29 - Arsh
1 - SE
9 - JAMO
8 - CS
13 - IM
15 - CSE

Page No.
Date / /

8

NP

NP Completeness

P

Class P

YES/NO problems with polynomial-time algorithm.

Class NP

YES/NO problems with polynomial-time checking algorithm like, given a solution, we can check in a polynomial time if that solution is what we are looking for

NP-hard

A problem is NP-hard if all other problems in NP can be polynomially reduced to it.

NP-complete

A problem is NP-complete if it is (a) in NP
(b) NP-hard

OR

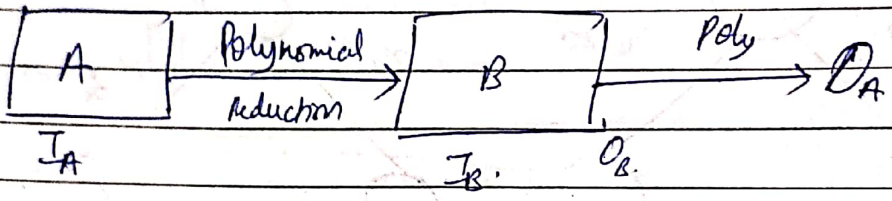
The most difficult problems in NP

DELTA Page No. Date / /

So, for a given problem A, if we want to prove the problem is NP-hard or NP-complete (thus a polynomial-time algorithm very unlikely).

Reduction

When a problem needs to be polynomially reduced so that it can be formed by another software and then again transformed.

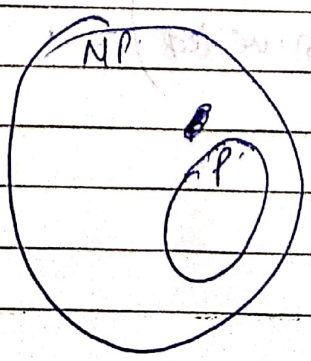


Problem A was reduced to Problem B

A $\xrightarrow{\text{reduction}}$ B

P → Deterministic Polynomial time taking algo.

NP → Non-deterministic Polynomial time taking algo



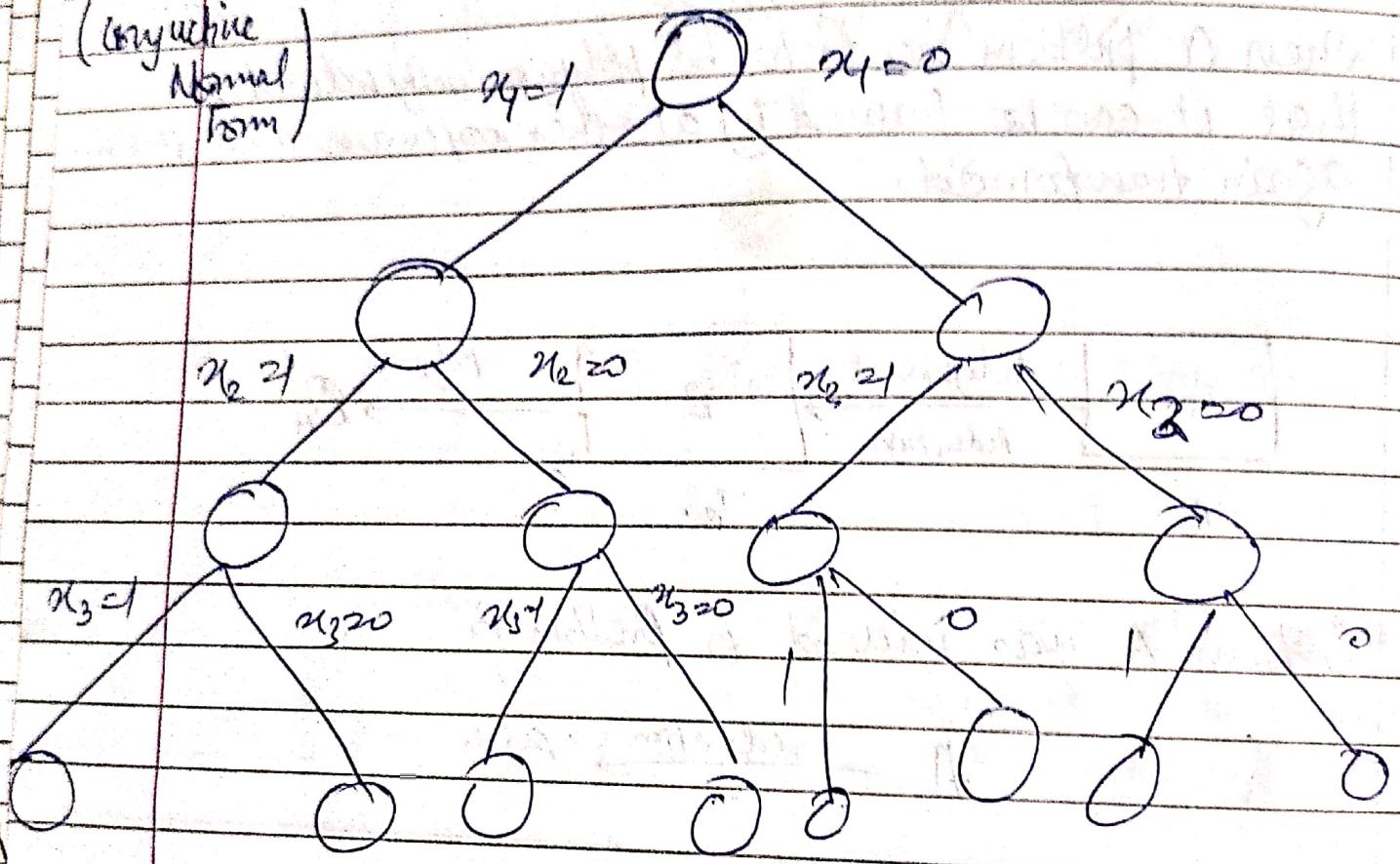
All problems are NP till the time we find a deterministic approach for them. When we have found a deterministic approach for them, then they become P problems.

