UNIT - 3

Dynamic programming:

Dynamic programming like the divide & conquer approach, solve problems by combining the sol^{n} to the subproblems.

In divide & conquer algo., partition the problem into independent subproblems, solve the problem recursively and combine their sol^{n}.

In Dynamic programming, it is applicable when the subproblems are dependents i.e. means subproblem have the another sub-problem.

Divide & conquer takes more times because Solving the common subproblems repeatedly.

Dynamic programming algo. solves every subproblem just once and save the answer in table.

Dynamic programming is applied on optimization problem.

Scanned by CamScanner

Optimization problem means the problem which have many solutions & each solution has a unique value and we wish to find out best solution (optimal sol^n.) (max. or min.)

Dynamic programming can be broken into a sequence of four steps :-

1. Characterize the structure of an optimal sol^n.
2. Recursively define the value of an optimal sol^n.
3. Compute the value of an optimal sol^n.
4. Construct the optimal sol^n from the computed information.

Matrix Chain Multiplication

step 1 characterize the structure of an optimal sol^n.

$M(i, j)$

$$\boxed{i \le k < j}$$

step 2 Recursively define the value of an optimal sol^n

$$M(i,j) = \begin{cases} 0 & \text{if } i = j \\ \min\left(M(i,k) + M(k+1, j) + p_{i-1} p_k p_j\right) & \text{if } i < j \end{cases}$$

Scanned by CamScanner

Q $A_{1(5\times4)}$, $A_{2(4\times6)}$, $A_{3(6\times2)}$, $A_{4(2\times7)}$

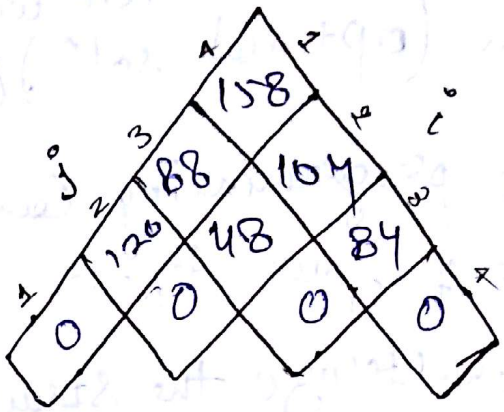
Sol

A_1 P_0 5×4 P_1

A_2 P_1 4×6 P_2

A_3 P_2 6×2 P_3

A_4 P_3 2×7 P_4



$$\underline{M(1,2)}_{\text{for } K=1} = \left[\min\left(M(i,K)+M(K+1,j)+P_{i-1}P_jP_K\right)\right]$$

$$= \min(0+0+120)$$

$M(1,2) = 120$ for $K=1$

$$\underline{M(2,3)}_{\text{for } K=2} = \left[\min(0+0+4\times2\times6)\right]$$

$M(2,3) = 48$ for $K=2$

$$\underline{M(3,4)}_{\text{for } K=3} = \left[\min(0+0+6\times7\times2)\right]$$

$M(2,3) = 84$ for $K=2$

$\underline{M(1,3)}$

for $K=1$ $= (0+48+5\times2\times4)$

$= 88$

for $K=2$ $= (120+0+5\times2\times6)$

Scanned by CamScanner

$M(1,3) = \min(88, 180)$

$\Rightarrow \boxed{M(1,3) = 88}$ for $K = 1$

$M(2,4)$

for $K=2 = (0 + 84 + 4 \times 7 \times 6)$

$= 252$

for $K=3 = (48 + 0 + 4 \times 7 \times 2)$

$= 104$

$M(2,4) = \min(252, 104)$

$\boxed{M(2,4) = 104}$ for $K = 3$

$M(1,4)$

for $K=1 \rightarrow (0 + 104 + 5 \times 7 \times 4)$

$= 244$

for $K=2 \rightarrow (120 + 84 + 5 \times 7 \times 6)$

$= 414$

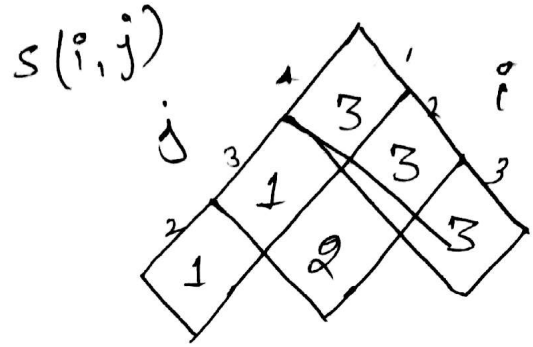
for $K=3 \rightarrow (88 + 0 + 5 \times 7 \times 2)$

$= 158$

$M(1,4) = \min(244, 414, 158)$

$\boxed{M(1,4) = 158}$ for $K = 3$.

