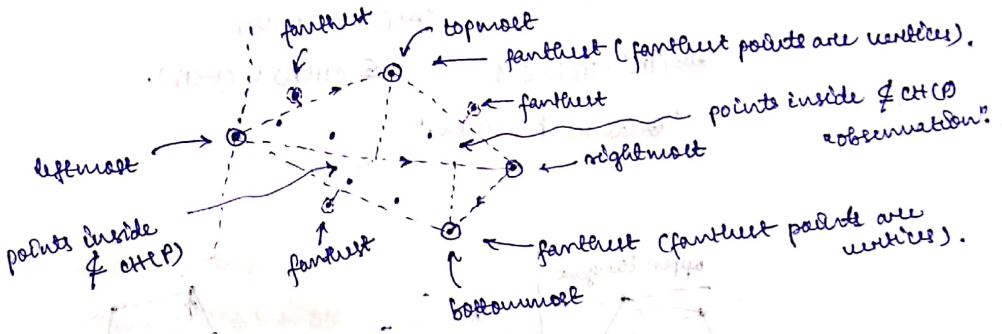
04/07/2019 **ALGORITHMS - II** 0800-1000

Convex Hull:

- Let P be a set of 2-dimensional points.
- The convex hull (CH(P)) of P is the smallest-perimeter polygon that contains all points of P.
- The specific vertices of CH(P) are described in a specific order, say clockwise.



→ "Convex hull" for a given set of points is unique, so rotating the axes won't matter.

"Quick Hull".

Time-complexity:

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

- Best Case: α is a constant;

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

- Worst case: $O(1)$

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

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

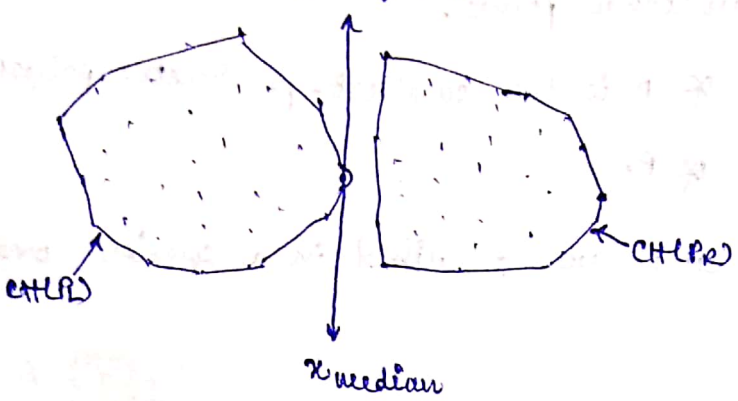
$$\Rightarrow T(n) = O(n^2).$$

#Plot
F1-Score or any Bias-Score wrt Biases, if something like breakdown can be observed.

CH(P) by Divide and Conquer:

CH is a sequence of vertices, in a specific order.

Divide, Conquer, combine.

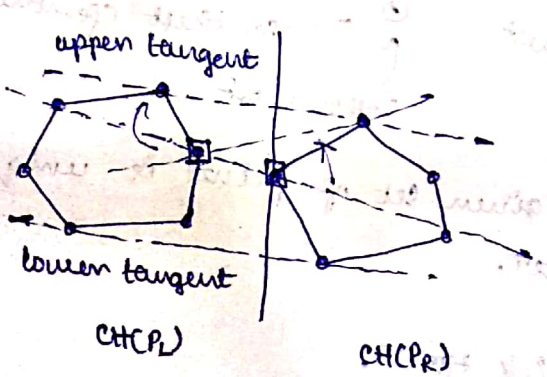


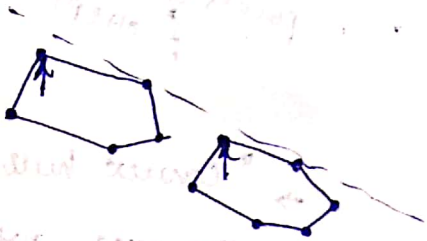
$CH(P) = CH(P_L \cup P_R)$

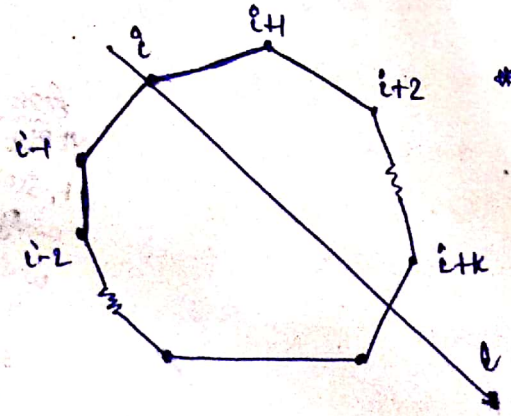
$\subseteq CH(P_L) \cup CH(P_R)$.

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

"Balanced partition."







* Another observation;

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

- Steps:

① Find the point whose x-coordinate is the median of the x-coordinates of all points of P. {can be found using worst case O(n)}.