CNF-Satisfiability

$x_i = \{x_1, x_2, x_3\}$

CNF → $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$

(Conjunctive Normal Form)



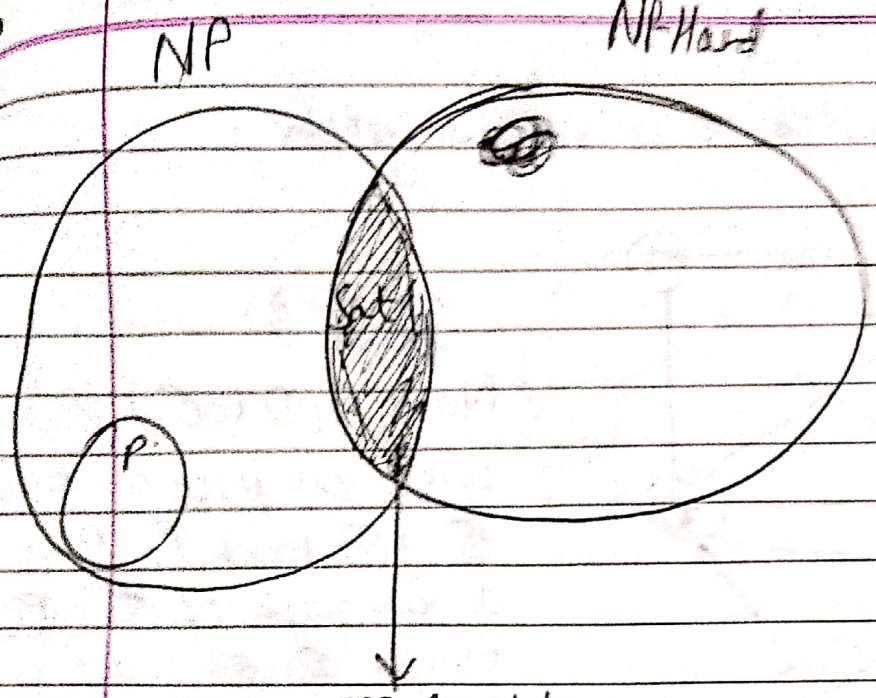
Reduction

Polynomial.

Sat $\propto$ 0/1 Knapsack.

reduces

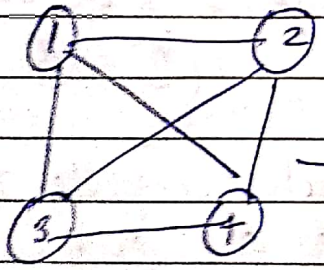
(A part of a similar problem is converted) → (To a part of Hard problems).



