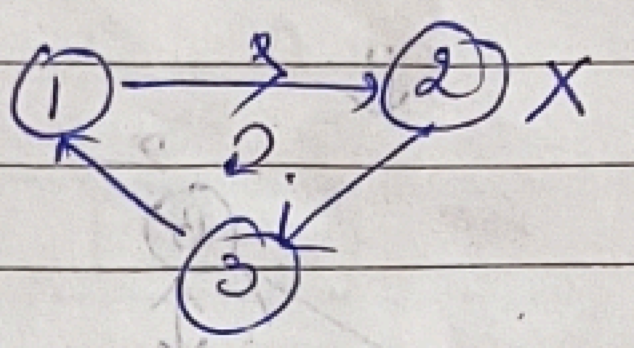
Unit :→ 3.

Graphs :→ It is a linear ordering of its vertices such that for every directed edge uv for vertex u to v, u comes before vertex v in the ordering.

→ Graph should be DAG.

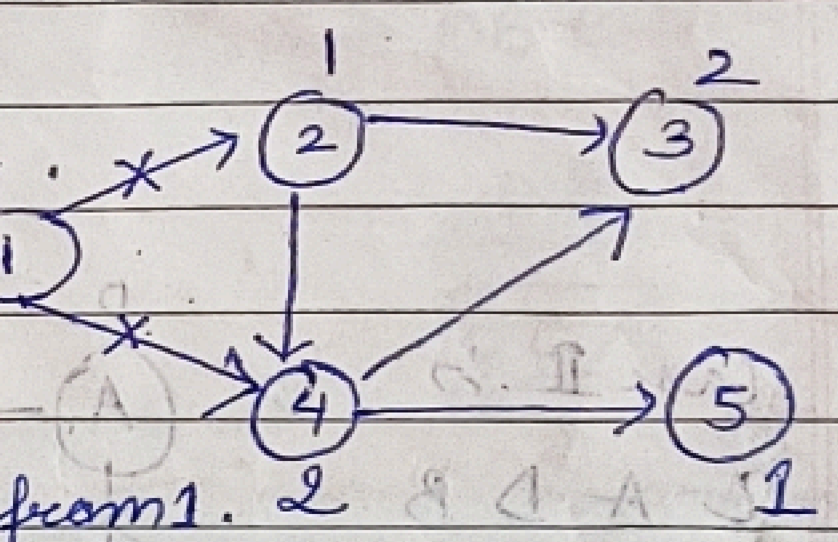
→ Every DAG will have atleast one topological ordering.

condition → It should be directed acyclic graph.

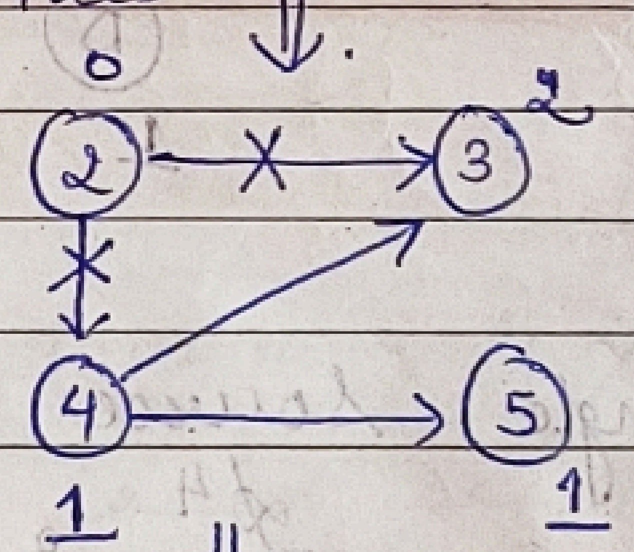


u → v

1. Find in degree of every vertex.
Start topological ordering from the vertex having indegree 0.
Select 1 and all outgoing edges from 1.
So the graph will be like this.
Update indegree of all the vertices.



⇓



⇓

2 4 3 . 5

or

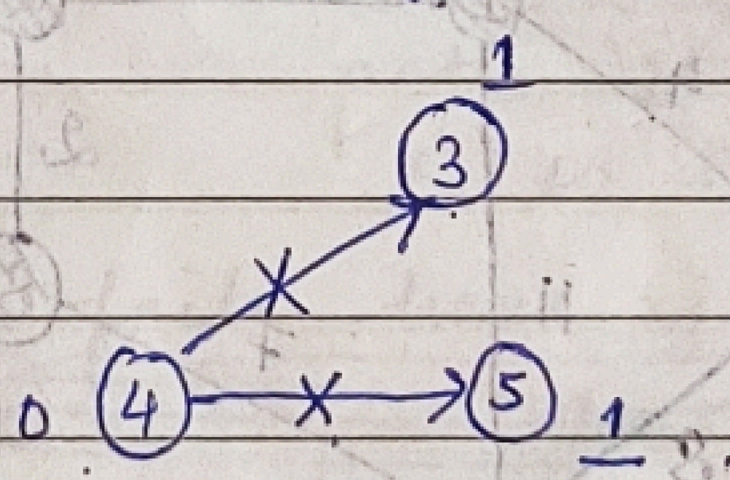
2 4 5 3.