② Partition P into (P_L, P_R) such that x of each point of P_L is less than x of each point of P_R.

③ Recursively compute CH(P_L) and CH(P_R).

④ Compute the upper tangent and the lower tangent of CH(P_L) ∪ 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 the upper tangent passes through

$i_{max} \in$ CH(P_L) and $j_{max} \in$ CH(P_R)

and the lower tangent passes through

$j_{min} \in$ CH(P_R) and $i_{min} \in$ CH(P_L).

$$T(n) = \underbrace{2T(n/2)}_{\text{conquer.}} + \underbrace{O(n)}_{\substack{\text{divide and} \\ \text{combine.}}}$$

$\Rightarrow T(n) = O(n \log n)$ {Can't be better in worst case}

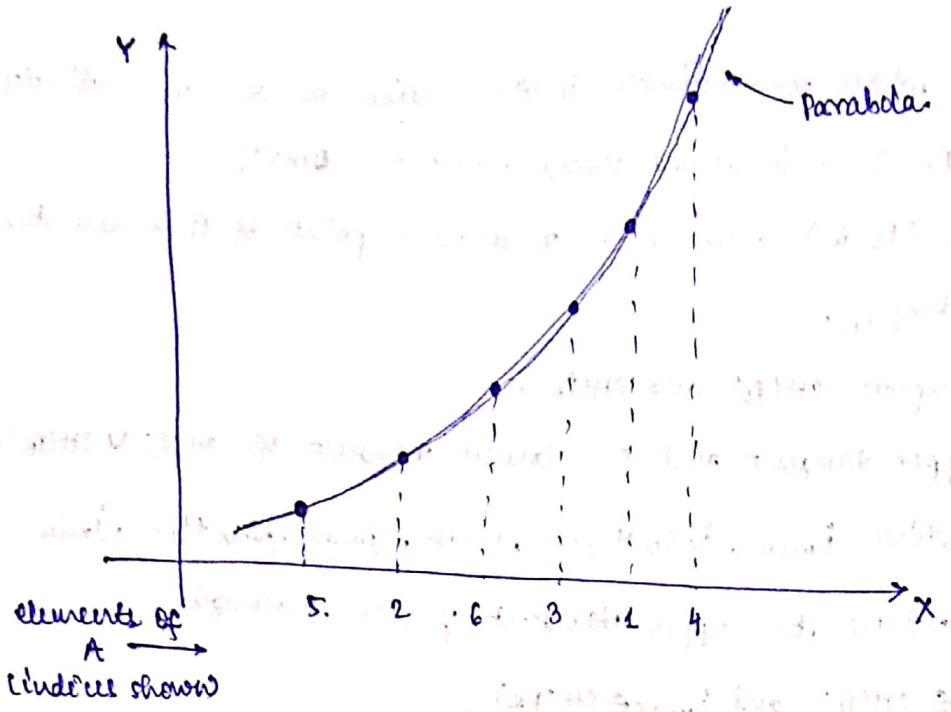
- Sorting of n numbers cannot be done in less than O(nlogn) time [in the worst case].

- convex hull of n points cannot be computed in less than O(nlogn) time.

Why? Let A[1..n] be an array of n numbers.

Define: P[1..n] be a set of n 2D points in which $x(P[i]) = A[i]$ and

$y[P[i]] = (A[i])^2$.



→ So, all points of P will lie on a parabola whose axis is parallel to y-axis.

→ As a parabola is convex from the interior, all points of P will be vertices of CH(P).

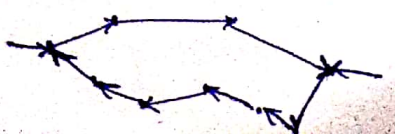
→ A convex-hull algorithm will report these vertices (i.e. all points of P) in clockwise order and hence we can find the sorted sequence of A.

→ If the convex-hull algo runs in less than O(nlogn) time, then A gets sorted in less than O(nlogn) time.

"contradiction".

#Tutorial-2:

- Input - P : a set of n-2D points.
- output - Q : a polygon whose vertex set is P in an order such that:
  1) the upper vertex chain of Q is x-monotone (inc.) from the leftmost vertex to the rightmost vertex of P.
  2) the lower vertex chain of Q is x-monotone (dec.) from the rightmost to the leftmost vertex of P.
  3) Total perimeter of Q is minimum.

