App^n's of SSSP

eg. 1) Routing Protocol works on this algorithm.
2) Google Maps
3) Social websites (How X & Y are connected ?)

23/04/18

NP

Algo :

L ⟵ A

very specific instructions ⟵ well defined inst^n's

finite time

⇓ o/p

also very specific

AI ⟶ AO ⟶ to be verify in polynomial time

(App. I/p)

like : $i \leftarrow i+1$ ⟹ Deterministic Statements

Suppose to solve L, we have 4 steps :

S_1

S_2

S_3

S_4

Till now, we have seen for algo's : $O(n^k)$

$\uparrow$ size of problem

$\Rightarrow T(n) = a_1 x^3 + a_2 x^2 + a_3 x + a_4$

$$f(x) = \sum_{i=0}^{n} a_i x^i$$

$\uparrow$

Polynomial Expression

will have $n + \log^2 n$

$\Downarrow$

$n + n^{\cancel{2}\log}$

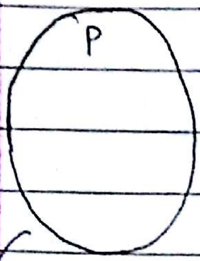
P → set of those algo's which 'ne complexity of type Polynomial Expression

↓

very specific instⁿ

$\Downarrow$

deterministic → ~~not~~ gives same result for same i/p



Set of deterministic ~~algorithm~~ polynomial time Algorithm & ~~verifiable in same time~~

when all steps in your algo are deterministic

$\Downarrow$

your algo is deterministic

→ should check whether appropriate i/p gives app o/p

↓ have to check in finite time

Eg. I/P [5 | 3 | 9 | 8] $\xrightarrow{\text{sort}}$ O/P [3 | 5 | 8 | 9]

check : $3 \le 5$ ✓
$5 \le 8$ ✓
$8 \le 9$ ✓

∴ O/p is ✓

(have to check in polynomial time)

→

Heap- sort (A)
{

ND ← ① Build - max - heap(A)

D { for $i \leftarrow$ len (A) to 2
do exchange $A[1] \leftrightarrow A[i]$
heap - size [A] ← heap-size[A]-1

just know it produces o/p.

∴→ should know what this funcⁿ is. ~~It states~~ We claim ∃ some steps ~~∃~~ but don't know how to implement

$\Downarrow$

Non - deterministic

ND ← ~~100~~ ② max_heapify (A, I)

In this, combⁿ of D & ND

↳ If algo has any ND step ⇒ algo is ND

→ This Heap - Sort algo : Non deterministic

↓

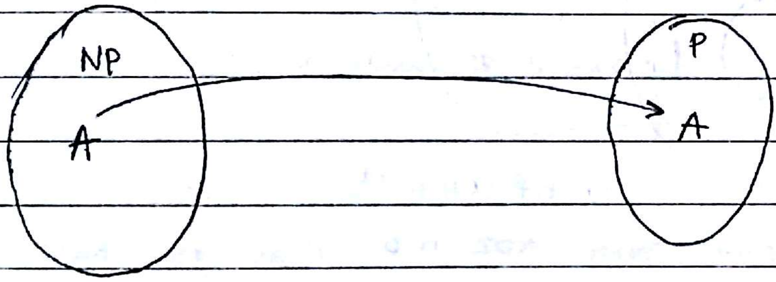
suppose it generates

| 5 | 3 | 9 | 8 | → | 3 | 5 | 8 | 9 |

can again check whether O/p is correct or not in Polynomial time

→ ~~Algo is ND if~~ :

→ Algo ∈ NP class : ~~Algo is~~ ND but polynomial Time verifiable.

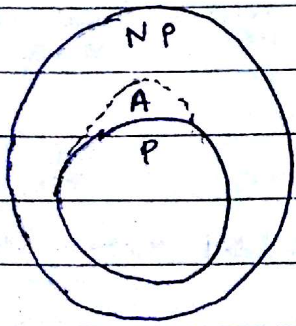


→ Once we know defⁿs of ① & ② ~~fun~~ steps :

ND ⟶ D both become D ⇒ Algo becomes deterministic

⇓

fnt in P class



Each P is subset of NP

As we get defⁿ of ND steps in ND Algo in NP

⇒ NP ⟶ P

⇒ P set expands

→ whatever we've in P now was part of NP ~~at~~ earlier at some point of time.