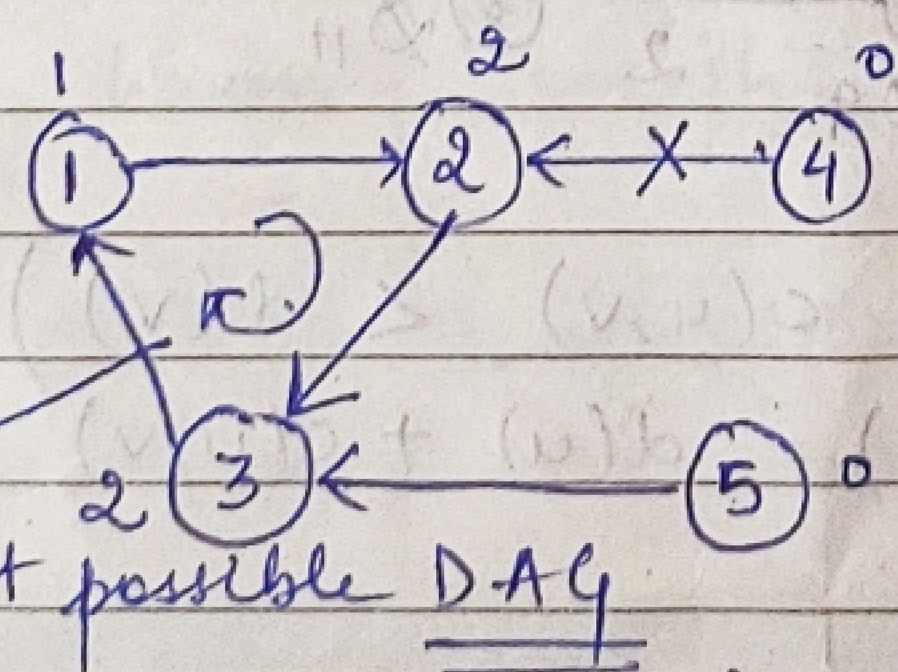
Choose any one

3 0

5 0

⇐





cycle

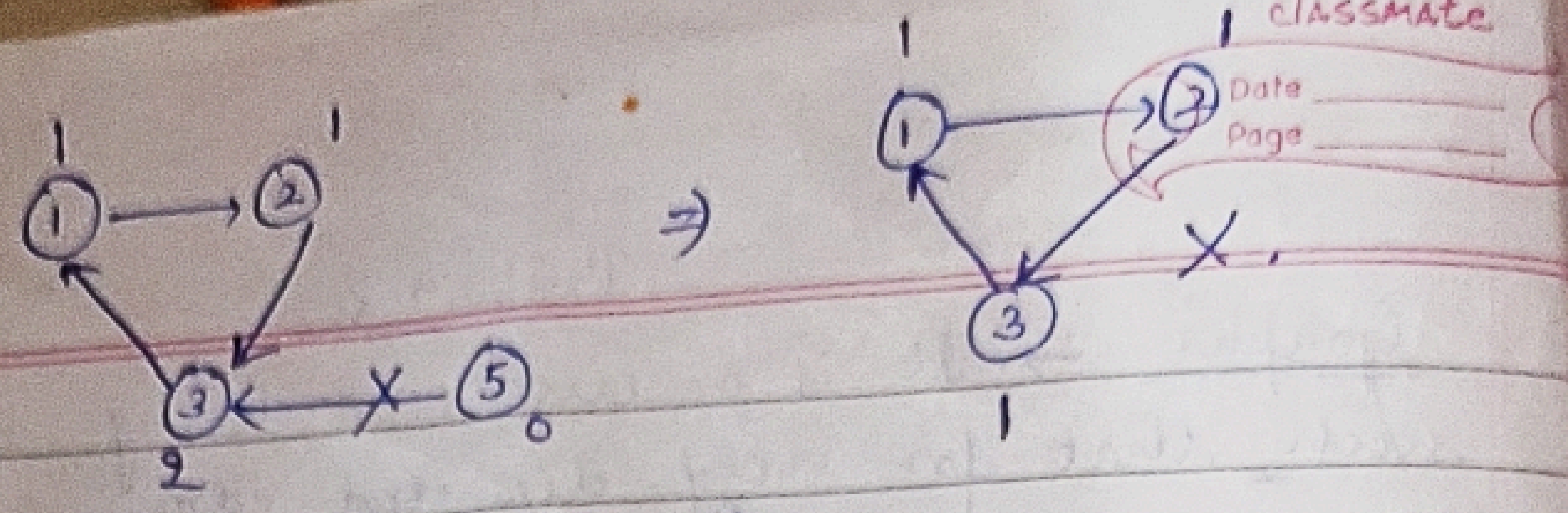
Not possible DAG

Case I :→

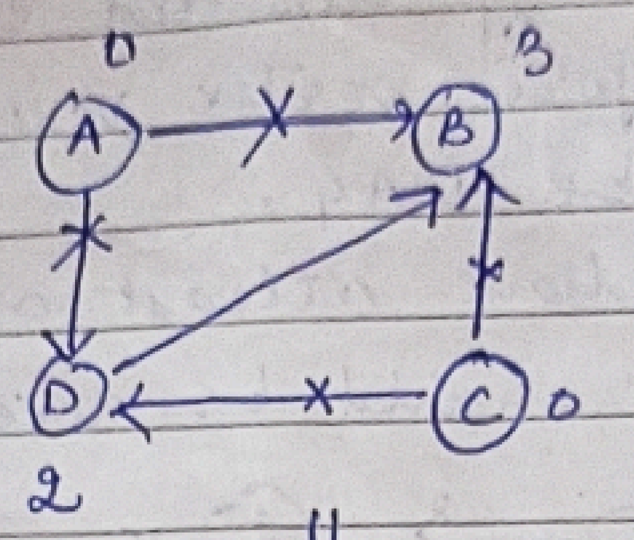
4 . 5 . — Not possible

Case II :→

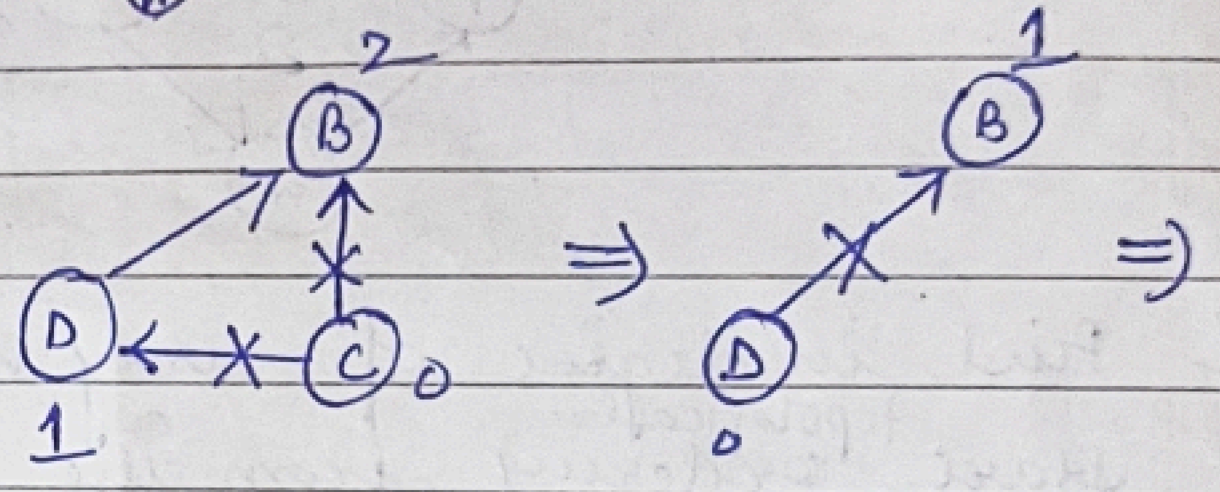
5.



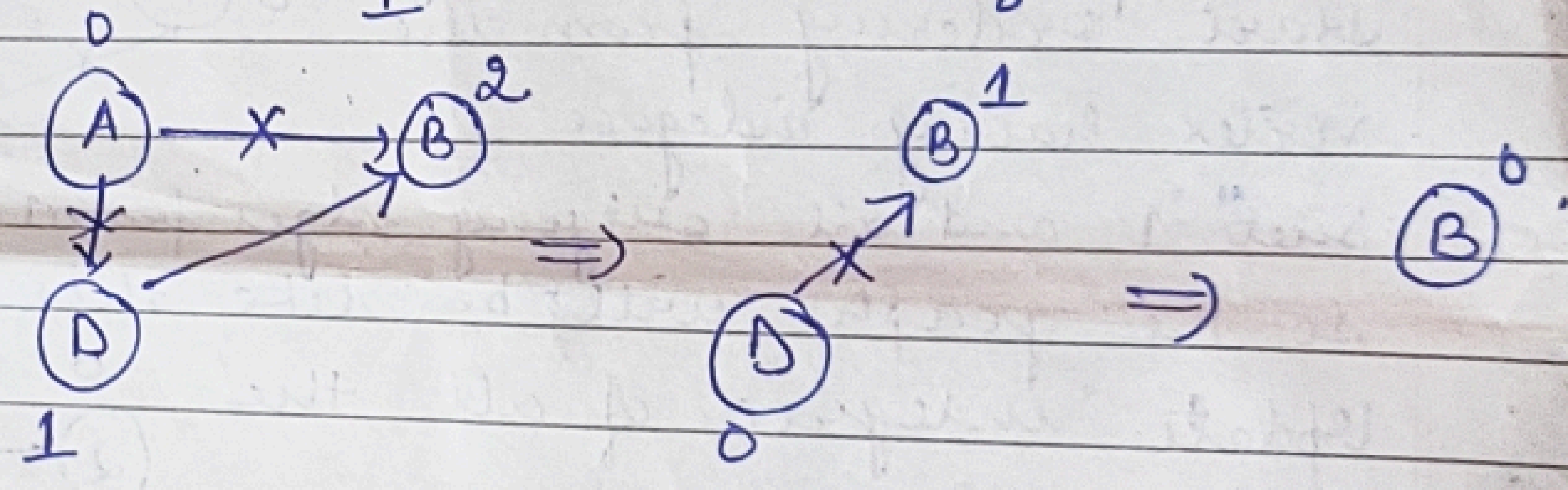
eg :- 3 :-



Case 1 :- A · C D B.