$P \subseteq NP$

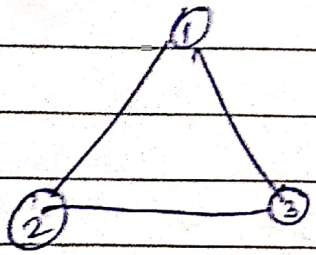
but we have to prove $P = NP$

CDP Clique Decision Problem



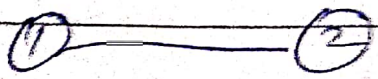
Complete Graph, where vertices connect all other.

→ Here, no. of vertices are ⇒ 4



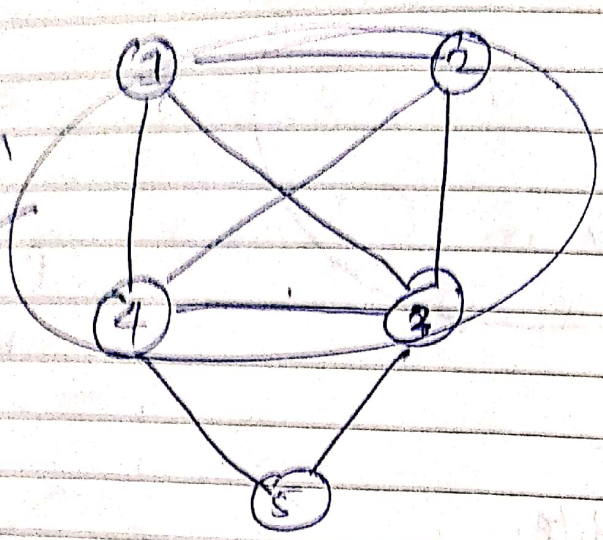
Its also a complete graph,

→ Here, no. of vertices are ⇒ 3



→ No. of vertices ⇒ 2

Now, add a vertex to a complete graph



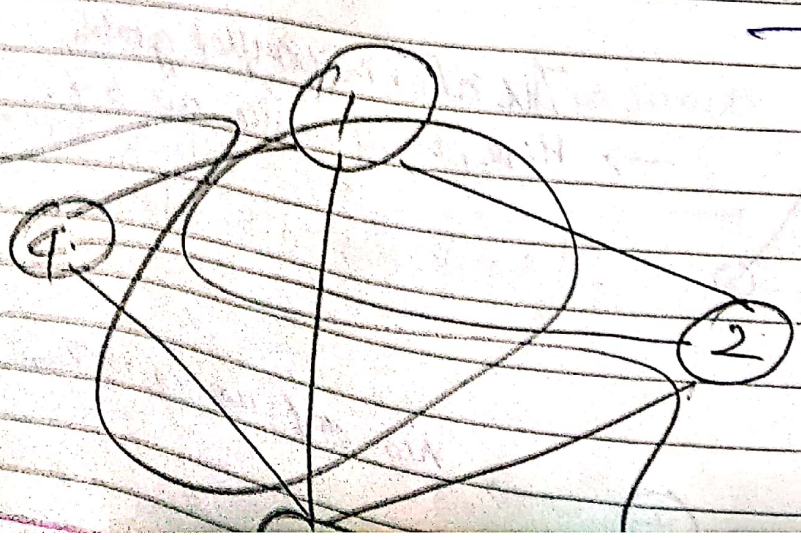
complete graph

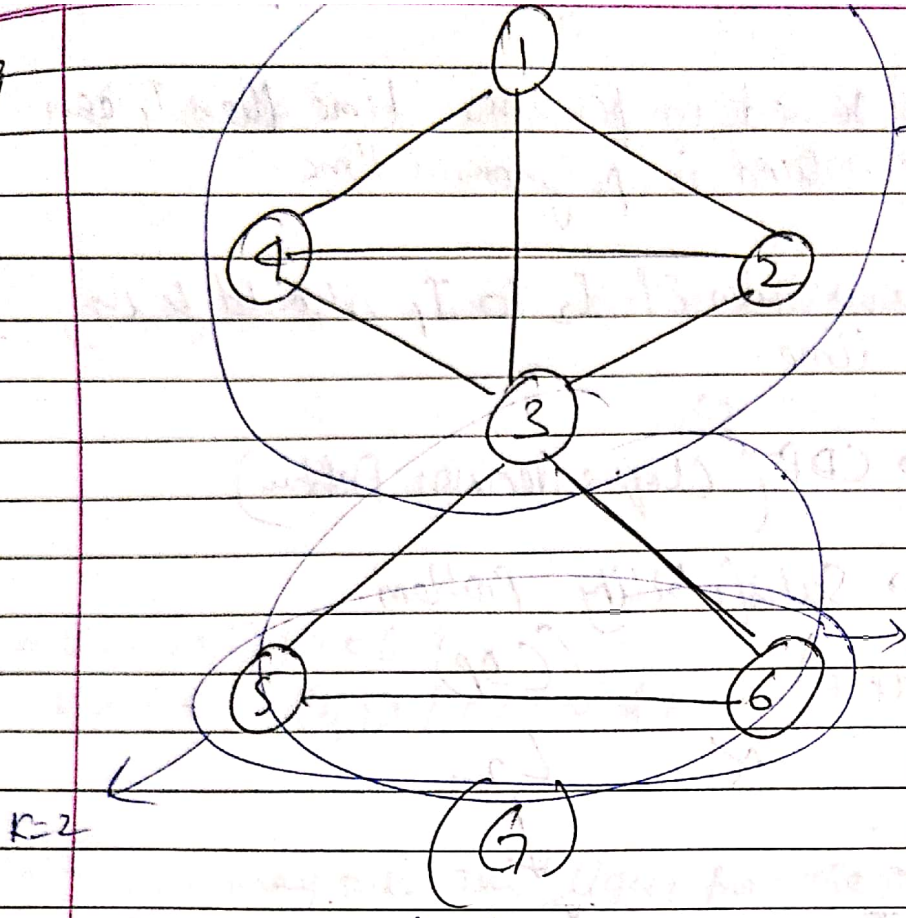
Now, G(① to ⑤) is not a complete as only ③ & ④ touch ⑤. But a sub-graph of Graph(① to ⑤) that is ①, ②, ③, ④ is a complete graph, then G(① to ⑤) is now having a clique.

So, G(① to ⑤) is now having a clique that is G(① to ④).

A sub-graph of a graph, which complete and the graph is not, then it's called a clique.

Eg





→ complete graph

K=4

V=6

K=3

Max clique (K) = 4.

Decision Problem: Find if a graph is having a clique of size K or not.

