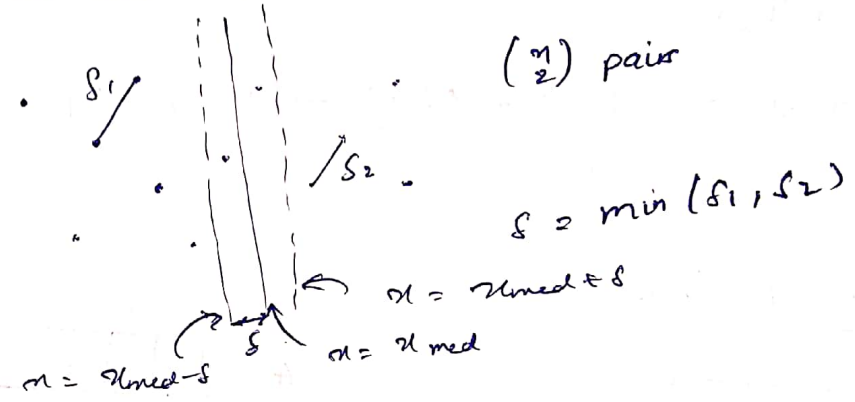
Karatsuba:

$$a_1 b_2 + a_2 b_1 = (a_1 + a_2)(b_1 + b_2) - a_1 b_1 - a_2 b_2$$

$$T(n) = 3T(\tfrac{n}{2}) + O(n)$$
$$T(n) = O(n^{\log_2(3)}) \approx O(n^{1.59})$$

Closest pair in a 2-D point set :

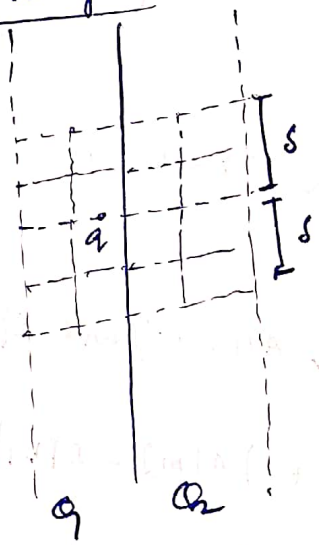
Input - P = a list of n - 2D points
Output - The closest pair of points in P

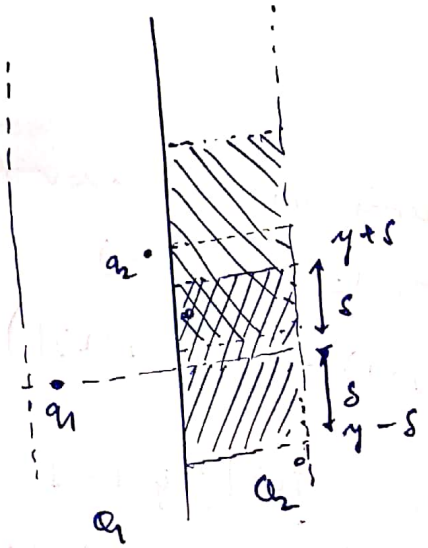


$T(n) = 2T(\frac{n}{2}) + t_{combine}$

{for the combine stage}

Combine stage :





Start from bottom-most point in Q_2 and mark points with y-coordinate $> y - \delta$ but $< y + \delta$
(y is y-coordinate of q_1)

for next point q_2 start from bottom-most point in prev. rectangle.

∴ Time taken to process the points of Q_2 corresponding to all points of $Q_1 = O(n)$

∴ Total time complexity:

$= O(n \log n)$ for sorting (wrt x & y)

$+ T(n)$

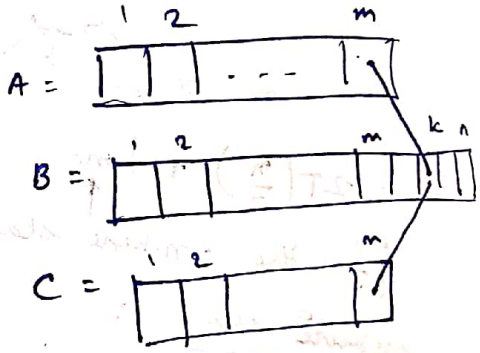
where $T(n) = 2T(\frac{n}{2}) + O(n) \Rightarrow T(n) = O(n \log n)$

Tutorial - 1 - soln':

A = | 2 | 7 | 4 |

B = | 3 | 1 | 9 | 5 |

C = ?



Observations:

(i) $len(C) = len(A)$

(ii) $C[m] = B[k]$ where $m \le k \le n$

Define:

$f(m,n)$ is the required sum over $A[1 \ldots m]$ and $B[1 \ldots n]$

So, $f(m,n) = \min_{m \le k \le n} \left(f(m-1, k-1) + |A[m] - B[k]| \right)$

↑ optimal substructure

Overlapping subproblems:

Let $f(i,j)$ denote the minimum sum over $A[1 \ldots i]$ and $B[1 \ldots j]$, $1 \le i \le m$, $1 \le j \le n$, $i \le j$

$\Rightarrow f(i,j) = \min_{i \le k \le j} \left(f(i-1, j-1) + |A[i] - B[k]| \right)$ if $i \ge 2$

$= \min_{i \le k \le j} \{ |A[1] - B[k]| \}$ if $i = 1$

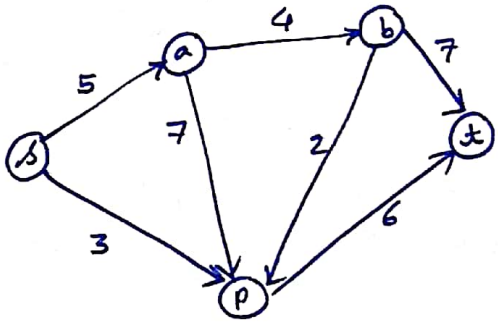
$= \min \{ f(1, j-1), |A[1] - B[j]| \}$

	3	1	9	5
2	1	1	1	1
7	X	7	3	3
4	X	X	12	4

$O(mn)\ O(n-m)$

Flow network :

A directed graph $G(V, E)$ with two distinct vertices called the source (s) and the sink (t) and each edge (u, v) having a capacity $c(u, v) \geq 0$. if $uv \notin E(G)$ then $c(u, v) = 0$



Flow

f is a function defined on $G(V, E)$ with the following necessary and sufficient properties.