Teacher's Signature..................

polynomially $(O(n^k))$
solvable

Problems

Easy → can find algo

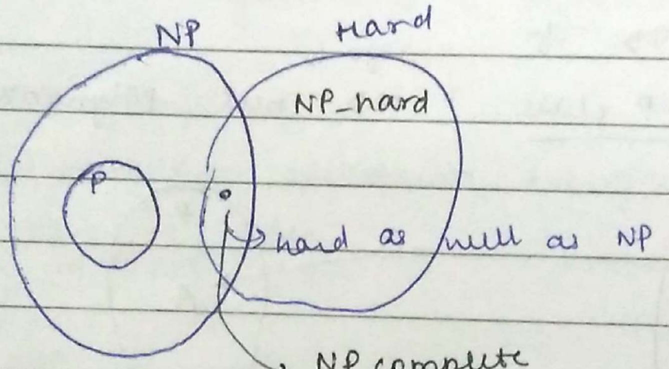
↳ All algo in P : easy

↳ Some NP also : easy.
(like Heap sort)
those algo from which we're trying to find better sol^n
(sol^n is already in P, but we want better complexity)

Hard

↳ exponentially solvable
$O(k^n)$

↳ ND Algo for hard problem
If we can verify sol^n in Polynomial time



NP Hard : set of Polynomial time ~~NOD~~ ND Algo for hard problem
+ ~~(Polynomially) exponential~~ sol^n is

NP complete = NP Hard + sol^n is polynomially ~~exp~~ verifiable

↳ A' → { S_1 — D ⟷ P
S_2 — ND ⟷ P
S_3 — D ⟷ P }

NP hard

How it will complete in poly. time → we don't know

To check NPC ⟵ sol^n is ~~is~~ ~~d~~P time verifiable

Scanned by CamScanner

Reduction : Try to

Unknown problem $\xrightarrow{\text{reduce}}$ known NP, hard problem

Known NP Problems

1. CNF - SAT

(Conjuctive Normal form satisfiability Problem)

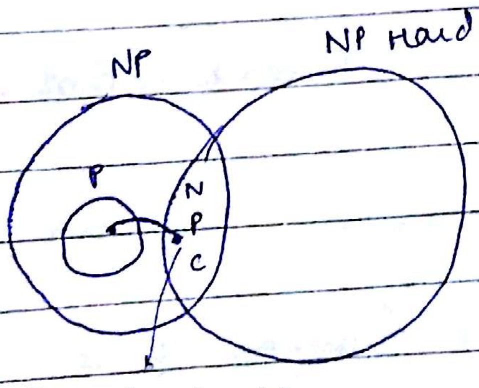
Cook - Levin : Paper

↓ proved

If we are able to develop a polynomial time Algo for any of the NPC problem, then we can develop " " " " for All NP Problem.

It gives result :

$\boxed{P = NP}$ Open Problem



Circuit L Problem

Problem :-

$$S = \underbrace{(x_1 \cup \bar{x}_2 \cup \bar{x}_3)}_{C_1} \wedge \underbrace{(\bar{x}_1 \cup x_2 \cup \bar{x}_3)}_{C_2} \wedge \underbrace{(\bar{x}_1 \cup x_2 \cup \bar{x}_3)}_{C_3}$$

C_1
↓
Clause 1

x_i are boolean variable

$x_i = \{0, 1\}$

↳ Have to find all possible values which satisfy s

x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

To solve problem of size = 3
no. of sub-problems $= 2^3$ (8)

→ can we check ~~this satisfies s~~ in Polynomial Time? ← Yes (substitute & check)

If problem size : n
complexity : 2^n (atleast)
⇓
$\boxed{\text{Hard problem}}$

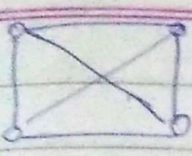
It will be more difficult to solve if x_i has more choi

* It is the base NP problem. Research has already been done.

2. Clique Problem

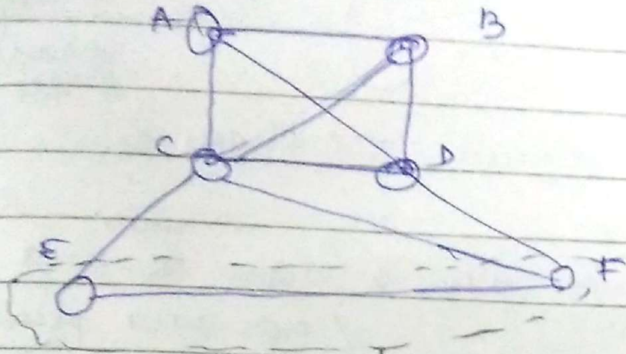
To find max. Clique in a ~~problem~~ graph