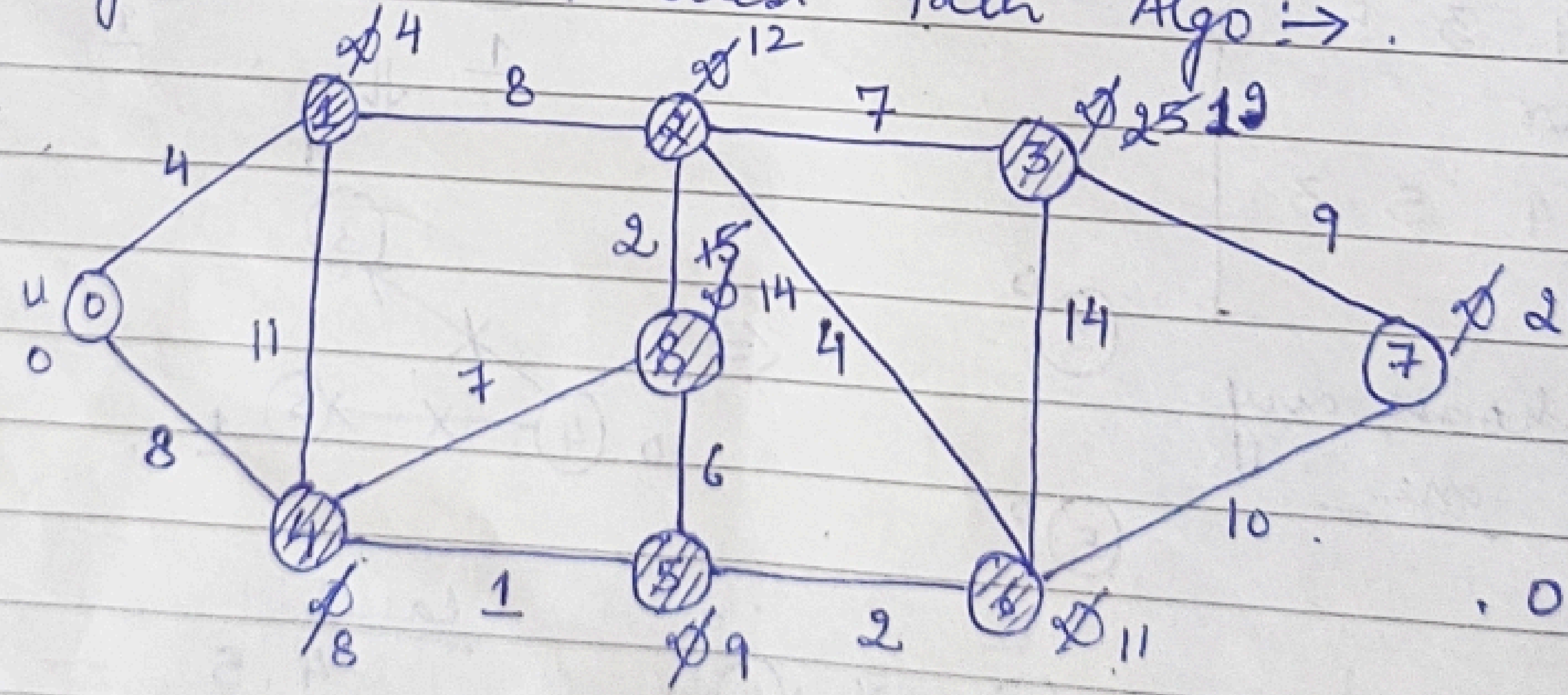


Case II :-
C A D B



Single Source Shortest Path Algo :-.

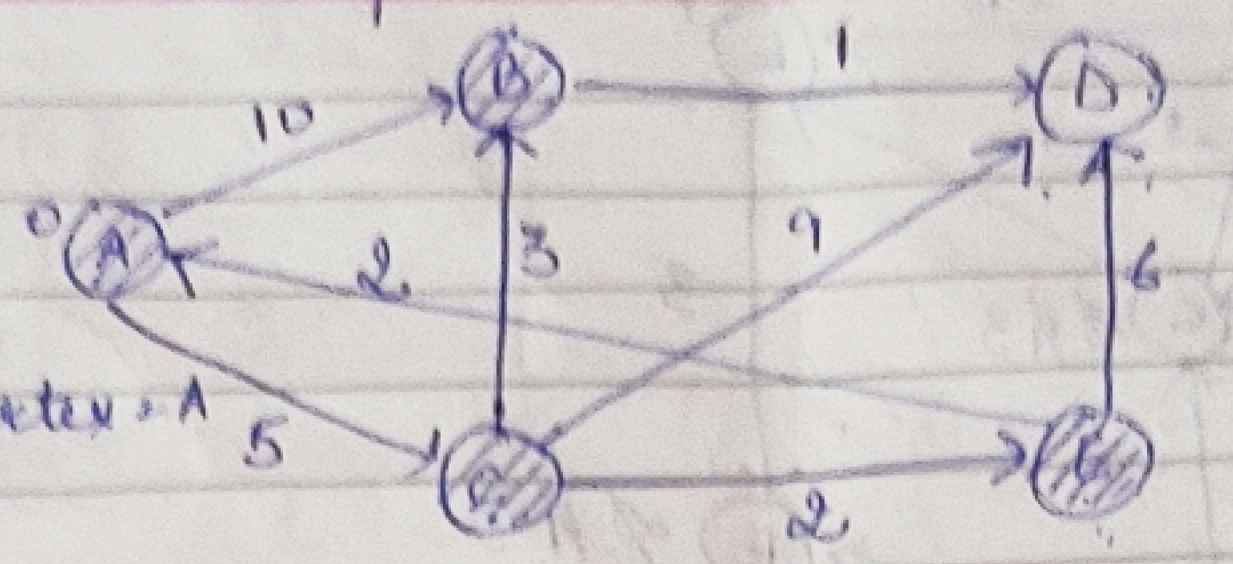
eg :- 1
Undirected graph



· 0 1 4 5 6 2
3 7

$$\text{if} \left\{ \left(d(u) + c(u,v) < d(v) \right), \; d(v) = d(u) + c(u,v) \right\}$$

Directed Graph :→



Source Vertex : A

Selected Vertex	A	B	C	D	E
A	$\boxed{0}$	~~∞~~10	~~∞~~5	∞	∞
A→C	0	8	$\boxed{5}$	14	7
A→C→E	0	8	5	13	7
A→C→E→B	0	8	5	9	7
A→C→E→B→D					

A to D.

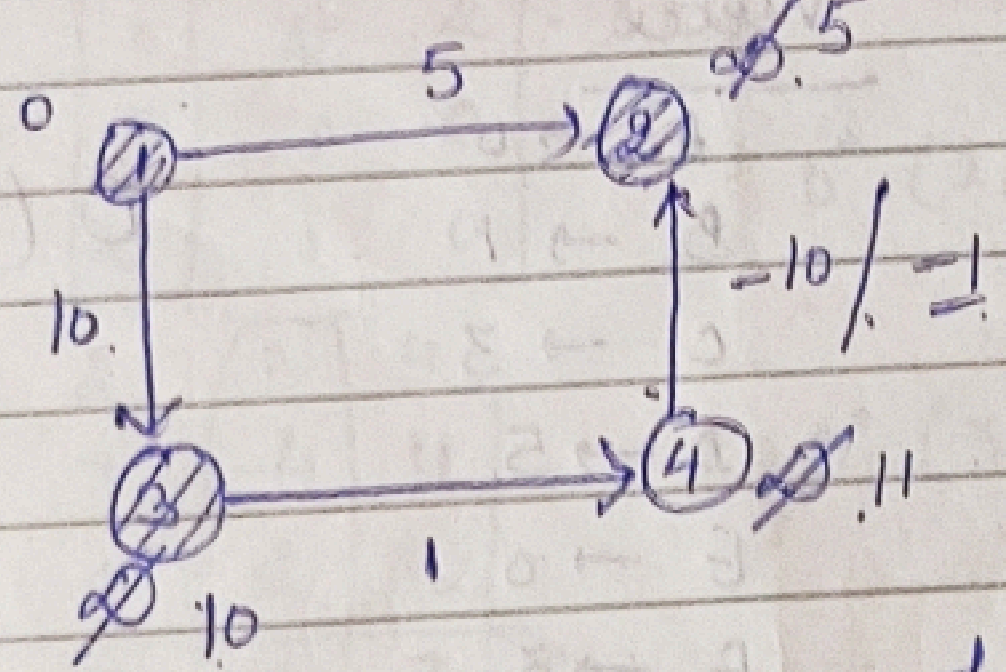
Path → DBECA → 9

A→B. → 8

BCA

ACB

It may or may not work when the edge weight is -ve.