Optimization " : Find the max. clique in graph.

Now, For proving NP-hard:-

→ Now if we have a prob L_2 that needs to be proved NP-hard, then select a problem L_1 that is already known as NP-hard and show that L_1 reduces to L_2.

→ Select a presently known NP-hard problem and then reduce is to Our original problem L_2.

→ ∴, we take an example of problem L_1 (I_1) and then prepare an example of problem L_2 (I_2), such that if

I_2 can be solved in polynomial time then I_1 can also be solved in polynomial time.

And the conversion of I_2 to I_1 should be in polynomial time.

So, L_2 → CDP (clique decision Problem)

L_1 → Satisfiability Problem.

(Satis) (CDP)

$L_1 \propto L_2$

I_1 I_2

So, now, we need to prove that Satisfiability leads to CDP.

→ So we need to take an example from " and from that we should prepare a graph with clique

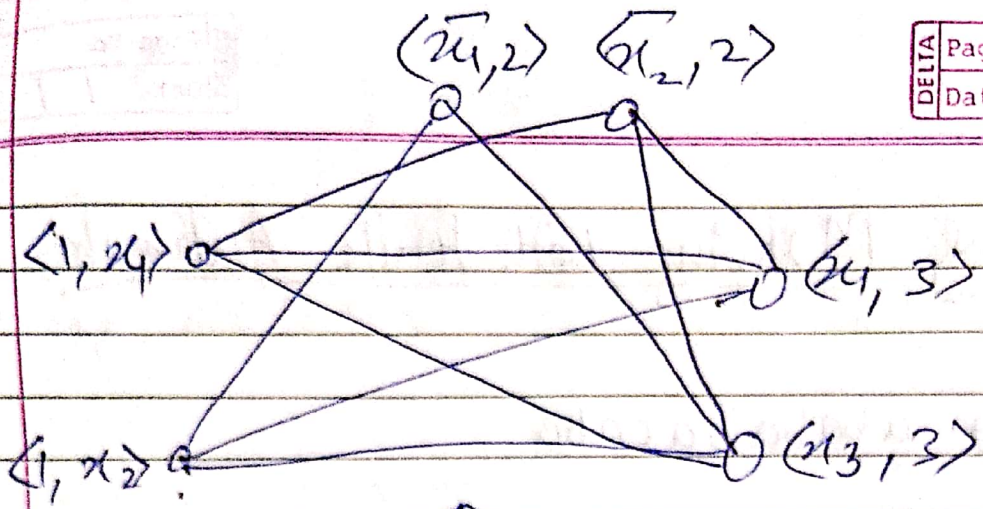
(x_1, x_2, x_3).