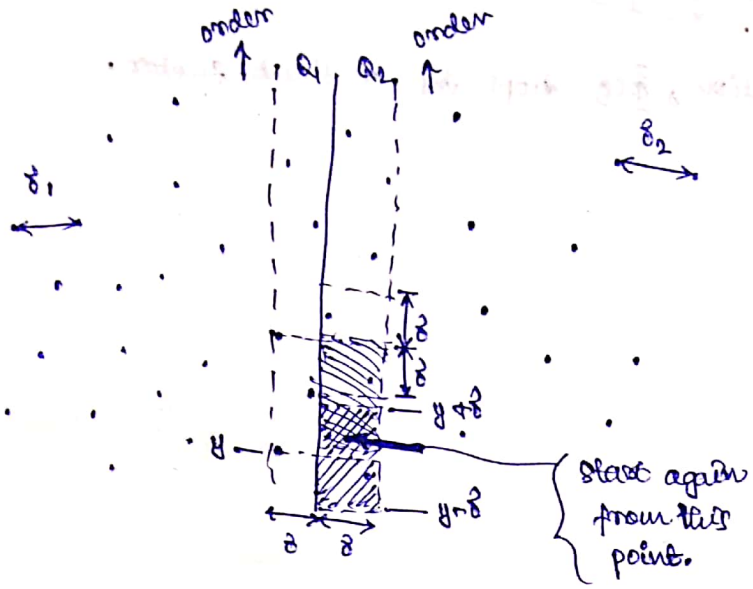


"Dynamic Programming".

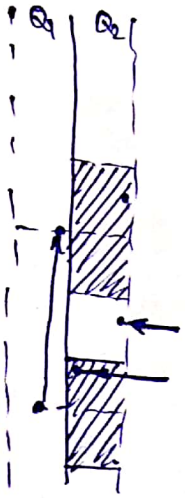
In Polynomial Time.

Closest pair in a 2D Point set;

combine stage;



$\delta = \min(\delta_1, \delta_2)$.



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 (w.r.t. x and w.r.t. y)

$+ T(n)$

where, $T(n) = 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$.

ALGORITHMS-II

31/08/2019

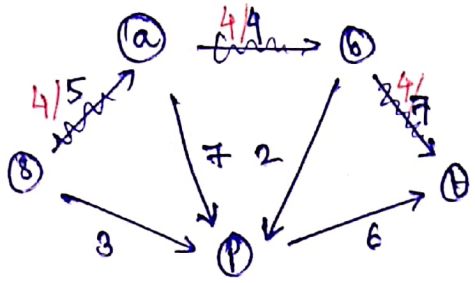
0800-1000.

Ex:

#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) ≥ 0.

  If (u,v) ∉ E, then c(u,v) = 0.



flow:

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

① capacity constraint: $0 \le f(u,v) \le c(u,v) \quad \forall (u,v) \in V^2$.

② flow conservation:

outgoing — incoming

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

#Max flow problem:

- Value of a flow f:

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

#Augmenting Path:

- A path from s to t along which a flow is possible.
- Critical edge = edge with minimum capacity in the augmenting path.

twoDarray :

- space.

at :

"at"

comma :

", "

space :

" "

newline :

"\n"

minMax :

"Min(row) Max(col): \n"

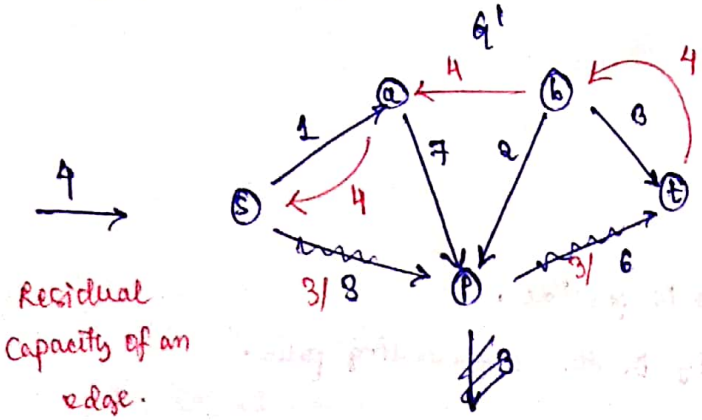
maxMin :

"Max(row) Min(col): \n"

noSad :

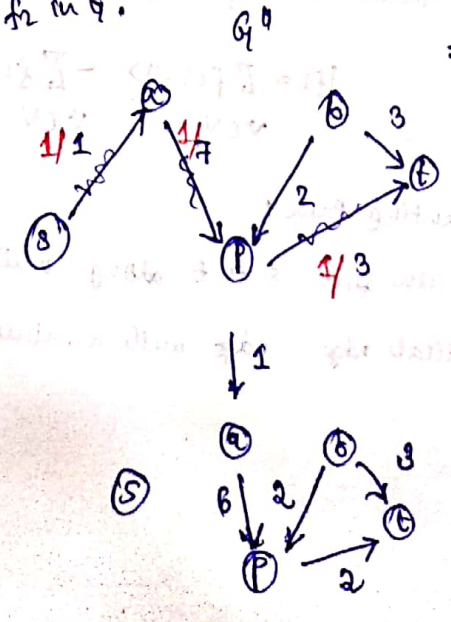
"No saddle point found!"