11 - 10 → 1

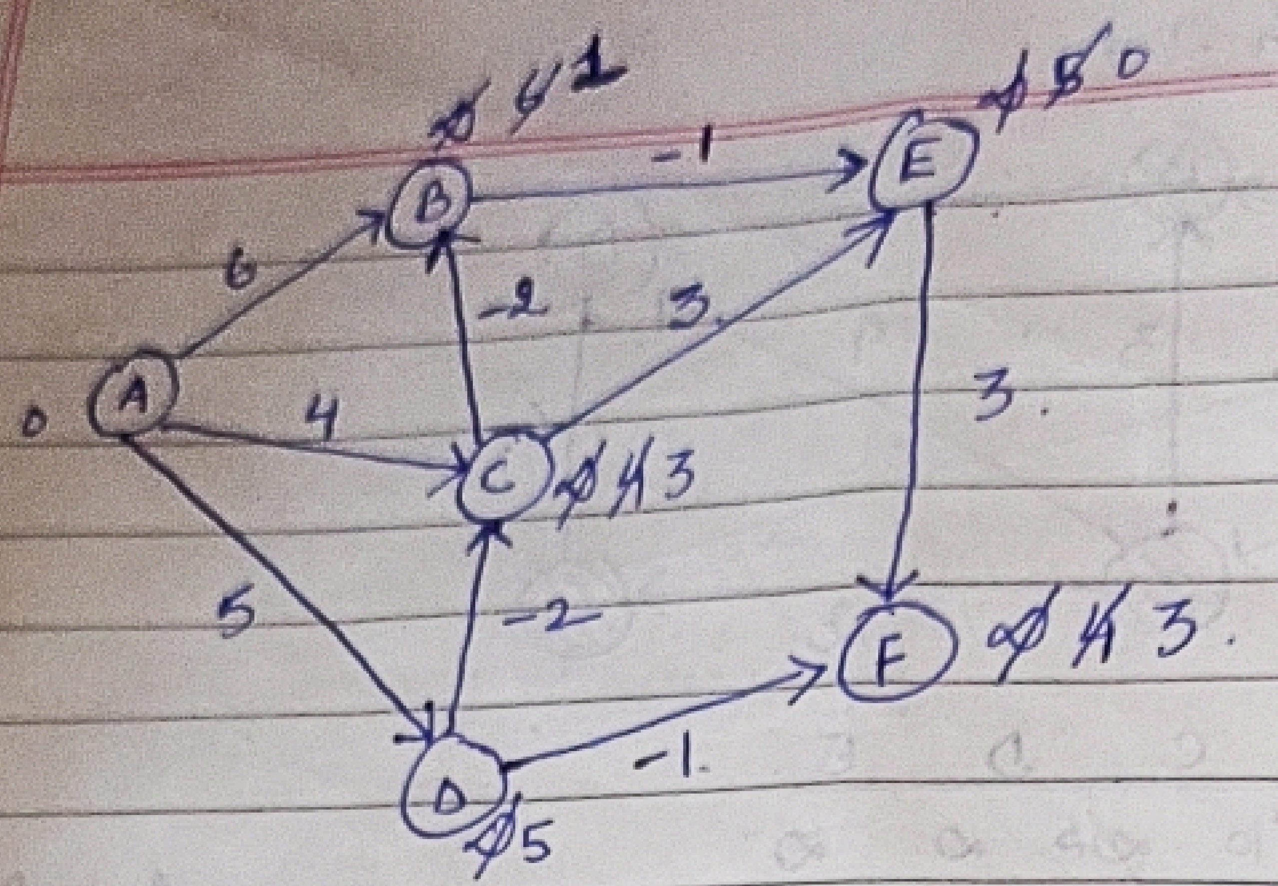
Cannot be updated already visited but the shortest path upto 2 is 1 not 5 so Dijkstra's algo fails here.

~~It surely will not work if cycle has~~ Single source to all the other nodes of a graph having weights.

→ Bellman Ford → Gives correct results even with the graph having -ve weights.

4

Date
Page



Go on relaxing all the edges $(n-1)$ times when n = no. of vertices.

$$\left\{ \begin{array}{l} \text{if } d(u) + c(u,v) < d(v) \\ \text{then } d(v) = d(u) + c(u,v) \end{array} \right\}$$

Write all the edges :→

(A,B) (A,C) (A,D) (B,E) (C,B) (C,E) (D,C) (D,F) (E,

1st ✓

2nd → 3rd → 4th →

5 times ✓

Vertex -

A → 0
B → 1
C → 3
D → 5
E → 0
F → 3

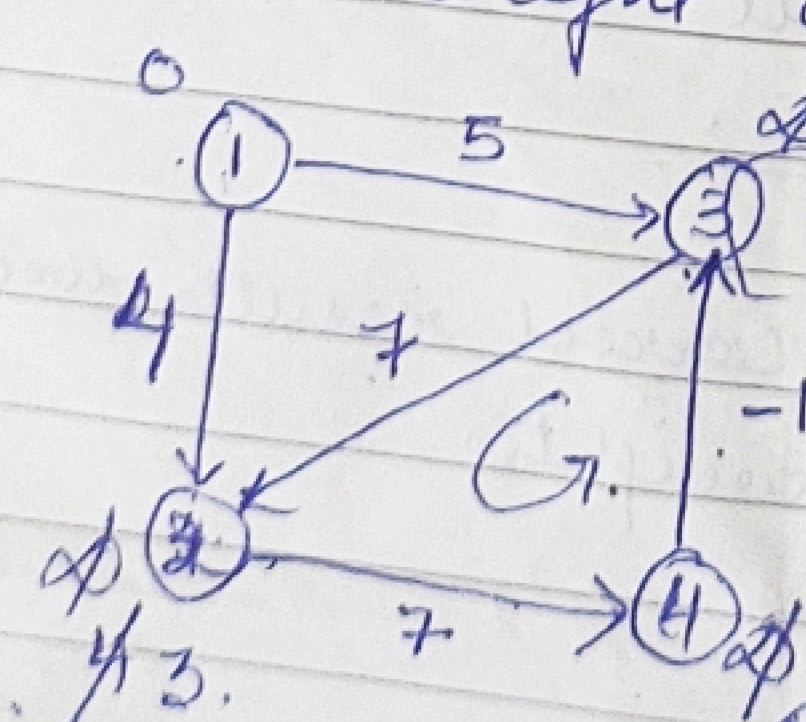
$O(E(V-1))$

$O(E \cdot V)$

$O(n^2)$

Drawback :→ It will not work if the graph contains -ve weight cycle.

1 → 0
2 → 3
3 → -5
4 → 10



sum of edge weight is -ve

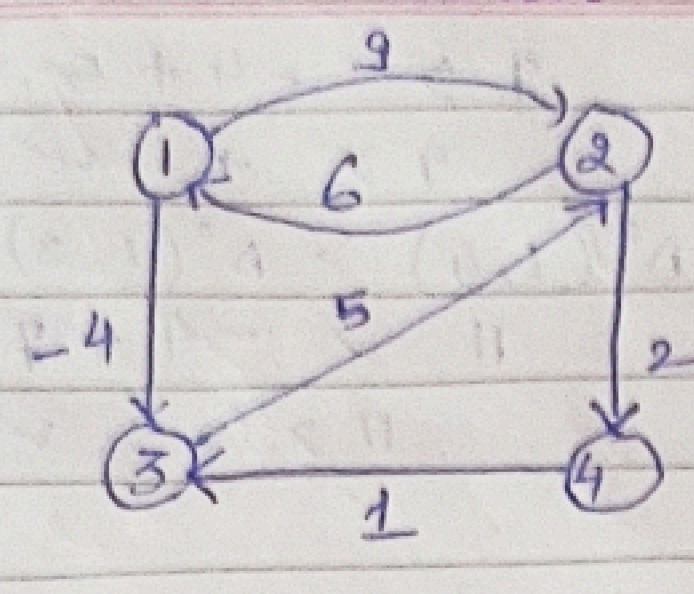
edges (1,3) (1,2) (2,4) (3,2)

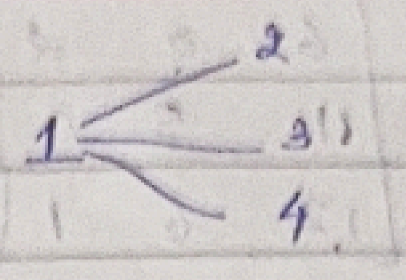
1st →

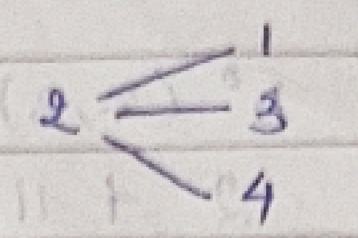
IInd → IIIrd →

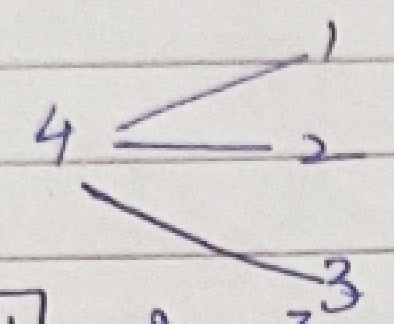
Acc. to this we are done but if we continue the iteration again changes are going on. this means it is not giving correct answer.

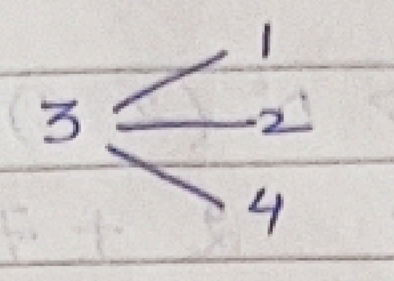
All Pair shortest Path (Floyd Warshall Algo) -

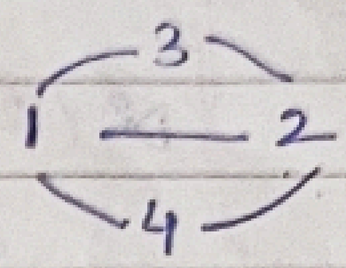












$D^0 \Rightarrow$

	1	2	3	4
1	0	9	-4	∞
2	6	0	∞	2
3	∞	5	0	∞
4	∞	∞	1	0

$D^1(4,3) \Rightarrow D(4,2) + D(2,3)$

$1 \ngtr \infty + 2$ X.