$F = (x_1 \vee x_2) \wedge (\bar{x_1} \vee \bar{x_2}) \wedge (x_1 \vee x_3)$

C_1 C_2 C_3.

CDP

$F = \bigwedge_{i=1}^{k} C_i$



$G \Rightarrow V = \{ \langle a, i \rangle \mid a \in C_i \}$

$E = \{ \langle a, i \rangle \langle b, j \rangle \mid i \neq j \text{ and } b \neq \bar{a} \}$

Hence

You may take any clique from the above graph but

$\langle x_2, 1 \rangle$, $\langle \bar{x_4}, 2 \rangle$ & $\langle x_3, 3 \rangle$ is selected.

hence,

x_4	x_2	x_3
0	1	1

As $x_4 = 1$

$\bar{x_1} = 0$

New fill x_4, x_2 & x_3 value in C_1, C_2 & C_3.

$$(\overset{0}{x_4} \vee \overset{1}{x_2}) \wedge (\overset{1}{\bar{x_4}} \vee \overset{0}{\bar{x_2}}) \wedge (\overset{1}{x_4} \vee \overset{1}{x_3})$$

$C_1 = 1$ $C_2 = 1$ $C_3 = 1$

so $C_1 \wedge C_2 \wedge C_3 = T$

$1 \wedge 1 \wedge 1 = 1$ ∴ Yes

So, it's a NP-Hard

Create set Union Find set

DELTA | Page No. | Date / /

String Matching with Finite Automata

Text ⇒ abababacaba

Pattern ⇒ abababaca , pattern ⇒ n = 7

M = {Q, q_0, F, Σ, δ}

Q ⇒ Set of finite states

$q_0 \in Q$ ⇒ Start state

F ⇒ set of final state

Σ ⇒ Input symbol

δ ⇒ Transition Function

$\delta: Q \times \Sigma \to Q$

n+1 states

δ(0,)

State	a	b	c	Pattern
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

(aa)

↓ Split into (n-1)

a a

↓

Then match so 1

(ac)

a c

↓

Don't match so 0

(abb) | abc

aβ βb | aβ βc

(abaa) | (abac)