(i) capacity constraint : $0 \leq f(u, v) \leq c(u, v)$

$\forall\ (u, v) \in V^2$

(ii) Flow conservation :

$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u) \quad \forall\ u \in V(G) \setminus \{s, t\}$$

incoming (to u)

outgoing (from u)

value of a flow f :

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

outgoing from source

incoming to source

(NOTE: back edges to s are ! possible!)

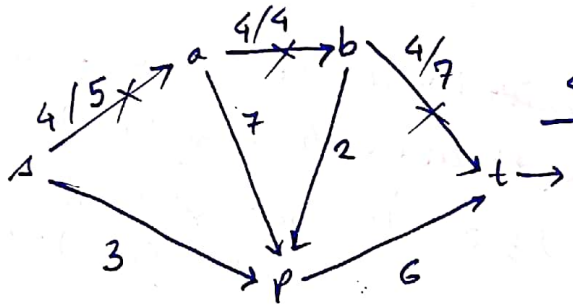
Max flow problem

Augmenting path

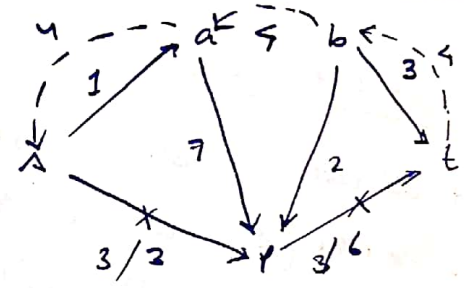
A path from s to t along which a flow is possible

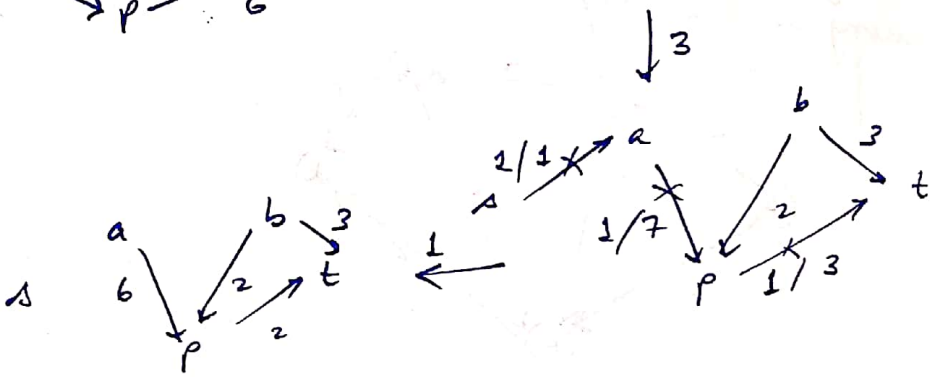
critical edge = edge with minimum capacity in the augmenting path.

Max flow problem



Residual flow Network

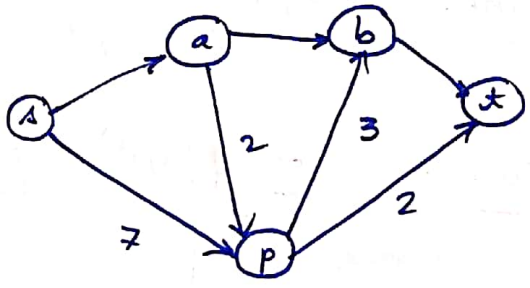




∴ $|f_{max}| = 4 + 3 + 1 = 8$

Residual capacity of an edge .. (denoted by dotted line)

Input. G :

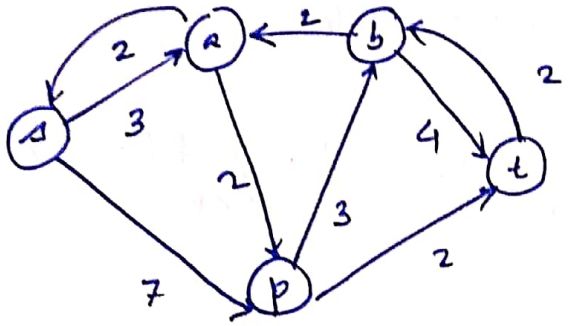


augmenting path : $s \to a \to b \to t$

critical edge : (a, b)

$|f| = 2$

G_f :



G_f = residual network :

$= (V, E_f)$

$(u, v) \in E_f$ if $(u, v) \in E$ or $(v, u) \in E$

note : $|E_f| \leq 2|E|$

Theorem : Let f be a flow in G, and f' be a flow in G_f.
Then, $f \uparrow f'$ is a flow in G and its value is

$$|f \uparrow f'| = |f| + |f'|$$

Let $(u, v) \in V^2$

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{o.w} \end{cases}$$

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

$f'(u,v) \leq c_f(u,v) \quad \forall (u,v) \in V^2$

Capacity constraint : let $(u, v) \in E$

$(f \uparrow f')(u,v) = f(u,v) + f'(u,v)$

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

$= c(u,v) - \{(c_f(u,v)) - f'(u,v)\}$

$\leq c(u,v)$ $\qquad f'(u,v) \leq c_f(u,v)$

flow concavalion : $u \in V \setminus \{s,t\}$

$$\sum_{v \in V} (f \uparrow f')(u,v) = \sum_{v \in V} f(u,v) + \sum_{v \in V} f(u',v)$$

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

$$= \sum_{v \in V} (f \uparrow f')(v,u)$$

Value of $(f \uparrow f')$:

$$|f \uparrow f'| = \sum_{v \in V} (f \overset{\uparrow}{\bullet} f')(s,v) - \sum_{v \in V} (f \overset{\uparrow}{\bullet} f')(v,s) \quad \text{by def.}$$

let $V_1 = \{v : (s,v) \in E\}$ and $V_2 = \{v : (v,s) \in E\}$

$(u,v) \in E \Rightarrow (v,u) \notin E$; hence, $V_1 \cap V_2 = \phi$.