$D^1 \Rightarrow$

	1	2	3	4
1	0	9	-4	∞
2	6	0	2	2
3	∞	5	0	∞
4	∞	∞	1	0

$D^0(2,3) = D^1(2,3)$

$\infty > D^0[2,1] + D^0[1,3]$

$6 - 4 = 2.$

$D^0(2,4) > D^1(2,4)$

$D^0(2,1) + D^0(1,4)$

$2 > 2 + \infty = \infty$. X.

$D^0(3,2) > D^0(3,1) + D^0(1,2)$

$\infty + 9$

$5 > \infty$ X.

$D^0[3,4] > D^0[3,1] + D^0[1,4]$

$\infty > \infty + \infty.$

$D^0[4,2] \ngtr D^0[4,1] + D^0[1,2]$

$\infty \quad \infty + 9.$

$D^0[4,3] > D^0[4,1] + D^0[1,3]$

$1 \quad \infty + (-4).$

$D^1[4,1] \Rightarrow D(4,2) + D(2,1)$

$\infty \ngtr \infty + 6$ X.

$D^1(3,4) > D(3,2) + D(2,4)$

$\infty > 5 + 2.$ ✓

$D^2 \Rightarrow$

	1	2	3	4
1	0	9	-4	11
2	6	0	2	2
3	11	5	0	7
4	∞	∞	1	0

$[1,3] \Rightarrow D[1,2] + D[2,3]$

$-4 \ngtr 9 + \infty$ X

$[1,4] \ngtr D[1,2] + D[2,4]$

$\infty \quad 9 + 2$

$\infty > 11.$ ✓

$[3,1] \ngtr D[3,2] + D[2,1]$

$\infty > 5 + 6$

11 ✓

$$D^3 \Rightarrow \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & -4 & 3 \\ 2 & 6 & 0 & 2 & 2 \\ 3 & 11 & 5 & 0 & 7 \\ 4 & 12 & 6 & 1 & 0 \end{array}$$

$D^0(1,2) > D^0(1,3) + D^0(3,2)$

$9 > -4 + 5.$

$9 > 1$ ✓

$D^0(1,4) > D^0(1,3) + D^0(3,4)$

$11 > -4 + 7$

$11 > 3$ ✓

$D^2(2,1) \geq D^0(2,3) + D^0(3,1)$

$6 > 2 + 11$ ×

$D^2(2,4) > D^0(2,3) + D^0(3,4)$

$2 > 2 + 7$ ×

$D^2(4,1) > D^0(4,3) + D^0(3,1)$

$\infty > 1 + 11$ ✓

$D^2(4,2) > D^0(4,3) + D^0(3,2)$

$\infty > 1 + 5$ ✓

$$D^4 \Rightarrow \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & -4 & 3 \\ 6 & 0 & 2 & 2 \\ 11 & 5 & 0 & 7 \\ 12 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

$D^3(1,2) > D^3(1,4) + D^3(4,2)$

$1 > 3 + 6$ ×

$D^3(1,3) > D^3(1,4) + D^3(4,3)$

$-4 > 3 + 1$ ×

$D^3(2,1) > D^3(2,4) + D^3(4,1)$

$6 > 2 + 12$ ×

$D^3(2,3) > D^3(2,4) + D^3(4,3)$

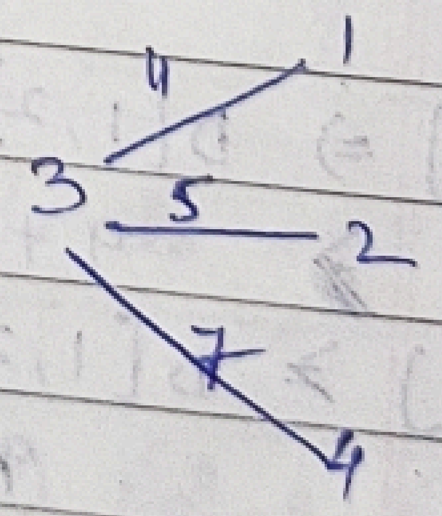
$2 > 2 + 1$ ×

$D^3(3,1) > D^3(3,4) + D^3($

$11 > 7 + 12$ ×

$D^3(3,2) > D^3(3,4) + D^3(4,2)$

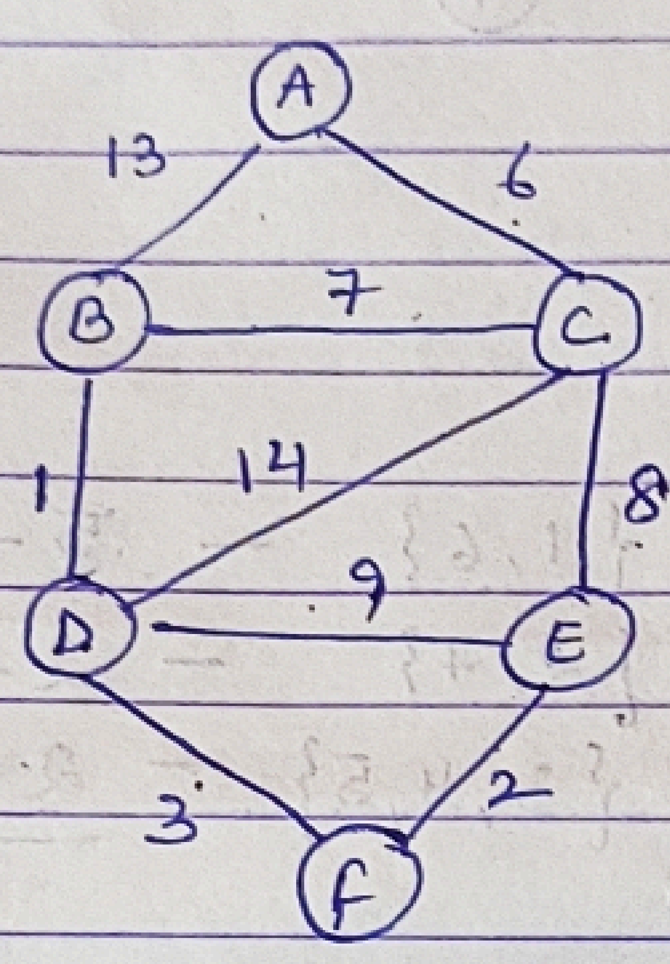
$5 > 7 + 6$ ×

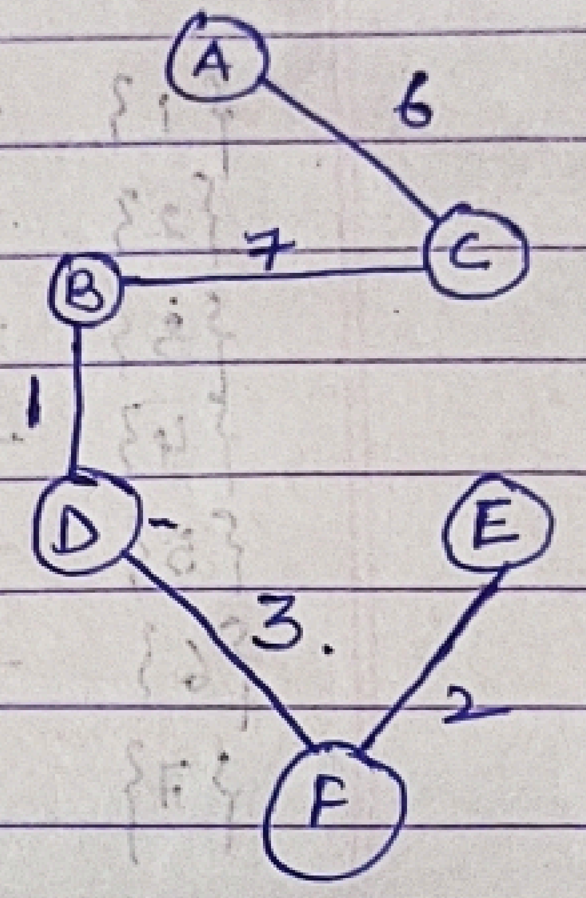


Formula :→

$$D^k[i,j] = \min\{ D^{k-1}(i,j),\ D^{k-1}(i,k) + D^{k-1}(k,j) \}.$$

Sollin's Algorithm :→





A → C → B → D → F → E

Total Cost → 19

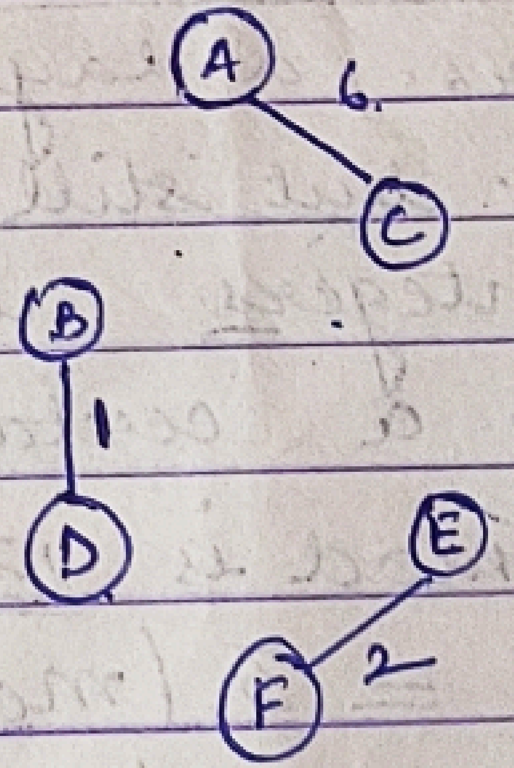
A → A - C.