Residual flow Network :



G1 in G
f2 in G'

⇒ f1 + f2 in G.



sad edith comment

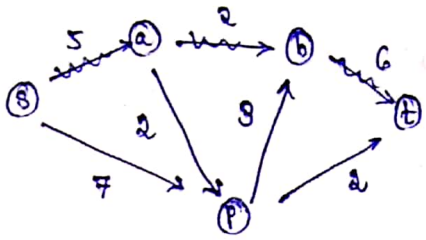
01/08/2019

ALGORITHMS - III

1000 - 1100

#Flow Network's

Input: G =



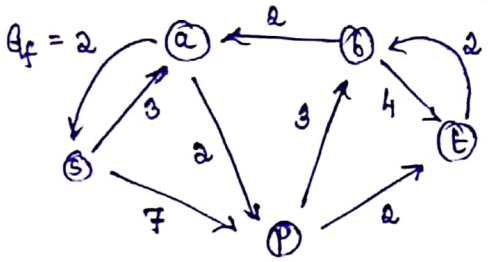
$G = (V, E)$

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 $(v, u) \in E$.

(if it is a back edge in G.)

$|E_f| \leq 2|E|$.

* **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 $(u, v) \in E_f$, V^2.

Residual capacity

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E. \\ f(v, u) & \text{if } (v, u) \in E. \\ 0 & \text{otherwise.} \end{cases} \quad \text{—— (1)}$$

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

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

$\Rightarrow c_f(u, v) - f'(u, v) \geq 0$ —— (3)

Capacity constraint :

- Let $u, v \in V$.

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

$$\leq c(u,v) + f'(u,v).$$

- Let $(u,v) \in E$:

a) $$(f \uparrow f')(u,v) = f(u,v) + f'(u,v) \geq 0$$

$$= c(u,v) - c_f(u,v) + f'(u,v) \quad \text{— from (1).}$$

$$= c(u,v) - (c_f(u,v) - f'(u,v)) \quad \text{— from (2).}$$

$$\leq c(u,v).$$

Flow conservation :

- $u \in V \setminus \{s,t\}$

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

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

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

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

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

→ 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 $V_1 \cap V_2 = \phi$.

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

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

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

$$= |f| + |f'|$$

ALGORITHMS - II

05/08/2019

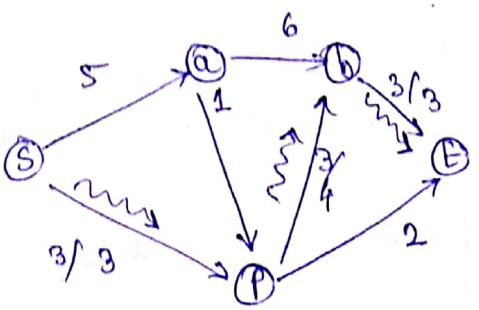
cut in a flow network:

s = source , t = sink

S = {s} ∪ {some vertices of V excepting t}

T = V \ S.

⇒ (S, T) is a cut in G(V, E).



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|.

- fact: f(S, T) ≤ c(S, T).

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

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 3:

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

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

a)

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

$$= \underbrace{\sum_{u \in S} \sum_{v \in V} (f(u,v) - f(v,u))}_{|f|} \underbrace{- \sum_{u \in S} \sum_{v \in S} f(u,v) + \sum_{u \in S} \sum_{v \in S} f(v,u)}_{0}$$

Ford and Fulkerson Algorithm: $\rightarrow O(E|f_{max}|)$.

- Based on Augmenting Paths.

b)

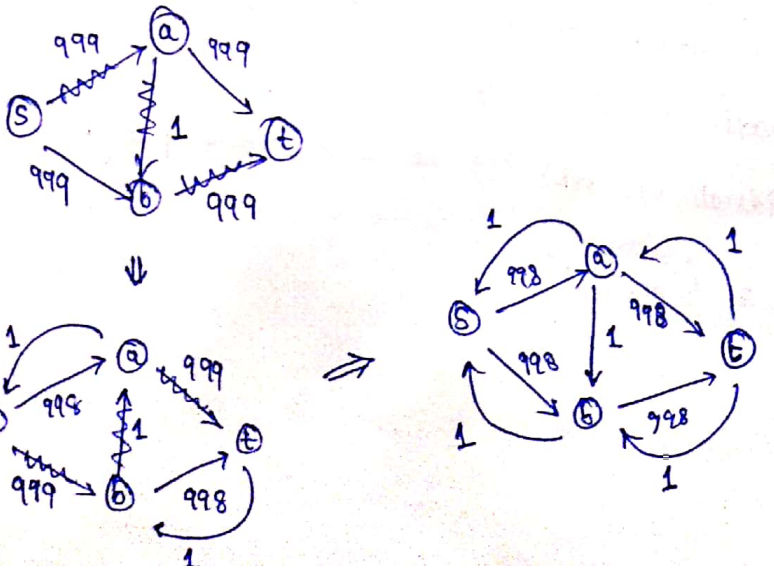
$G = (V, E)$

$G_f = (V, E_f)$

$|E_f| \leq 2|E|$

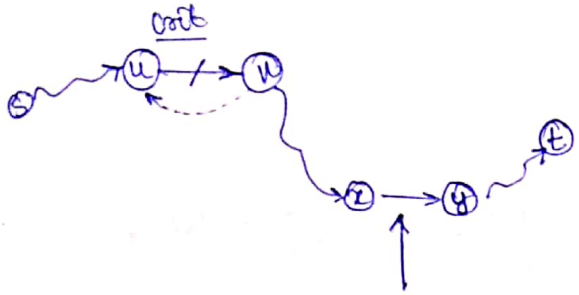
T(find an aug. path) = $O(E)$.

aug. paths = ? $|f_{max}|$.



contribution by Edmonds and Karp's

- Use shortest augmenting path.
  - Distance of t from s never decreases over successive flows augmentations.