$$\Rightarrow |f \uparrow f'| = \sum_{v \in V_1} (f \uparrow f')(s,v) - \sum_{v \in V_2} (f \uparrow f')(v,s)$$

$$= \sum_{V_1} f(s,v) + \sum_{V_1} f'(s,v) - \sum_{V_2} f(v,s) - \sum_{V_2} f'(v,$$

now we can replace V_1 & V_2 by V in here alsone summation,

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

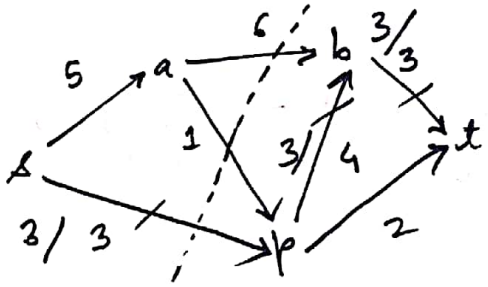
Cut in a flow network :

s = source ; t = sink

$S = \{s\} \cup$ { some vertices of V , except t }

$T = V \setminus S$

(S, T) is a cut in $G(V, E)$



→ flow occurs

$S = \{s, a\}$

$T = \{b, p, t\}$

$f(S,T) = 3+0+0 = 3$ flow of cut

$c(S,T) = 3+1+6 = 10$ capacity of cut

Lemma :

for any cut (S,T) and for any flow, 'f',

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

also :

NOTE: $f(S,T) \leq c(S,T)$ (flow across a cut cannot be greater than capacity of the cut)

$$\Rightarrow |f|_{max} = \min_{(S,T)} \{c(S,T)\}.$$

Max flow, min cut theorem :

The following statements are equivalent :

1) f_{max} is maximum flow in G

2) $G_{f_{max}}$ (residual network) has no augmenting path.

3) $|f_{max}| = \min_{(S,T)} \{c(S,T)\}$

Proof of lemma :

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

$$= \sum_{u \in S} \sum_{v \in V \setminus S} f(u,v) - \sum_{u \in S} \sum_{v \in V \setminus S} f(v,u)$$

$$= \sum_{u \in S} \sum_{v \in V} f(u,v) - \sum_{u \in S} \sum_{v \in S} f(u,v) - \sum_{u \in S} \sum_{v \in V} f(v,u) + \sum_{u \in S} \sum_{v \in S} f(v,u)$$

$$= \underbrace{\sum_{u \in S} \sum_{v \in V} \left(f(u,v) - f(v,u)\right)}_{|f|} + \left\{ - \sum_{u \in S} \sum_{v \in S} f(u,v) + \sum_{u \in S} \sum_{v \in S} f(v,u) \right\}$$

$= 0$, as it is the flow between any two pairs of vertices in S.

Ford & Fulkerson Algorithm :

Based on augmenting paths

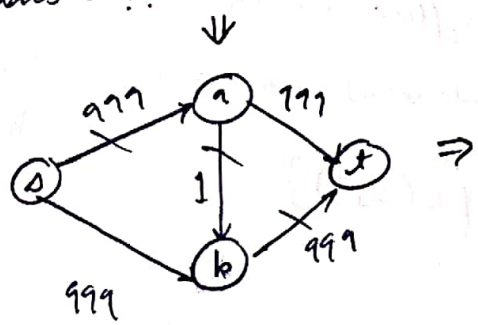
$G = (V, E)$

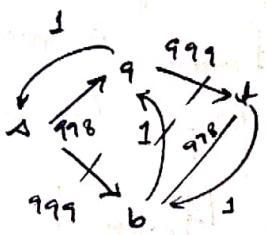
$G_f = (V, E_f)$

$|E_f| \leq 2|E|$

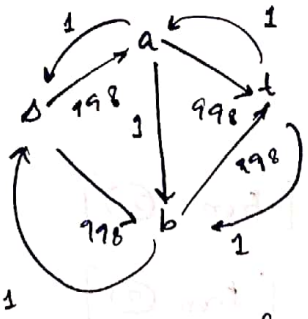
T (find an augmenting path) $= O(E)$

aug. paths = ??





=>



⇒ and so on

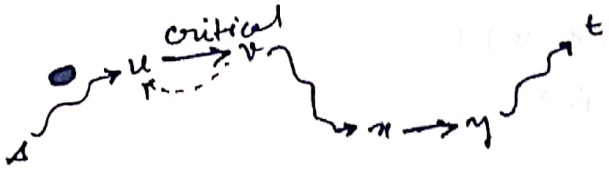
$\therefore$ $2 \times f_{max}$ augmenting paths.

$\therefore$ # aug. paths : $O(|f_{max}|)$

Contribution of Edmonds & Karp : (in easy words, use BFS to find augmenting paths)

Use shortest augmenting path.

- Distance of 'd' from 's' never decrease over successive flow augmentations.



distance of x from y never decreases !!

Theorem - Over successive flow augmentations, the shortest path of a vertex v from s, never decreases.

Proof : By contradiction :
Let the theorem be true upto a flow f and not true when the flow is augmented to $f \uparrow \Delta f = f'$.
Let v be a vertex with minimum shortest-path distance (SD) for which it is not true.

Let $\delta_f(u, v)$ denote the SD from u to V in G_f.

Then $\delta_{f'}(s, v) < \delta_f(s, v)$ — ①

Let $p : s \rightsquigarrow u \rightarrow v$ be the S.P. in G_f

Then $\delta_{f'}(s, u) \geq \delta_f(s, u)$ —— ②

and $\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$ — ③

Let $(u,v) \in E_f$

Then $\delta_f(s,v) \leq \delta_f(s,u) + 1$

$\leq \delta_{f'}(s,u) + 1$ [from ②]

$\leq \delta_{f'}(s,v)$ [from ③]

which is a contradiction [w.r.t ①]

$(u,v) \notin E_f \Rightarrow (v,u) \in E_f$ & $(u,v) \in E_{f'}$, $\Delta f(v,u) > 0$

(the above eqn implies (v,u) is an edge in the augmenting path ⇒④) through which Δf has been pushed.

$\Rightarrow \delta_f(s,u) = \delta_f(s,v) + 1$ ——— ④

③ implies:

$\delta_{f'}(s,v) = \delta_{f'}(s,u) + 1$

$\geq \delta_f(s,u) + 1$ [from (2)]

$= \delta_f(s,v) + 2$

Hence it contradicts ①:

Theorem: Any edge of a flow network can become critical at most $O(V)$ times, where V = vertex-set.

So # of augmenting paths $= |E| \cdot O(V) = O(VE)$.

Proof: Let (u,v) be an edge in $G(V,E)$. Let f be the flow upto which (u,v) is not critical. and $f \uparrow \Delta f = f'$ be the augmented flow when (u,v) becomes critical.

$\Rightarrow (u,v) \in E_{f'}$ & $(v,u) \in E_f$

$\delta_f(s,v) = \delta_f(s,u) + 1$

and $\delta_{f'}(s,u) \leq \delta_{f'}(s,v) + 1$

As. $(u,v) \notin E_{f'}$, (u,v) can reappear as an edge in some subsequent flow augmentation only if (v,u) is an edge of the corresponding augmenting path.