B → B - D.

C → A - ~~B~~ C

D → D - B.

E → E - F

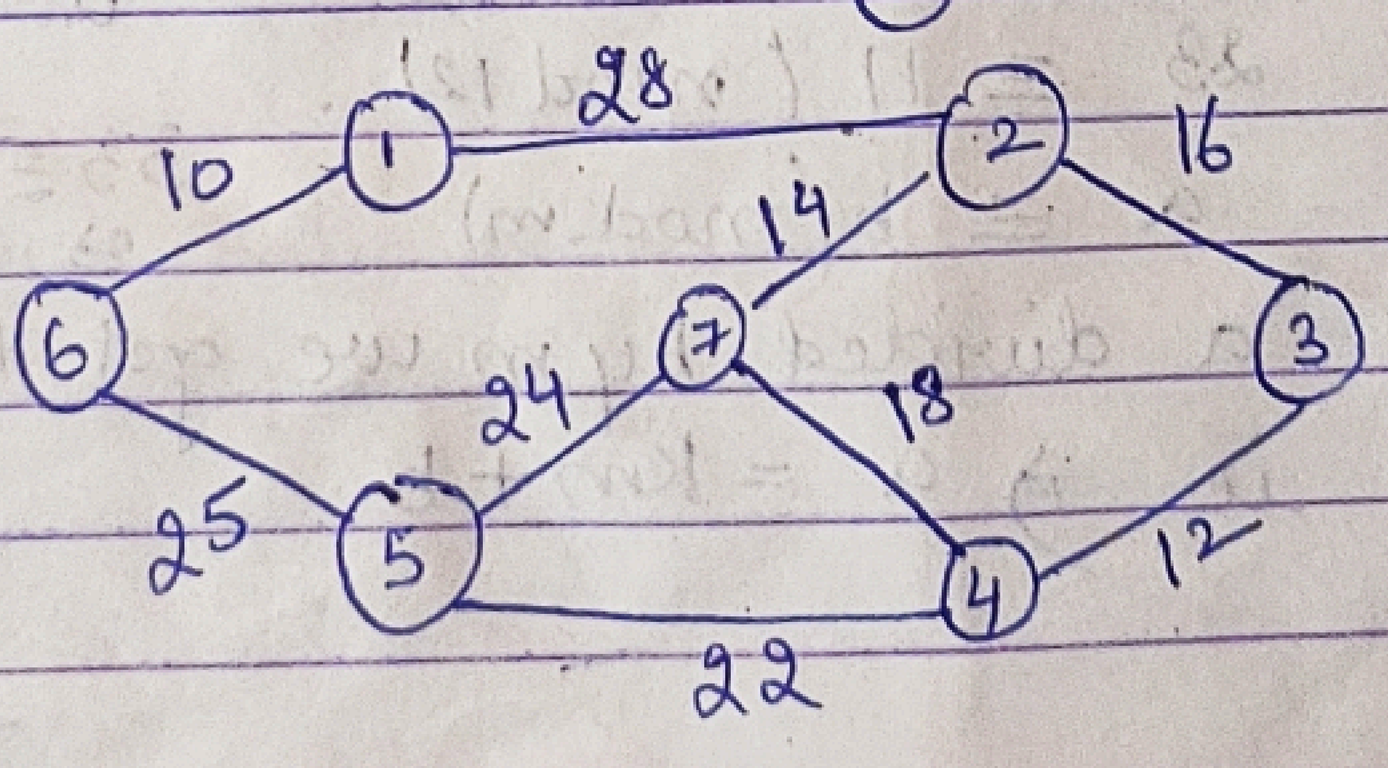


{A-C} → {B-C}

{B-D} → {D-F}

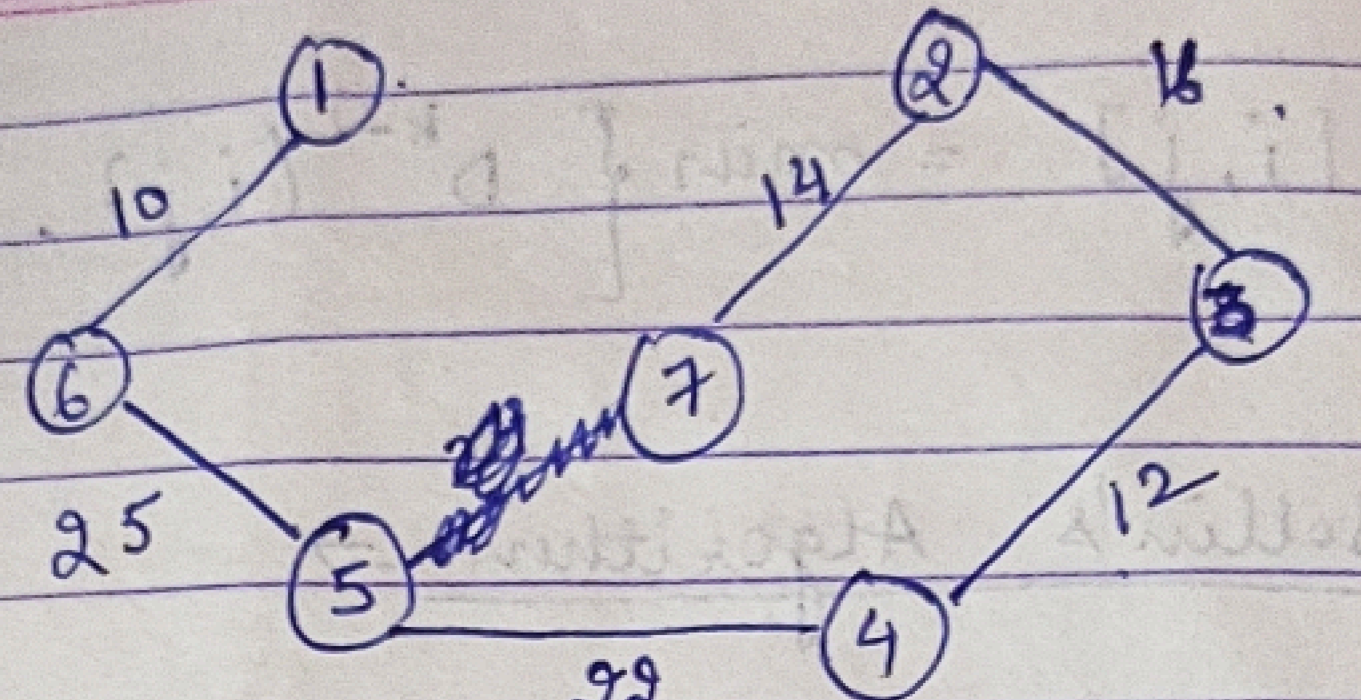
{E-F} → {D-F}

eg :- 2



7 - 3 - 1 = 3

7 - 5 - 1 = 1   E ⇒ 7-4-1 ⇒ 2



$\{1\} \rightarrow 1-6$
$\{2\} \rightarrow 2-7$
$\{3\} \rightarrow 3-4$
$\{4\} \rightarrow 3-4$
$\{5\} \rightarrow 4-5$
$\{6\} \rightarrow 1-6.$
$\{7\} \rightarrow 2-7$

$\{1,6\}$ — $5-6$
$\{2,7\}$ — $2-3$
$\{3,4,5\}$ — $2-3$

Modulo Arithmetic :→

24 hrs. a day but no. 24 hrs. in a clock. but still we understand the time

12 integers. wrapping around after reaching a certain value called modulus.

Here mod is 12   In cryptography.

$15 \equiv 3 \pmod{12}$.   Congruence ($\equiv$) instead of equality ($=$)

$23 \equiv 11 \pmod{12}$.