07/08/2019

ALGORITHMS - II

0800 - 1000

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

- Proof: By contradiction:

Let the theorem be true upto a flow f and ~~not~~ 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 shortest path in $G_{f'}$.
(u → immediate predecessor.)

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

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

Let $(u, v) \in E_f$

$\Rightarrow$ 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)]

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

$\Rightarrow \delta_f(s, u) \not\leq \delta_f(s, v) + 1$. ——— (4)

(v, u) is an edge in the augmenting path through which Δf has been pushed.

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

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

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

Hence, it contradicts (1).

aug. path

$s \rightsquigarrow v \rightarrow u \rightsquigarrow t$

Δf.

Eg. 4.

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

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

- Proof: Let (u,v) be any edge in $G(V,E)$.

~~Let f be the flow when (u,v) becomes critical.~~

Let f be the flow upto which (u,v) is not critical,
and $f \uparrow Af = f'$ be the augmented flow when (u,v) is critical.

$\Rightarrow (u,v) \notin E_{f'}$ and $(v,u) \in E_{f'}$.

$$\delta_f(s,v) = \delta_f(s,u) + 1. \quad \text{—— (1)}.$$

and $\delta_{f'}(s,u) \leq \delta_{f'}(s,v) + 1$. —— (2).

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

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

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

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

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

S.P.

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

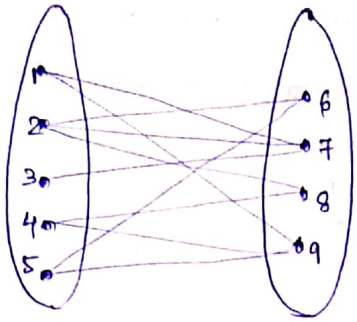
$\Rightarrow (u,v)$ can become critical for at most $|V|/2$ times.

Proved.

Time complexity of Edmonds-Karp Algorithm:

- $O(E)$ to find each aug. path (S.P. from s to t),
  - $O(VE) = O(VE^2)$.

Bipartite Graph Matching:



X Y

$G = (V, E)$

$V = X \cup Y$

$X \cap Y = \phi$

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

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

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

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

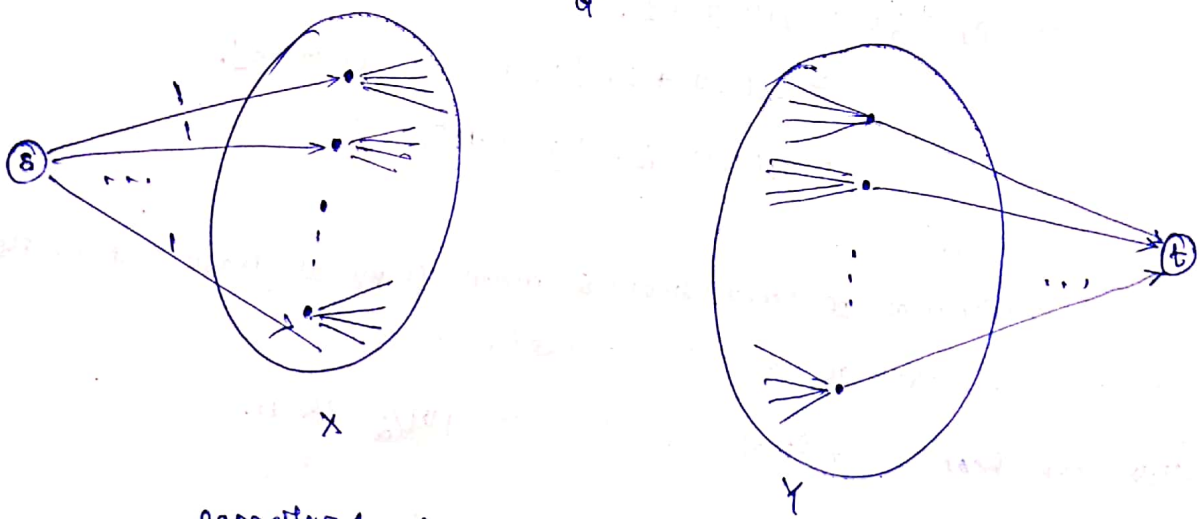
$\Rightarrow \{x,y\} \cap \{u,v\} = \phi$.