$\Rightarrow \delta_{f'}(s,v) = \delta_{f'}(s,u) - 1$

$\delta_{f'}(s,u) = \delta_{f'}(s,v) + 1$

$\geq \delta_f(s,v) + 1$ [By previous thm.]

$= \delta_f(s,u) + 2$ [from (1)]

$\Rightarrow$ distance of (u,v) from s increases by at least 2 from its one critical state to the next

$\Rightarrow$ (u,v) can become critical for at most $|V|/2$ times.

Proved:

Time complexity of Edmonds-Karp Algorithm:

$O(E)$ to find each aug. path (s.p. from s to t)

$\times O(VE) = O(VE^2)$

Bipartite graph matching:

$G = (V, E)$

$V = X \cup Y$

$X \cap Y = \phi$

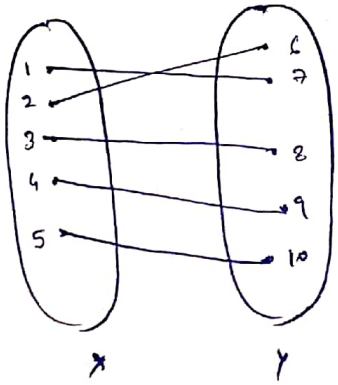
$\forall (u,v) \in E$

$u \in X$, $v \in Y$

$M = \{(x,y) : x \in X, y \in Y\}$

$(x,y) \in M, (u,v) \in M$

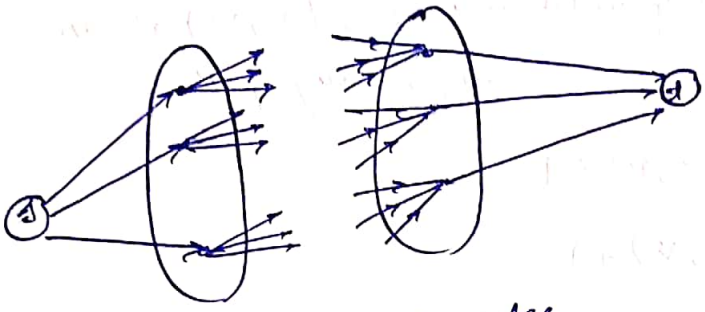
$\Rightarrow \{x,y\} \cap \{u,v\} = \phi$



if $(1,7) \in M$

then $(2,7), (3,7) \in M$

Q: find M s.t. $|M|$ is maximum



cap = 1 for each edge

Theorem: f is a max flow in G'
$\Leftrightarrow$ max matching in G has size $|f|$

Integrality Theorem:

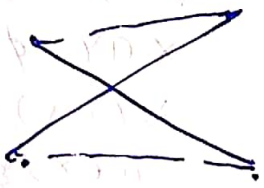
If all edges have int capacities in a flow network then max flow is an int.

Alternating path:

$s \to x \xrightarrow{f} y \xrightarrow{r} x \xrightarrow{f} y \to t$

or $s \to x \xrightarrow{f} y \xrightarrow{r} x \xrightarrow{f} y \xrightarrow{r} x \xrightarrow{f} y \to t$

Hall's Theorem: (Perfect matching $|X| = |Y| = |M|$)



NP complete Problems :

- Optimisation problems :

Examples : Minimum Spanning Tree
Min cut
Max Flow
Shortest Path
Convex Hull

- Decision problems : (whether there exists a soln or not..)

Circuit Satisfiability Problem (SAT)
Formula satisfiability Problem (ϕ-Sat)
3-CNF Satisfiability Problem (3-SAT)
3 - colorability of graph.

Deterministic Algorithm : (steps & results are predetermined)
e.g. quicksort, mergesort

Non deterministic Algorithm :
(Not for implementation, only for understanding the hardness of the problem)
extra power : can choose the right path at every step.

Procedures :

(i) choice (S) → returns an arbitrary element of S
(ii) success → terminates a program on successful completion
(iii) failure → terminates a program on unsuccessful completion

1. i ← choice ({1, 2, ... n})
2. if A[i] = x
   SUCCESS
3. FAILURE

Time = O(1)

Nondeterministic sorting (A[1...n] of +ve nos.)

```
1. for i ← 1 to n
2.     B[i] ← 0
3. for i ← 1 to n                          } permutation
4.     j ← CHOICE({1, 2, ... n})             (non deterministic)
5.     if B[j] ≠ 0
6.         FAILURE
7.     B[j] ← A[i]
8. for i ← 1 to n-1                        } Verification
9.     if B[i] > B[i+1]                      (deterministic)
10.        FAILURE
11. print B
12. SUCCESS
```

Time = $O(n)$

Problem class P :

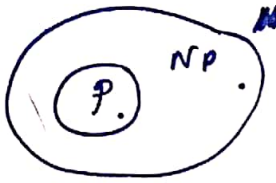
Comprises all (decision) problems that can be solved by deterministic algorithms in polynomial time. (w.r.t input size)

Problem class NP :

comprises all (decision) problems that can be solved by nondeterministic algorithms in polynomial time

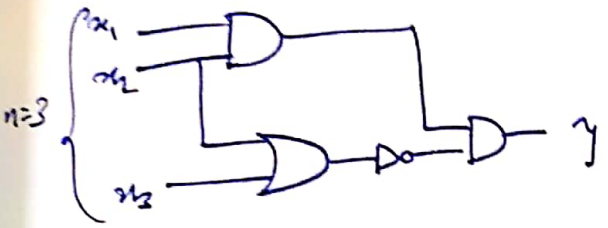
$P \subseteq NP$ { either prove $P = NP$ or $P \subset NP$ }

?



SAT $\in$ NP

P.S: Given a Boolean circuit made of AND, OR, NOT gates, properly connected by wires, we have to determine whether there exists an input certificate for which the output is TRUE.



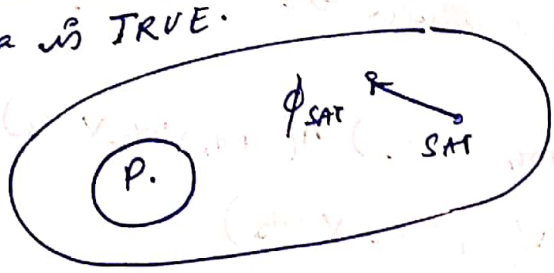
Nobody till date has found a deterministic polynomial time algorithm.