Clique : Size of Complete subgraph in a graph
↓
no. of vertices



max
$\Rightarrow$ Clique $= 4$

4



$\hookrightarrow$ $K=2$

clique :

$K=1$ ✓ (move to same vertex) ✓ Every

$K=2$ ✓ $\rightarrow$ EF

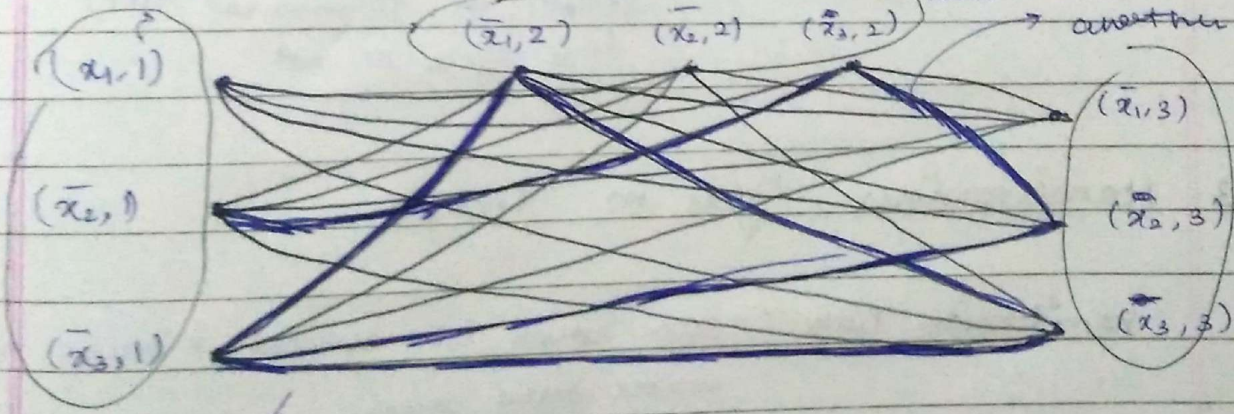
$K=3$ ✓ $\rightarrow$ CEF

$K=4$ ✓ $\rightarrow$ ABCD

$K=5$ ✗

CNF
circuit
CNF
well known

* Now, we'll try to reduce clique $\longleftrightarrow$ CNF

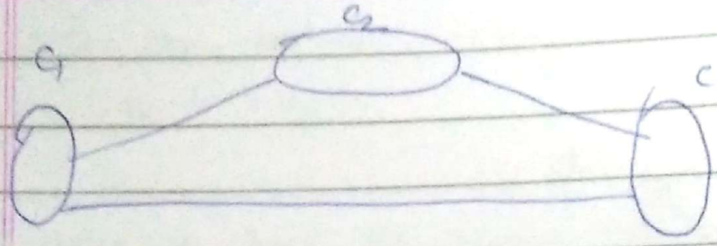


clique of len $= 3$

edge represents : end of edges are 1 $\rightarrow$ No edge
can't have x_1 & $\bar{x}_1$ 1 at same time

within a clause $(x_1 \cup \bar{x}_2 \cup \bar{x}_3)$
if we know $x_1 = 1$, don't have to check for $\bar{x}_2$ & $\bar{x}_3$

Teacher's Signature.................



→ ~~There could be~~ a clique ($k=3$) because we want all C_1, C_2, C_3 to be

→ have to Reduce from circuit → clique

We may get more than 1 clique with $K = 3$ (more pairs possible to satisfy S $\{x_1, x_2, x_3\}$)

If we show this

→ ~~Circuit~~ Clique : NP Hard Problem.

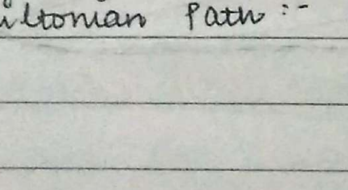
↳ From Highlighted : we've reduced

→ clique also NP Hard Problem.

Once get it (path) : can check (verify) sol^n seeing graph in polynomial time
↳ (if cycle or not)

3. Hamiltonian Cycle in Graph :

Hamiltonian Path :- can cover all vertices, visiting each vertex only once (simple path)



cycle (return to start point)

↳ If you 're magical f^n to know whether G has HP or not

Steps to verify :

1. Pick an edge
2. Remove that edge from the graph
3. Check : G have HP (True for all edges)

$\Downarrow$

Hamiltonian ~~Path~~ Cycle exists

If any False ⇒ NO

Don't need to proceed further,
Just give the output.