$(1,7) \in M$

$\Rightarrow (2,7), (3,7) \notin M$.

* Q: Find M with Maximum $|M|$. {Cardinality}.

Q'



capacity = 1 for each edge

Theorem: f is a max flow in Q'.

$\Leftrightarrow$ Max. matching in Q has size |f|.

Integrality Theorem:

- If all edges have integer capacities in a flow network, then max flow is an integer.

Alternating path: $s \to x \xrightarrow{f} y \xrightarrow{c} x \xrightarrow{f} y \to t$.

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

Hall's Theorem:

Perfect Matching: $|X| = |Y| = |M|$.

ALGORITHMS - II

11/06/2019

0800 - 1000

N-P-complete Problems;

- optimisation Problems;
  - Example - Minimum Spanning Tree.
    Min ~~Cut~~ cut.
    Max flow.
    Shortest Path.
    Convex Hull.
- Decision Problems;
  - Example - Circuit Satisfiability Problem (SAT).
    - Formula satisfiability Problem (ϕ-SAT).
    - 3-CNF Satisfiability Problem (3SAT).
    - 3-Colorability of Graph.

Deterministic Algorithm; Given an I/P, Intermediate states are fixed.

Non-Deterministic Algorithms;

- More Powerful than Deterministic Algorithms.
- Motivation: To understand the difficulty of a Problem.
- Procedures:
  - CHOICE (S) → returns an arbitrary element of S.
  - SUCCESS → terminates a program on successful completion.
  - FAILURE → terminates a program on unsuccessful completion.

Non-Deterministic search;

- Input: A[1..n], key x.

Steps:

1. i ← CHOICE (1, 2, ..., n)
2. if A[i] = x
3. SUCCESS
4. FAILURE

Time = O(1).

Nondeterministic sorting (A[1...n] of positive numbers);

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

→ External Power is given to the CHOICE function.

Problem class $\mathcal{P}$;

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

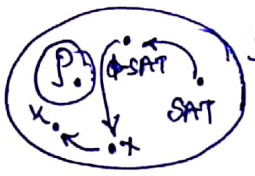
Problem class $\mathcal{NP}$;

- comprises all problems that can be solved by nondeterministic algorithms in polynomial time.

$$\mathcal{P} \overset{?}{\subseteq} \mathcal{NP}$$

$\mathcal{P}$ is a subset of $\mathcal{NP}$.

whether proper subset or not, not proved yet.

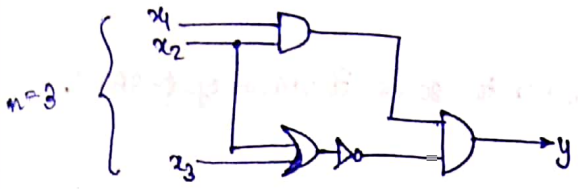


↳ Is it finite or infinite? (countable?)

* How to compare two NP problems??

SAT $\in$ NP ;

- Given a Boolean circuit made of AND ($\wedge$), OR ($\vee$), NOT ($\neg$), XOR ($\leftrightarrow$) gates, properly connected by wires, we have to determine whether there exists an input certificate for which the output is TRUE.



→ If I can verify a problem in Polynomial time, then Yay! it belongs to NP.

Cook-Levin Theorem ;

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

ϕ-SAT ;

- Given a Boolean formula with n Boolean variables and AND, OR, NOT operators, determine whether there exists an input certificate for which the formula is TRUE.

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

n variables.

→ K clauses = poly(n).

⇒ Verification is P

⇒ ϕ-SAT $\in$ NP.

NP-Complete (decision) Class ;

- A Problem X belongs to this class if :

1) $X \in NP$.

2) SAT $\leq_p$ X. { Satisfiability Problem can be reduced to the problem in polynomial time. }.

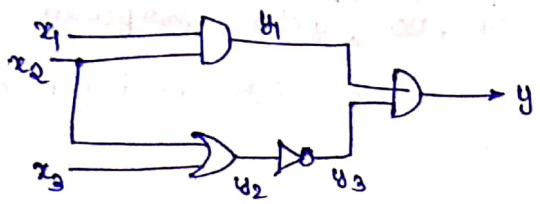
↳ or a known NP complete Problem.

- SAT $\in$ NP-Complete.
- ϕ-SAT $\in$ NP-Complete. [Because ϕ-SAT $\in$ NP and SAT $\leq_p$ ϕ-SAT].

Means : If I can solve ϕ-SAT ; I can solve SAT.

* Any Problem which belongs to NP but is not NP-complete ?

* If decision problem is NP-complete, optimization problems are also NP-complete.



→ Label all the output wires.

"Any instance of SAT can be converted to some instance of ϕ-SAT".

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

19/08/2019

ALGORITHMS-2

Class Test
1000 - 1100

NP-Complete Problems:

3-SAT: Given a Boolean formula in conjuctive normal form (CNF) in which each clause consists of exactly 3 literals, we have to decide whether there exists an input certificate for which the Boolean formula is satisfiable.