Cook - Levin Theorem

P = NP if and only if SAT $\in$ P

$\phi = ((x_1 \vee x_2) \wedge (\neg x_1 \wedge x_3)) \vee ((x_1 \vee \neg x_2))$

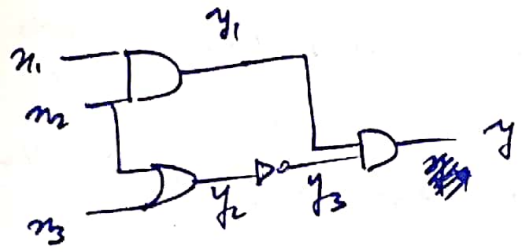
ϕ-SAT: Given a problem (Boolean formula with n Boolean variables and AND ($\wedge$), OR ($\vee$), NOT ($\neg$), and $\leftrightarrow$ (iff) operators, determine whether there exists an input certificate for which the formula is TRUE.



Problem class NP-complete: A problem X belongs to this class if:

1) X $\in$ NP
2) SAT $\leq_p$ X (SAT reduced in polynomial time to problem X)

SAT $\in$ NP - complete
ϕ-SAT $\in$ NP - complete [Because ϕ-SAT $\in$ NP and SAT $\leq_p$ ϕ-CAT]



(Label the o/p of every gate)

$y = y_1 \wedge y_3 = y_1 \wedge \neg y_2 = (x_1 \wedge x_2) \wedge (\neg (x_2 \vee x_3))$

$\phi = y \wedge (y_1 \leftrightarrow x_1 \wedge x_2)$
$\wedge (y_2 \leftrightarrow (x_2 \vee x_3))$
$\wedge (y_3 \leftrightarrow \neg y_2)$
$\wedge (y \leftrightarrow y_1 \wedge y_3)$

NP Compete Problems:

3-SAT : Given a Boolean formula in conjunctive normal form (CNF) in which each clause consists of exactly 3 literals, decide whether there exists an i/p certificate for which the Boolean formula is satisfiable.

Ex: $\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$
$\wedge (x_1 \vee x_2 \vee x_3)$

3 variables and 4 clauses.

3-Sat is NP complete:

Proof: 1) $SAT \leq_p 3\text{-}SAT$

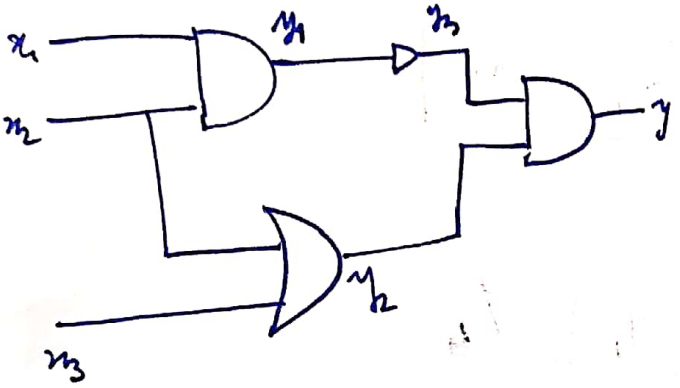
2) $3\text{-}SAT \in NP$

clauses = k = poly(n) } — incl. in problem statement
i/p vars = n

Verification can be done in poly(n) time
$\therefore$ $3\text{-}SAT \in NP$ — (2) Proved

To prove (1), we have to propose a deterministic poly(n) time algorithm that can reduce any instance of SAT problem to some instance of 3-SAT problem.

SAT instance :



i/p vars: x_1, x_2, x_3

o/p: y

(1) Intermediate vars: y_1, y_2, y_3

gates : poly(n)

n = # i/p vars

(2) equivalent Boolean formula :

$$\phi = y \wedge (x_1 \wedge x_2 \leftrightarrow y_1) \wedge (x_2 \vee x_3 \leftrightarrow y_2) \wedge (\neg y_1 \leftrightarrow y_3) \wedge (y_2 \wedge y_3 \leftrightarrow y)$$

(ϕ_2 : $x_1 \wedge x_2 \leftrightarrow y_1$, ϕ_3 : $x_2 \vee x_3 \leftrightarrow y_2$, ϕ_4 : $\neg y_1 \leftrightarrow y_3$, ϕ_5 : $y_2 \wedge y_3 \leftrightarrow y$)

$\therefore \phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_{poly(n)}$

(3) $\phi_2 : x_1 \wedge x_2 \leftrightarrow y_1$

x_1	x_2	y_1	ϕ_2
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$\neg\phi_2 : (\neg x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_2 \wedge y_1) \vee (x_1 \wedge \neg x_2 \wedge y_1) \vee (x_1 \wedge x_2 \wedge \neg y_1)$

$\phi_2 = (x_1 \vee x_2 \vee \neg y_1) \wedge (x_1 \vee \neg x_2 \vee \neg y_1) \wedge (\neg x_1 \vee x_2 \vee y_1)$
$\wedge (\neg x_1 \vee \neg x_2 \vee y_1)$

$\phi_4 = (\neg y_1 \leftrightarrow y_1)$

Truth table : 3 columns, 4 rows.

$SAT \leq_p 3\text{-}SAT$

Contd...

y_1	y_3	ϕ_4
0	0	0
0	1	1
1	0	1
1	1	0

clearly,

$\neg \phi_3 = (\neg y_1 \wedge \neg y_3) \vee (y_1 \wedge y_3)$

Introduce dummy := z

$\neg \phi_3 = (\neg y_1 \wedge \neg y_3 \wedge z) \vee (\neg y_1 \wedge \neg y_3 \wedge \neg z)$
$\vee (y_1 \wedge y_3 \wedge z) \vee (y_1 \wedge y_3 \wedge \neg z)$