$a \equiv b \pmod{m}$ $33 \equiv 3 \pmod{10}$ 23, 43. (Remainder is always same)

a divided by m we get the remainder b

ie $\Rightarrow a = km + b$.

	k
m	a
	b

$(2,3) > D^3(2,4) + D^3(4,3)$

$2 > 2 + 1$ X

values we use the formula.

Page No.
Date: / /

Valid or invalid.

$38 \equiv 2 \pmod{12}$ ✓

$38 \equiv 14 \pmod{12}$ ✓

$5 \equiv 0 \pmod 5$ ✓

$10 \equiv 2 \pmod 6$ ✗

$13 \equiv 3 \pmod{13}$ ✗

$M_1 = \frac{M}{m_1}$

$\Rightarrow \frac{105}{3} \Rightarrow 35.$

$M_2 = \frac{105}{5} \Rightarrow 21$

$M_3 \Rightarrow \frac{105}{7} \Rightarrow 15$

$\boxed{M_1 \times M_1^{-1} \Rightarrow 1 \bmod m_1}$

$35 \times M_1^{-1} \Rightarrow 1 \bmod 3$

$35 \times 2 \Rightarrow 1 \bmod 3$

$M_1^{-1} = 2.$

$3 \overline{)35}$

$3 \overline{)70} \, 2$

69

$\overline{1}$

$M_2 \times M_2^{-1} = 1 \bmod m_2$

$21 \times M_2^{-1} = 1 \bmod 5.$

$M_2^{-1} = 1$

$M_3 \times M_3^{-1} = 1 \bmod m_3.$

$15 \times M_3^{-1} = 1 \bmod 7$

$M_3^{-1} = 1$

Chinese Remainder Theorem :→

It is used to solve a set of different congruent equations with one variable but different moduli which are relatively prime as shown below:-

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x \equiv a_n \pmod{m_n}$$

find value of the single variable

It states that the above equations have a unique solⁿ of the moduli are relatively prime.

Formula is $X = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + \dots + a_n M_n M_n^{-1}) \bmod M.$

eg:- 1.

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

$$x = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1}) \bmod M.$$

$a_1 = 2$	$m_1 = 3$	$M_1 =$	M_1^{-1}	M
$a_2 = 3$	$m_2 = 5$	$M_2 =$	M_2^{-1}	$= 105$
$a_3 = 2$	$m_3 = 7$	$M_3 =$	M_3^{-1}	

are prime

$M = m_1 \times m_2 \times m_3 \Rightarrow 3 \times 5 \times 7 = 105.$

2,3) > $\Delta^3(2,4) + \Delta^3(4,3)$

2 > 2 + 1 X

the remaining values we use the formula.
(length of pattern)

Page No.
Date

$$X = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1}) \bmod M$$

$$\Rightarrow (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105$$

$$\Rightarrow 233 \bmod 105$$

$$\boxed{X = 23}$$ ✓

String Matching Algorithms :→

Naive method :→

$n \to$ length [T]　　T – long.

$m \to$ length [P].　} shift will start from 0 to $n-m$.

for $i = 0$ to $n-m$

if (P [0 to m] = T(i to i + m] .　0 to 11

0 0 0 1 0 0 0 1 0 1 0 0 0 1

~~Shift~~ Pattern matches at shift $s = 1$, $s = 5$, $s = 11$

shift 0 -　　0001 — Pattern.

No. of Comparisons (0 to n - m).

0 to 11

T → a b c a a b b c x y z p Q — 15.

P → a b b c → 4

$O(n \times m)$

Rabin Karp Algorithm for string matching

Hash Table

Hash func.

T ⇒ a a a $\boxed{a\ a\ b}$ ✓

P → a a b.

1 + 1 + 2 ⇒ 4 Hash Value

(Addition is my hash func.)

a = 97 ⇒ 1

b = 98 ⇒ 2

x = 24

y = 25

z = 26

a a a ⇒ 1+1+1 = 3.

Spurious hits (false hits where hash value is same but match is not same).

T ⇒ c c a c c a d b a

P ⇒ d b a ⇒ 4+2+1 ⇒ 7.

c c a ⇒ 3+3+1 ⇒ 7 ✗ (SH)

c a c ⇒ 3+1+3 ⇒ 7 ✗ (SH)

multiple SH. easy hash func. so maximum collisions.

To avoid this we take diff hash func or complex hash function.

d ⇒ 4×26^2 b = 2×26^1 a = 1×26^0 = x.

c: 3×26^2 c: 3×26^1 a: 1×26^0 = y.

,3) > $D^3(2,4) + D^3(4,3)$

2 > 2 + 1 ✗

$[x \neq y]$ —

Choosing a correct hash func. may reduce the complexity.

$O(n-m+1)$

RK algo (T, P, d, q)
 $n \leftarrow$ length (T)
 $m \leftarrow$ " (P)
 $h' \leftarrow d^{m-1} \bmod q$.
 $p \leftarrow 0$
 $t_0 \leftarrow 0$

for $i \leftarrow 1$ to m
do $p \leftarrow (\cancel{dp}\ dp + P[i]) \bmod q$
 $t_0 \leftarrow (d\, t_0 + T[i]) \bmod q$
for $s \leftarrow 0$ to $n-m$
 do if $p = t_s$
then if $P[1 \ldots m] = T[s+1 \ldots s+m]$
then " Pattern match occurs with shift "s"

if $s < n-m$
then $t_{s+1} \leftarrow (d(t_s - T[s+1]h)\ T[s+m+1])$
 mod q

Rabin Karp (T, P)

n = T. length.

m = P. length.

h(p) = hash (P)

h(T) = hash (T)

for s = 0 to n-m

if (h(p) = h(T))

if [P(0 - - m-1) = T[s0 - - - s+m-1]

print (" pattern found with shift "s")

if (s < n-m)

h(T) = hash (T(s+1 - - - s+m)]

Boyer Moore Algorithm :->

T = WELCOME TO SURANA COLLEGE 22.

P = SURANA

0 1 2 3 4 5

Here we match the characters of the pattern right to left & then left to right.

We construct a Bad match Table.

Here it works on the idea of a bad match rule & a good match rule.

$n = |T| = 22$.

$m = |P| = 6$.

Bad match Table.

S	U	R	A	N	*
5	4	3	6	1	6

No repeated characters should be there last character is *.

any character other than matching pattern we are going to consider it as *. Value of * is always equal to length of the pattern. Last charactervalue of the pattern is always length of pattern = 6.

2,3) > $D^3(2,4) + D^3(4,3)$

2 > 2 + 1 X

For the remaining values we use the formula.

value = length − index − 1 (length of pattern)

S = 6 − 0 − 1 = 5. (index ⇒ position value of that character)

U = 6 − 1 − 1 = 4

R = 6 − 2 − 1 = 3

N = 6 − 4 − 1 = 1

A = 6 − 5 − 1 = 0

Case [SURANAS]

A = 6 − 3 − 1 = 2.

A = 6 − 5 − 1 = 0.

second value is considered.

Step 1 :→ We match the characters always from right to left.

T = WELCOME TO SURANA COLLEGE.

P = SURANA

mismatch we will search in table.

* ⇒ 6. shift to the right direction from the first character already started matching from 1st character so we will leave this.

~~WELCOME~~ ⇒ WELCOMETOSURANACOLLEGE

~~SURANA~~ SURANA

SURANA ✓

Match each & every character.

P present in T at position 9th.

T ⇒ WELCOME TO SURANA COLLEGE.

P ⇒ COLLEGE

0 1 2 3 4 5 6

COLLEGE COLLEGE

COLLEGE

(n) = T = 22

m = P = 7

C	O	L	E	G
6	5	3	7	1

C ⇒ 7 − 0 − 1 = 6

O ⇒ 7 − 1 − 1 = 5

L ⇒ 7 − 2 − 1 = 4

L ⇒ 7 − 3 − 1 = 3

G = 7 − 5 − 1 = 1

E ⇒ 7 − 4 − 1 ⇒ 2

Present at Position 15

Knuth Morris Pratt Algorithm :→

The algorithm works on proper prefix and proper suffix.

Text backtrack नहीं होता।

abcd } Prefix starts from LHS. {a, ab, abc}.

~~suffix } starts from RHS {d, cd, bcd}~~

$T \to m \quad P \to n \quad O(n+m)$.

Eg :- to find π table / Longest Proper prefix.
(only pattern is considered).