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

It has 3 variables and 4 clauses.

- 3-SAT is NP-complete;

Proof: 1) SAT $\leq_p$ 3-SAT.

2) 3-SAT $\in$ NP.

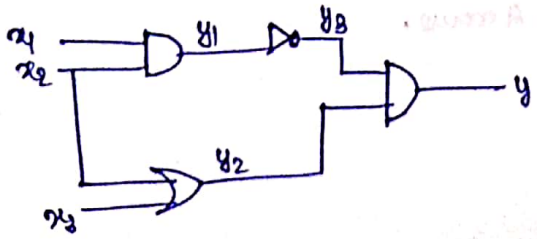
clauses, k = poly(n). } Problem statement
input vars = n.

Verification can be done in poly(n) time.

$\Rightarrow$ 3-SAT $\in$ NP — (1)

To prove (2), 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:



Input vars: x_1, x_2, x_3.
output: y.

(1) → Intermediate vars:
y_1, y_2, y_3
gates = poly(n)
n = # input vars.

(c) Equivalent Boolean formula :

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

(ϕ_1, ϕ_3, ϕ_4)

For 3-SAT, it should be in DNF (Disjunctive Normal Form);

$$\phi = \phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_{poly(n)}.$$

$$\phi_2 = x_1 \wedge x_2 \leftrightarrow y_1 = \text{3CNF?}$$

(d)

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

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

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

$$\phi_4 = (\neg y_1 \leftrightarrow y_3)$$

TRUTH TABLE: 3 columns, 4 rows.

$x_i \vee x_j$ or $\neg x_i \vee x_j$ or $x_i \vee \neg x_j$ or $\neg x_i \vee \neg x_j$

$x_i \vee x_j = x_i \vee x_j$.

ALGORITHMS - 2

SAT
↓
q-SAT
↓
3-SAT
↳ CLIQUE → Independent set → Vertex cover

- for every edge of $\bar{G}$, at least one vertex is outside the Independent set.

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

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

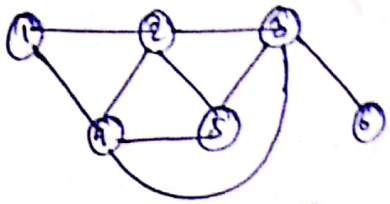
* V\I is the vertex cover of $\bar{G}$.

General Statement;

- I is an Independent set in G(V, E) if and only if V\I is a vertex cover in G. so, $\boxed{\text{Max I} \Rightarrow \text{Min V'}}$.

[V' is a vertex cover of G if for every (u, v) ∈ E, {u, v} ∩ V' ≠ ϕ].

G =



V' = {1, 3, 2, 4}

V' = {2, 3, 4} - min - vertex cover.

Incidence Matrix

A (Vertices)	B - Family (Edges)
1	1, a
a	1, 3, 4, 5
3	5, 6, 8, 9
4	2, 3, 6, 7
5	4, 7, 8
6	9

set cover.

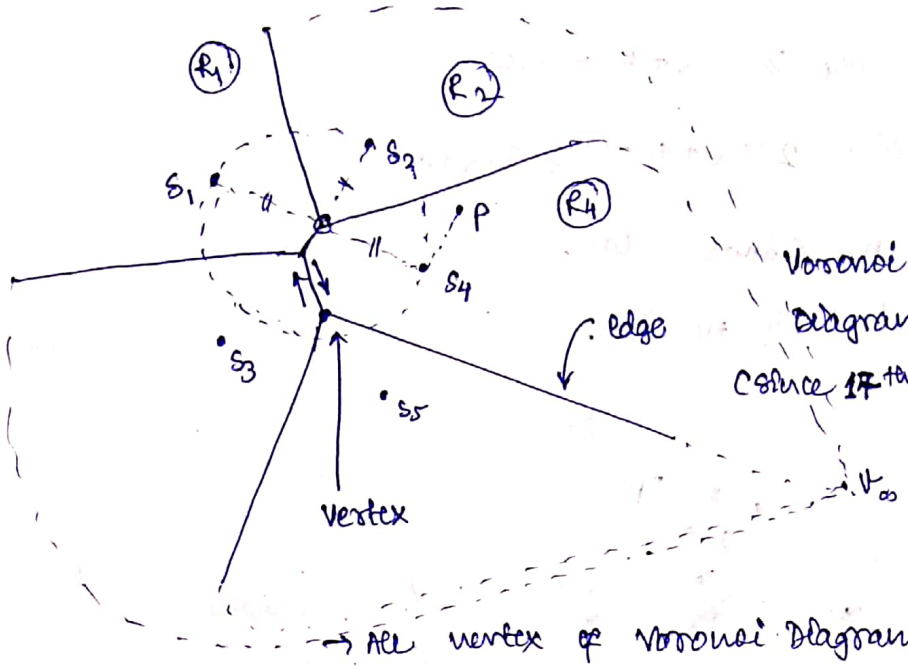
ALGORITHMS - II

09/08/2019

1000 - 1100

* $S = \{ s_1, s_2, \cdots, s_n \}$ is a set of sites on xy-plane. We have to subdivide the xy-plane into n regions:

$R_1, R_2, \cdots, R_n$ such that for any point $p \in R_i$, s_i is the nearest site.