$\Rightarrow \phi_3 = (y_1 \vee y_3 \vee \neg z) \wedge (y_1 \vee y_3 \vee z) \wedge (\neg y_1 \vee \neg y_3 \vee \neg z)$
$\wedge (\neg y_1 \vee \neg y_3 \vee z)$

now,

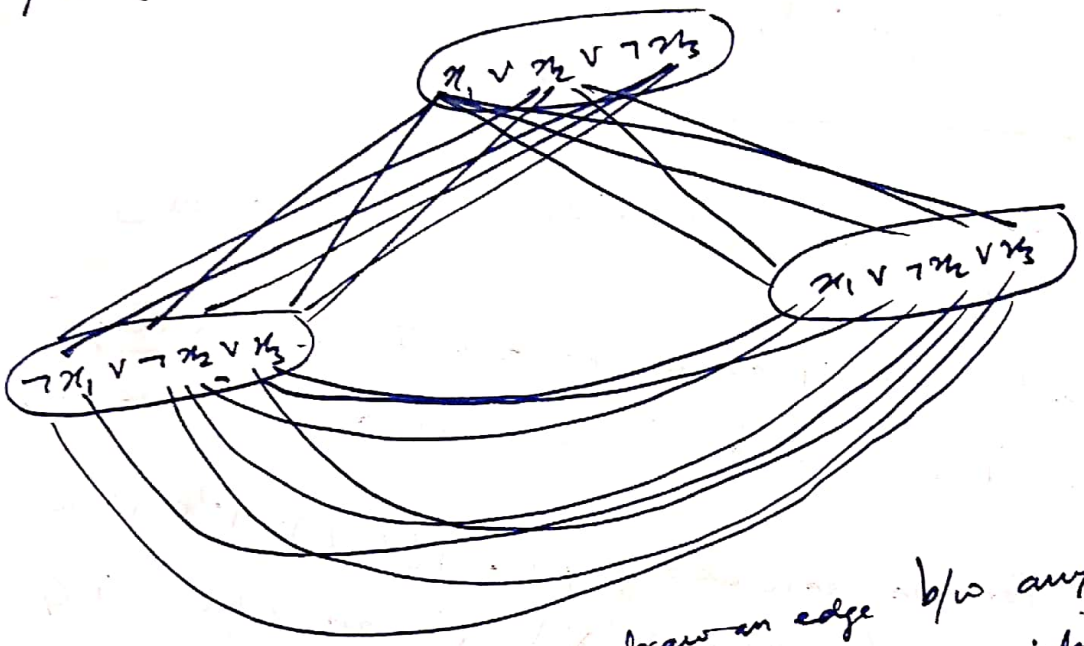
y_1	ϕ_1
0	0
1	1

Introduce 2 dummy variables, z_1 & z_2

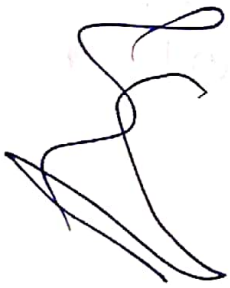
$\phi_1 \Rightarrow (x_1 \wedge z_1 \wedge z_2) \vee (y \wedge \neg z_1 \wedge \neg z_2) \vee (y \wedge \neg z_1 \wedge z_2)$
$\vee (y \wedge \neg z_1 \wedge \neg z_2)$

3 - Sat Instance :

$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$



draw an edge b/w any 2 non conflicting variables !!
so x_1 and $\neg x_1$ conflict !
but x_1 and $\neg x_1$ / $(\neg) x_2$ / $(\neg) x_3$ do not conflict.



∴ for sol'n to exist, there MUST exist a cycle covering all the clauses !!

In general,

Let $\phi = \bigwedge_{i=1}^{k} C_i$

Let $G(V, E) \leftarrow \phi$

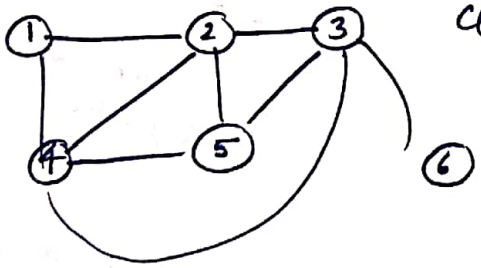
$C_i = \bigvee_{j=1}^{3} y_{ij}$, $i = 1, 2, \ldots, k$

$V = \{ y_{ij} \mid 1 \le i \le k, \ 1 \le j \le 3 \}$

$E = \{ (y_{ij}, y_{uv}) : i \neq u, \ y_{ij} \neq \neg y_{uv} \}$

ϕ is satisfiable if and if and only if G has a clique of size 'k', where k is the number of clauses in ϕ.

Clique: For an undirected graph $G(V, E)$; $V' \subseteq V$ is a clique if $(u, v) \in V' \times V'$ is an edge in G for all $(u, v) \in V' \times V'$.



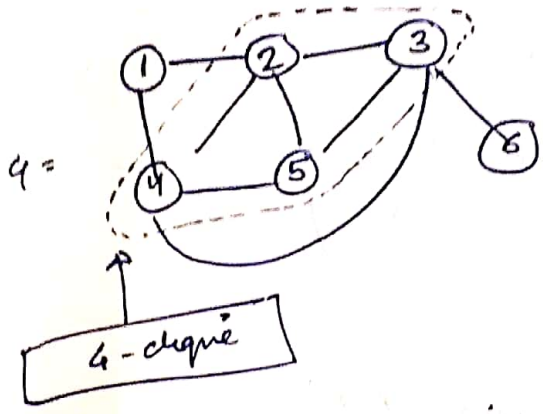
Clique = {2, 3, 4, 5}

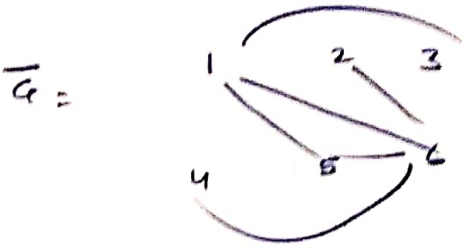
|Clique (V)| = 4

clique (G, k) :

Given a graph G and a number k, determine whether G contains a clique of size 'k'.

Clique is NP-Complete :





K_{max} is max clique in G.

K_{max} max-independent set in $\bar{G}$?

A subset of vertices having no edge

for every edge of $\bar{G}$, atleast one vertex is outside the independent set (I)

$\Rightarrow$ Every edge of $\bar{G}$ is incident on some vertex outside the independent set.

$\Rightarrow$ The complement of the Independent set ($V \setminus I$) covers all the vertices of $\bar{G}$.

$\Rightarrow$ $V \setminus I$ is the vertex cover of $\bar{G}$.

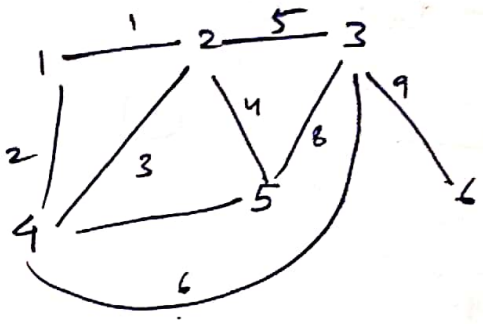
General statement:

I is an independent set in $G(V, E)$ if and only if $V \setminus I$ is a vertex cover in G.