$P_1 \to$

1	2	3	4
a	b	a	b
0	0	1	2

$P_2 \to$

1	2	3	4	5	6	7	8
a	b	c	d	a	b	c	Y
0	0	0	0	1	2	3	0

$P_3 \to$

1	2	3	4	5	6	7	8	9	10
a	b	c	d	a	b	e	a	b	f
0	0	0	0	1	2	0	1	2	0

$P_4 \to$

1	2	3	4	5	6	7	8	9	10	11
a	b	c	d	e	a	b	f	a	b	c
0	0	0	0	0	1	2	0	1	2	3

T :→

i → 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	b	c	a	b	c	a	b	a	b	a	b	d

P :-

j → 0	1	2	3	4	5
	a	b	a	b	d
	0	0	1	2	0

1) Take 2 variables i and j
i = String (T(1)),
j = P[0].

Imp. 2) Compare T(i) with P(j+1)
→ If match is found - (Move both i and j to right)

2) If mismatch (move j to the location as per the π table index).

3) If $j = 0$ move i to the right.

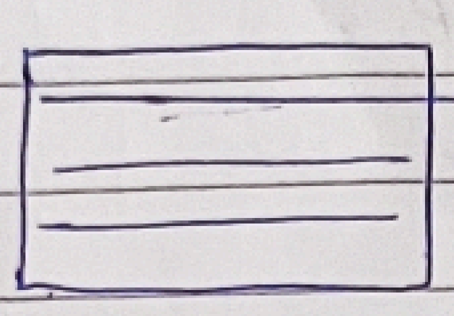
P, NP, NP Hard, NP Complete.

P → Which can be solved in a polynomial time. all sorting & searching algorithms.

NP Class → Which cannot be solved in polynomial time but verified in polynomial time. (Non Deterministic Polynomial)

eg :→ TSP, Knapsack, Subset sum.

(Exponential).



Solving the problem is difficult but verifying is easy.

Hard to solve

Easy to verify

Takes Exponential time.

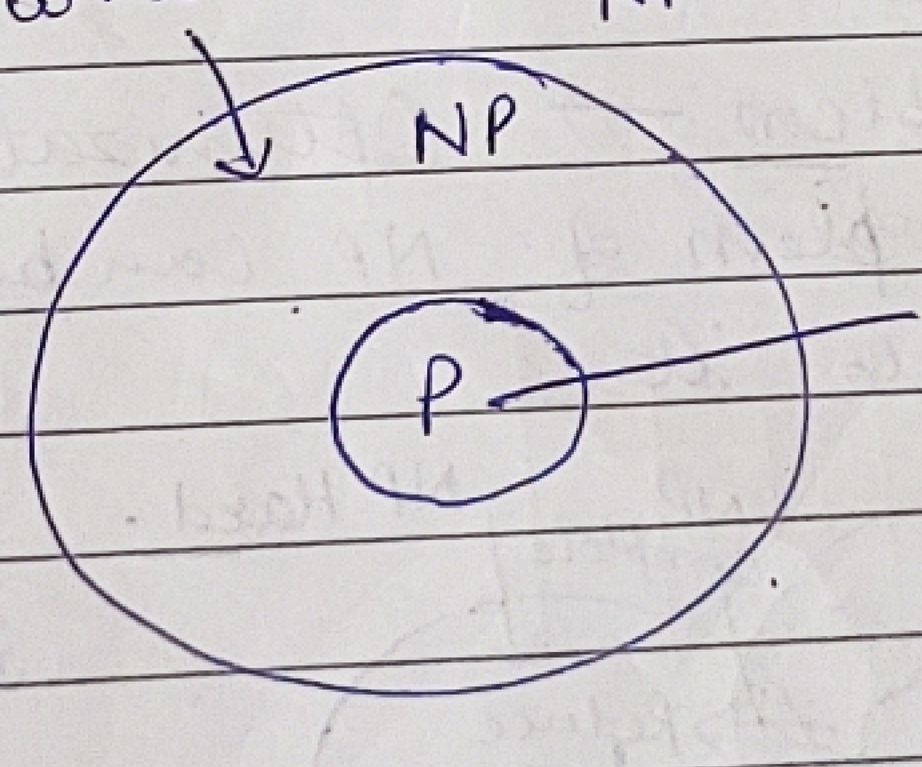
Solⁿ of algo.

Polynomial Time — Non Polynomial Time.

P → Easy to solve, easy to verify & polynomial time.

NP → intractable

which comes under



Those problem which comes under P. Tractable

$P \subseteq NP$.

If $P = NP$? If Yes, then information security / online security is vulnerable to attack. Everything becomes easy such as transportation,

Decision can be converted to optimization problems.
All NP Hard are NP Complete but vice versa not possible.

Scheduling, Understanding DNA etc.
comes under NP - diff. to find a solⁿ but if a solⁿ is there it is easy to verify.
If easy to solve & verify then security is vulnerable to attack.

P ≠ NP. then you can say that there are problems which cannot be solved.

Reduction.

$$A \xrightarrow[\text{Convert}]{\text{Reduce}} B.$$

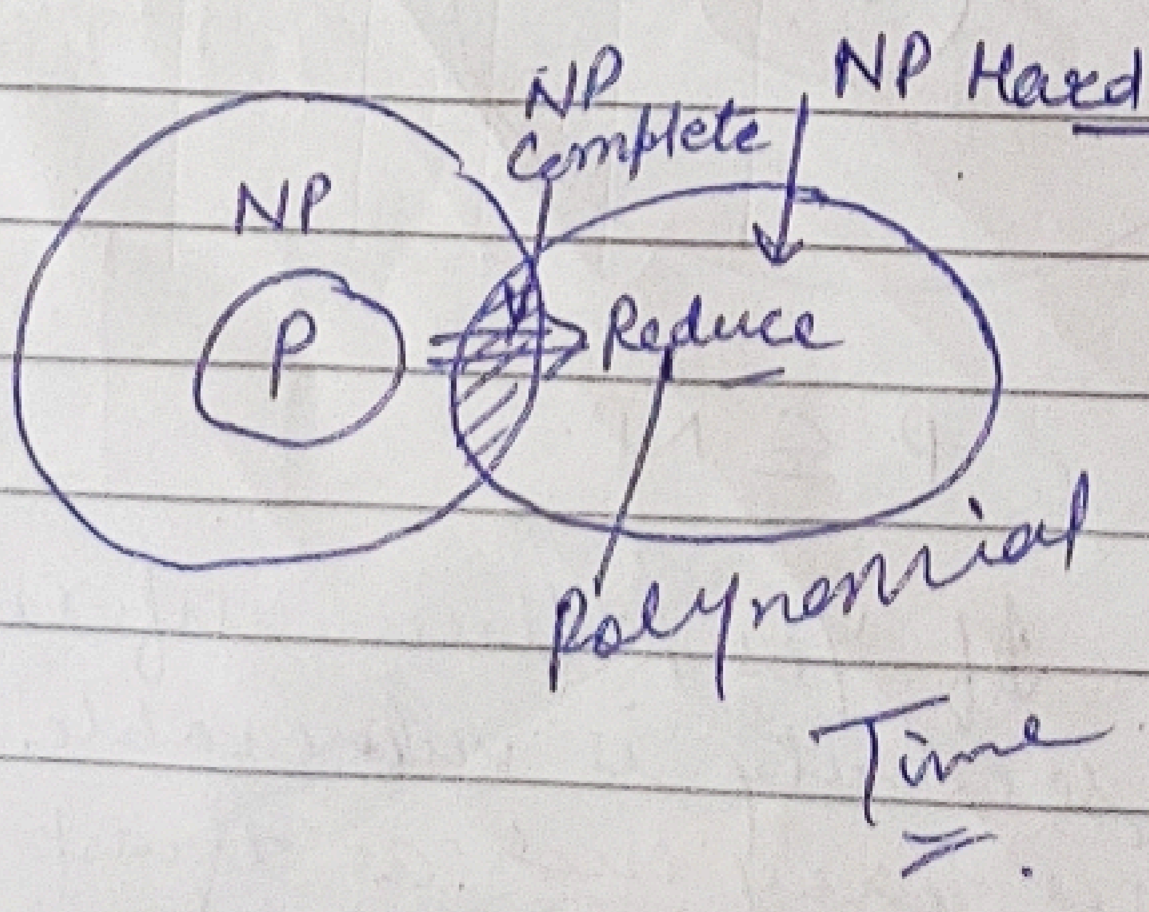
Polynomial time.

A & B are two problems then A reduces to B iff. there is a way to solve A by deterministic algorithm that solve B in polynomial time.

If A is reducible to B → $A \propto B$.

Properties → 1) A is reducible to B & B in P then A in P.
2) A is not in P implies B is not in P.

NP Hard problem → Optimization Problems
If every problem of NP can be polynomial reduced to it.



$\boxed{\text{NP Complete}}$
If it is in NP & NP Hard.
Decision Problems