Voronoi Diagram (VD).
(since 17th century).

→ All vertex of Voronoi Diagram are circumcentres for 3 sites.

→ But all circumcentres are not vertices.

* Assume: ① All sites have distinct x- and y- coordinates.
② No four sites are concyclic.

p is a vertex

⇔ ~~largest~~ empty circle centered at p is a circumcircle of ~~three~~ sites.

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

* How many vertices and edges are present in VD?

regions = n.

Euler's formula: (planar graphs) $(n_v+1) - n_e + n = 2$ — (1).

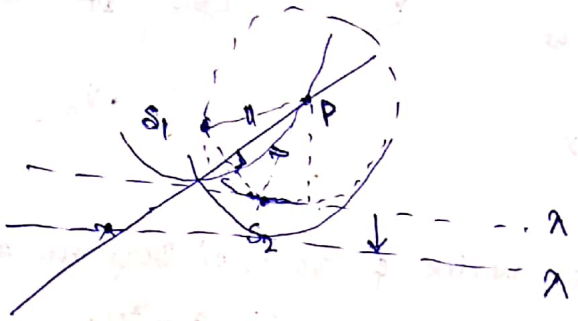
sum of degrees of all (n_v+1) vertices $\geq 3(n_v+1)$.

$\Rightarrow 2n_e \geq 3n_v + 3$ — (2).

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

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

similarly $n_e = O(n)$.



05/09/2019

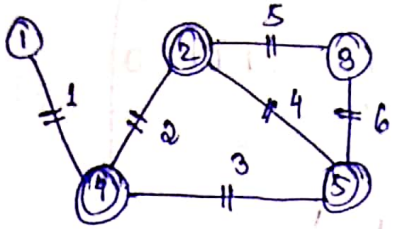
ALGORITHMS-II

1000-1100

#Vertex Cover Problem:

- Given an additional graph $G(V,E)$ and a natural number k, determine whether G has a vertex cover of size k.
- [Def: $V' \subseteq V$ is a vertex cover of $G(V,E)$ if each edge $(u,v) \in E$ has either u or v (or both) in V'.]

- ex:



$V = \{1, 2, \cdots, 5\}$

$E = \{e_1, e_2, \cdots, e_6\}$

$V' = \{2, 4, 5\}$

	e_1	e_2	e_3	e_4	e_5	e_6	Decimal value (radix = 2)
1	1	0	0	0	0	0	~~5120~~
2	0	1	0	1	1	0	4352 ✓
3	0	0	0	0	1	1	~~4101~~
4	1	1	1	0	0	0	5440 ✓
5	0	0	1	1	0	1	14177 ✓
0	0	0	0	0	0	1	1 ✓
0	0	0	0	0	1	0	4 ✓
0	0	0	0	1	0	0	~~16~~
0	0	0	1	0	0	0	~~64~~
0	0	1	0	0	0	0	~~256~~
Column sum → 0	1	0	0	0	0	0	1024 ✓
3	2	2	2	2	2	2	15,018

#Subset sum problem.

Given $S \subset \mathbb{N}$ and $t \in \mathbb{N}$, determine whether some elements of S add up to t.

(N) → (1) : Photo taken :

09/09/2019

ALGORITHMS - II

NP-Complete Problems :

- Hamiltonian Cycle Problem (HamCycle). ⟵ Reduction.
- Travelling Salesman Problem (TSP).

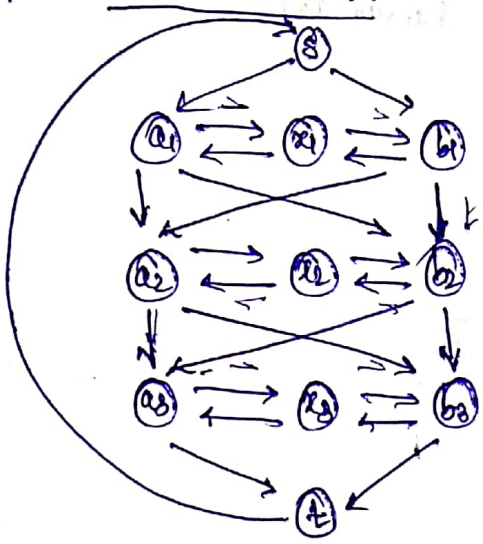
HamCycle :

- Given a (directed or undirected) graph $G(V, E)$, determine whether G contains a cycle passing through all the vertices.

3-SAT : - Given ϕ, determine whether all clauses are TRUE for some input.

→ HamCycle instance $G(V, E)$.

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



$\phi(1, 0, 1) = 1$.