aβα βa | aβα βc

ababb | ababc

αβαβ β | αβαβ c

(ababaa) | ababab

αβαβα β | αβαβ | βαβαb

4 4

ababacb

ababaa

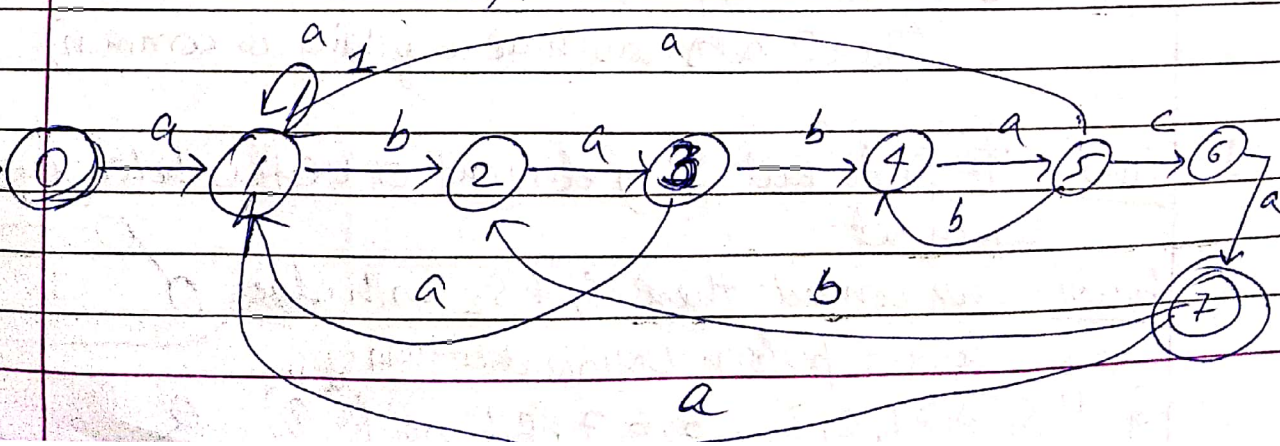
αβαβα | βαβαa

abαβα c | βαβacb

ababacabc

abaβαcα | βαβαcaa

aβαβαcαc | βαβαcαc

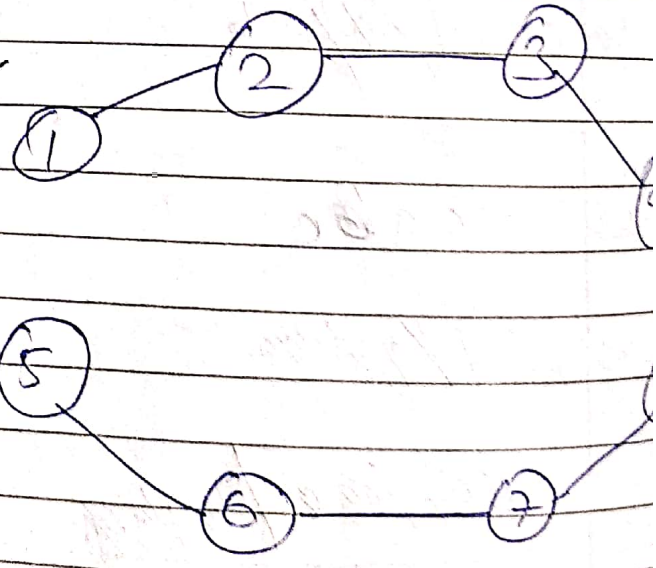


Algo

```
Finite matcher(P, δ)
{
  n = P.length
  q ← 0
  for (i = 1 upto n)
      q ← δ(q, [i])
  end loop.
  if (q is Final state) then
      String accepted
  else
      rejected.
}
```

```
Finite_Matcher(P, δ)
{
  n = P.length
  q ← 0
  for (i = 1 to n)
      q ← δ(q, [i])
  end loop
  if (q is Final state) then
      string accepted
  else
      rejected
}
```

Disjoint Sets



$S_1 = \{1, 2, 3, 4\}$

$S_2 = \{5, 6, 7, 8\}$.

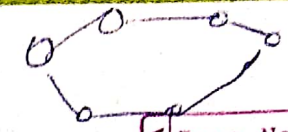
$S_1 \cap S_2 = \phi$ (Disjoint)

(These 2 sets dont have anything in common)

1. Find - It finds out any element or vertex belonging to which set
2. Union - We connect these S_1 & S_2 with edges. So, we perform UNION operation (u, v)

$S_1 \cup S_2 \Rightarrow \{1, 2, 3, 4, 5, 6, 7, 8\}$ via (4, 8)