[V' is a vertex cover of G if for every $(u, v) \in E$, $\{u, v\} \cap V' \neq \emptyset$]

for G, $V' = \{2, 3, 4\}$ min. vertex cover

SAT
↓
ϕ - SAT
↓
3-SAT
↓
CLIQUE
↓
INDEPENDENT SET
↓
VERTEX COVER
↓
Set cover.



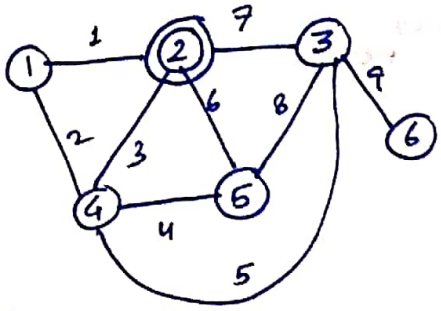
Incidence matrix

(vertex) A	B (edges)
1	1, 2
2	1, 3, 4, 5
3	5, 6, 8, 9
4	2, 3, 6, 7
5	4, 7, 8
6	9

Set cover! : min no. of sets s.t their union gives the entire family.

Set cover problem : (NP-Complete)

Let $\mathcal{U}$ be a universal set. Let $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ be a family with each member $S_i \subseteq \mathcal{U}$ for $i = 1, 2, \ldots, n$ Let k be a given number. Decide whether there exist k members of $\mathcal{F}$ such that their union is $\mathcal{U}$.



Vertex	Incident edges
1	1, 2
2	1, 3, 6, 7
3	5, 7, 8, 9
4	2, 3, 4, 5
5	4, 6, 8
6	9

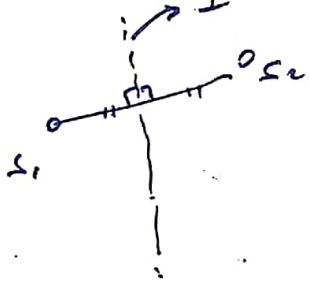
$U = E = \{1, 2, 3, \ldots, 9\}$

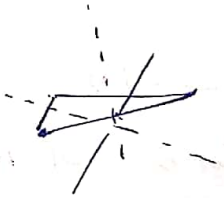
$T = \{\{1,2\}, \{1,3,6,7\}, \{5,7,8,9\}, \{2,3,4,5\}, \{4,6,8\}, \{9\}\}$

These three cover the set $\{1, 2, \ldots, 9\}$

Post office location problem :

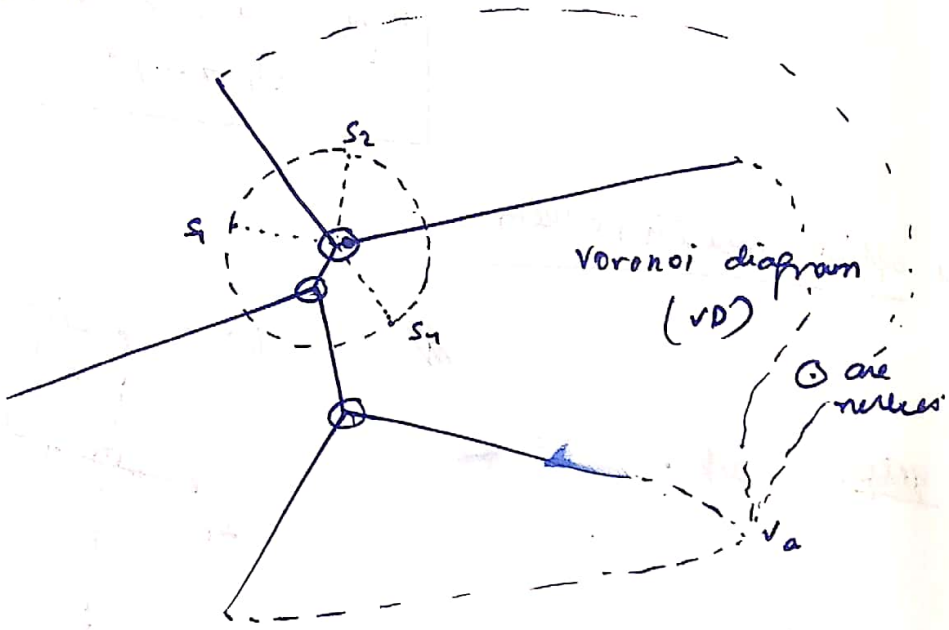
for 2 points ⊥ bisector





$S = \{S_1, S_2, \ldots, S_n\}$ is a set of sites on x-y plane.

we have to subdivide the x-y plane into n regions: $R_1, R_2, \ldots, R_n$ such that for any point $p \in R_i$, S_i is the nearest site.



Assume: 1) all sites have distinct x & y coordinates

2) No four sites are concyclic.

p is a vertex.

$\Leftrightarrow$ p is a circumcentre of three sites and the corresponding circumcircle doesn't contain any site in its interior.

now, how many vertices, edges & faces are present in V.D.?

$\Rightarrow$ # regions $= n$

now add some vertex V_a to the VD and extend all edges (maybe curved) to V_a. we then get a planar graph and can use Euler's formula which states in a planar graph (no. of vertices) $-$ (no. of edges) $+$ (no. of faces/regions) $= 2$

$\therefore n_v + n_1 - n_e + n = 2$ — (1)

now sum of degrees of $n_v + 1$ vertices $\geq 3(n_v + 1)$

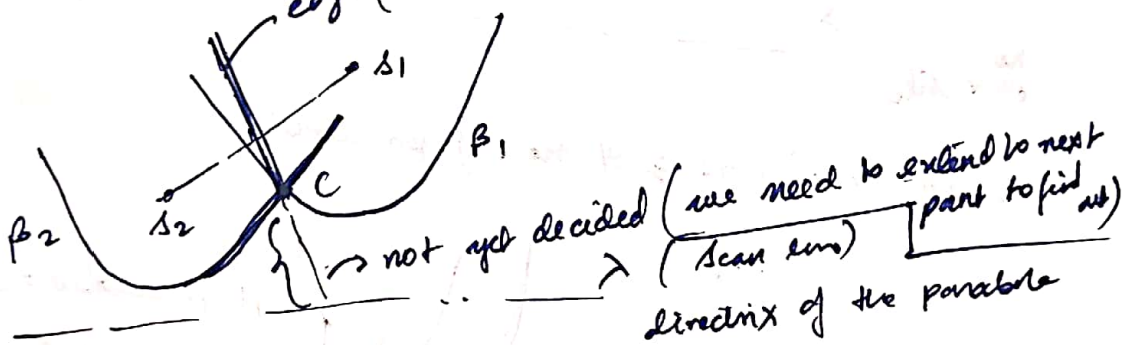
$\therefore 2n_e \geq 3n_v + 3 \quad -(2)$