$S_3 = \{1,2,3,4,5,6,7,8\}$ via u, v $(4,8)$ $(1,5)$



DELTA Page No. Date / /

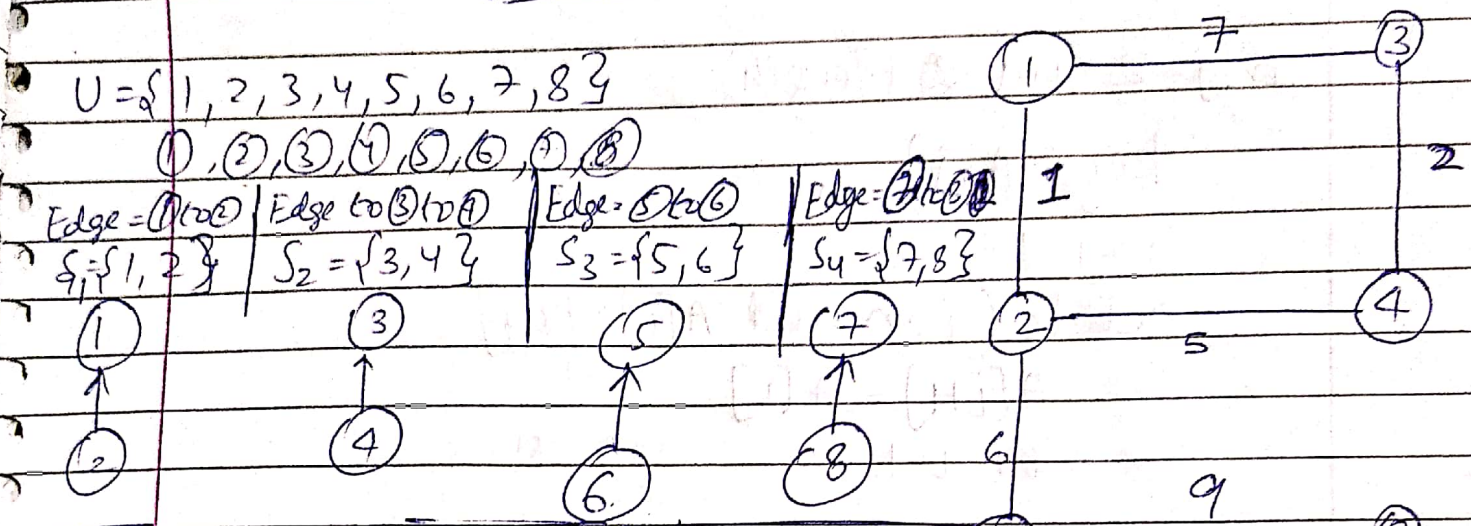
→ Now, new set ⇒ $S_3 = \{1,2,3,4,5,6,7,8\}$ found via $(1,5)$.

→ Check (u,v) i.e. $(1,5)$, do they belong to same set → Yes, they do hence there is a cycle. So, cycle detected.

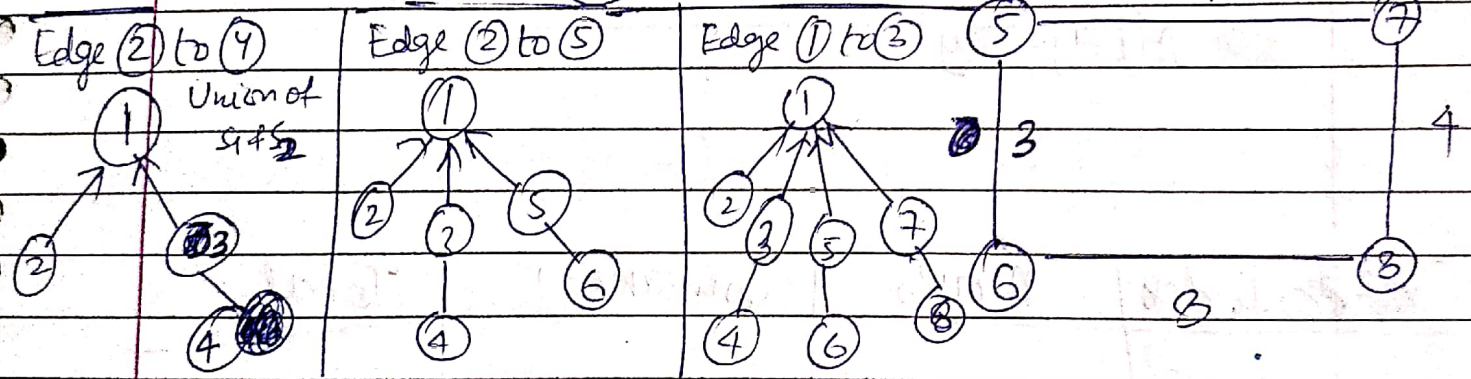
$U = \{1,2,3,4,5,6,7,8\}$

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

Edge = ① to ②	Edge ③ to ④	Edge = ⑤ to ⑥	Edge = ⑦ to ⑧
$S_1 = \{1,2\}$	$S_2 = \{3,4\}$	$S_3 = \{5,6\}$	$S_4 = \{7,8\}$



Edge ② to ④	Edge ② to ⑤	Edge ① to ③
Union of S_4 & S_2		



Now, we can use Array to do it.