1 & 2 $\Rightarrow 2(n_v + n - 1) \geq 3n_v + 3$

$\Rightarrow n_v \leq 2n - 5 = O(n)$

similarly $n_e = O(n)$

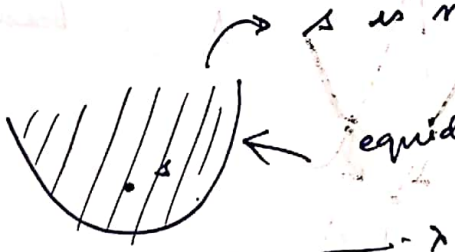
Voronoi diagram



β_1 = parabolic arc with s_1 as focus

β_2 = parabolic arc with s_2 as focus

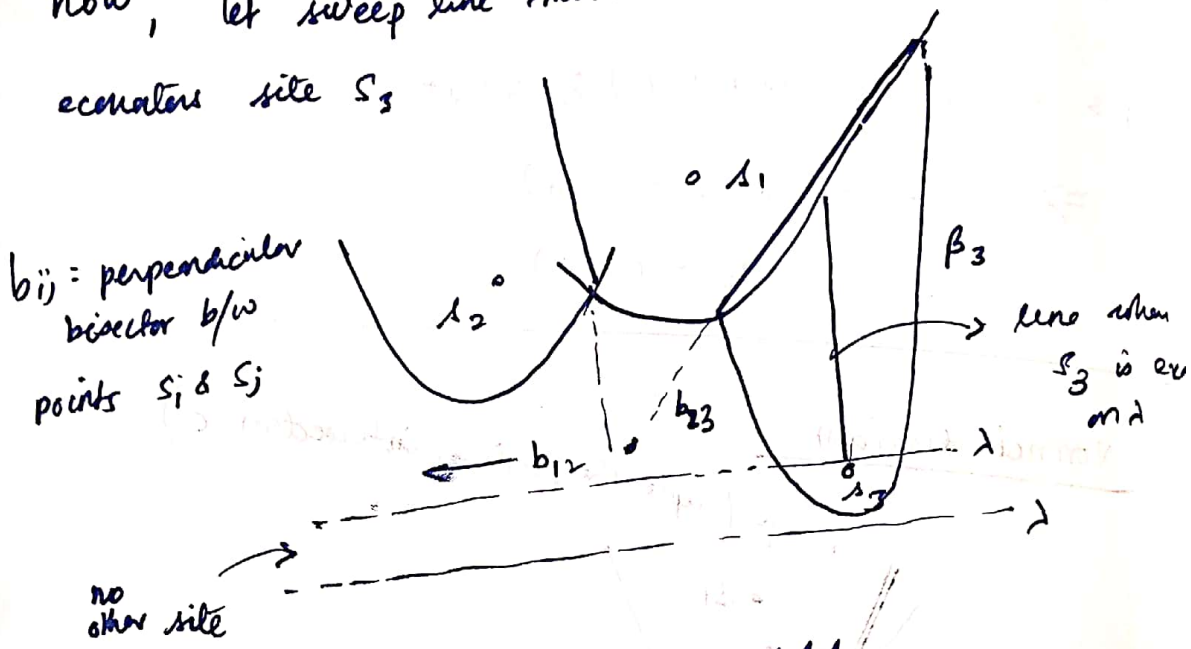
Parabola :



$\therefore$ any point in interior of $(\beta_1 \cup \beta_2)$ is nearer to either s_1 or s_2 than ∂.

$\Rightarrow$ s_1 or s_2 is the nearest among all n sites for p.

now, let sweep line move downwards till it encounters site s_3



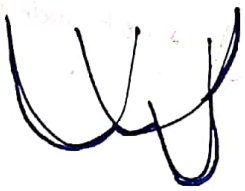
b_{ij} : perpendicular bisector b/w points s_i & s_j

→ line when s_3 is encountered on λ

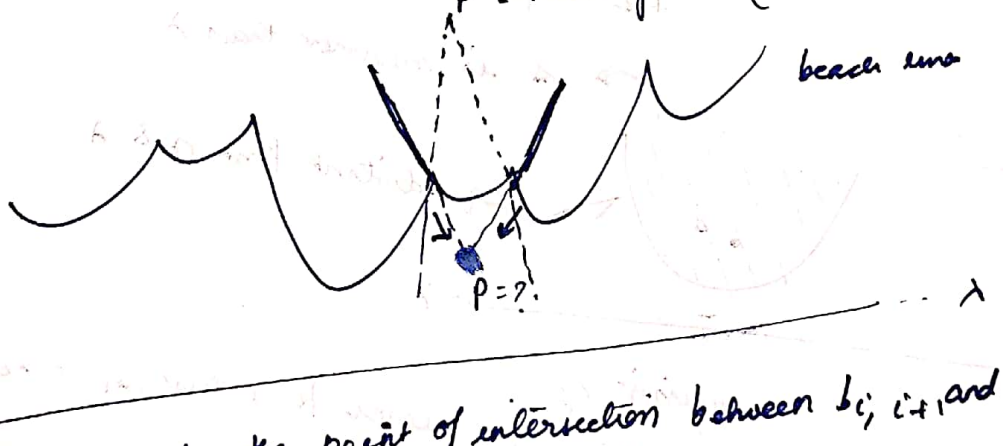
no other site

Note parabolic arcs ≥ # no. of parabolas

e.g.



parabolas = 3
arcs = 4



Let p be the point of intersection between $b_{i,i+1}$ and $b_{i+1,i+2}$ remain consecutive until β_{i+1} disappears from the beach line.

When can we say a new possible edge has been found?

→ when a new site is encountered by λ.

↑
Site Event (all known in the beginning)

circle event : (gradually known as l comes down)

A new possible vertex is found ⇒ a new Triplet of parabolic arcs has appeared in the Beach line (some arc appears or disappears)

lowest point of the circumcircle

a new vertex will only be found between 3 sites α, β, γ, iff the circle event occurs before discovering a new site s: else, that vertex is scrapped

Binary Search Tree