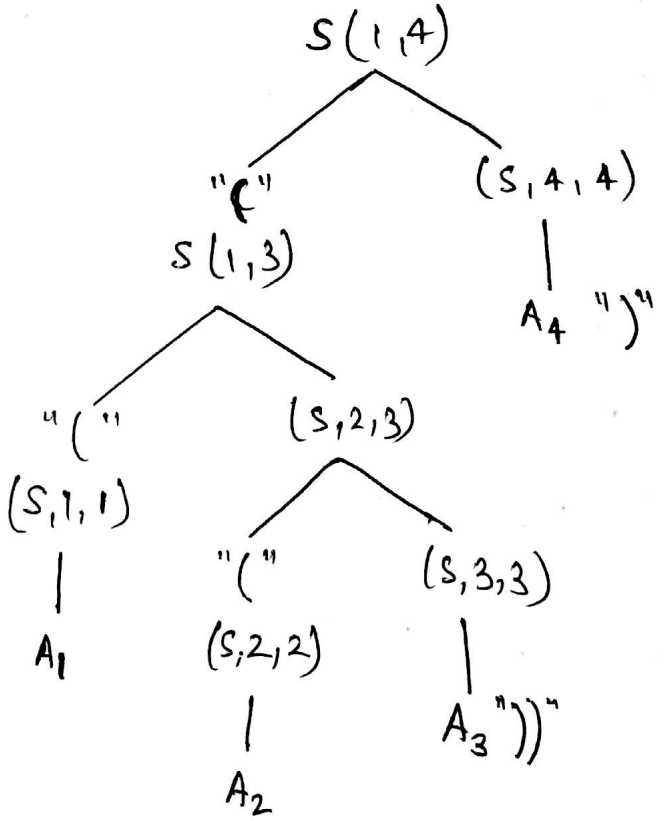
Scanned by CamScanner



④ Construct the optimal sol^{n} (algo.)

print optimal paran (s, i, j)

(i) if i = j
(ii) print A_i
(iii) else
(iv) print "("
(v) print optimal parean (s, i, s(i, j))
(vi) print optimal parean (s, s[i, j]+1, j)
(vii) print ")"



→ ((A_1 (A_2 A_3)) A_4)

Scanned by CamScanner

Q. A_1 30×35 | A_2 35×15 | A_3 15×5 | A_4 5×10 | A_5 10×20 | A_6 20×25

Sol

A_1 P_0 P_1 30×35

A_2 P_1 P_2 35×15

A_3 P_2 P_3 15×5

A_4 P_3 P_4 5×10

A_5 P_4 P_5 10×20

A_6 P_5 P_6 20×25

i \ j	1	2	3	4	5	6
1	0	0	0	0	0	0
2	15750	2625	750	1000	5000	
3	7875	4375	2500	3500		
4	9375	7125	5375			
5	11875	10500				
6	15126					

$M(1,2) = [\min(M(i,k) + M(k+1,j) + P_{i-1} P_j P_k)]$

for $k=1$

$= 0 + 0 + (30 \times 15 \times 35)$

$\boxed{M(1,2) = 15750}$ for $k=1$

$M(2,3) = 0 + 0 + (35 \times 5 \times 15)$

for $k=2$

$\boxed{M(2,3) = 2625}$ for $k=2$

$M(3,4) = 0 + 0 + (15 \times 10 \times 5)$

for $k=3$

$\boxed{M(3,4) = 750}$ for $k=3$

$M(4,5)$

for $k=4$ $= 0 + 0 + (5 \times 20 \times 10)$

$\boxed{M(4,5) = 1000}$ for $k=4$

Scanned by CamScanner

$M(5,6)$

for $K=5$ = $0+0+(10\times25\times20)$

$\boxed{M(5,6) = 5000}$ for $K=5$

$M(1,3)$.

for $K=1$ => $0+2625+(30\times5\times35)$

$M[1,3] = 7875$ for $K=1$

for $K=2$ => $15750+0+(30\times5\times15)$

$= 18000$ for $K=2$.

$M(1,3) = \text{Min}(7875, 18000)$

$\boxed{M(1,3) = 7875}$ for $K=1$

$M(2,4)$

for $K=2$ => $0+750+(35\times10\times15)$

$= 6000$ for $K=2$.

for $K=3$ => $2625+0+(35\times10\times5)$

$= 4375$ for $K=3$.

$M(2,4) = \text{Min}(6000, 4375)$

$\boxed{M(2,4) = 4375}$ for $K=3$.

Scanned by CamScanner

$M(3,5)$

for $K=3 \Rightarrow 0+1000+(15\times5\times20)$

$= 2500$

for $K=4 \Rightarrow 750+0+(15\times10\times20)$

$= 3750$

$M(3,5) = \min(2500, 3750)$

$\boxed{M(3,5) = 2500}$ for $K=3$

$M(4,6)$

for $K=4 \Rightarrow 0+5000+(5\times10\times25)$

$= 6250$

for $K=5 \Rightarrow 1000+0+(5\times20\times25)$

$= 3500$

$M(4,6) = \min(6250, 3500)$

$\boxed{M(4,6) = 3500}$ for $K=5$

$M(1,4)$

for $K=1 \Rightarrow 0+4375+(30\times10\times35)$

$= 14875$

for $K=2 \Rightarrow 15750+750+(30\times10\times15)$

$= 21000$

for $K=3 \Rightarrow 7875+0+(30\times10\times5)$

$= 9375$

$M(1,4) = \min(14875, 21000, 9375)$

$\boxed{M(1,4) = 9375}$ for $K=3$

Scanned by CamScanner

M(2,5)

for k=2 ⇒ 0+2500+(35x20x15)

= 13000

for k=3 ⇒ 2625+1000+(35x20x5)

= 7125

for k=4 ⇒ 4375+0+(35x20x10)

= 11375

M(2,5) = Min(13000, 7125, 11375)

M(2,5) = 7125 for k=3.

M(3,6)

for k=3 ⇒ 0+3500+(15x25x5)

= 5375

for k=4 ⇒ 750+5000+(15x25x10)

= 9500

for k=5 ⇒ 2500+0+(15x25x20)

= 10000

M(3,6) = Min(5375, 9500, 10000)

⇒ **M(3,6) = 5375** for k=3

M(1,5)

for k=1 ⇒ 0+7125+(30x20x35)

= 20125

for k=2 ⇒ 15750+2500+(30x20x15)

= 27250

Scanned by CamScanner

for K=3 ⇒ 7875 + 1000 + (30x20x5)

= 11875

for K=4 ⇒ 9375 + 0 + (30x20x10)

= 15375

M(1,5) = Min (28125, 27250, 11875, 15375)

M(1,5) = 11875 for K=3

M(2,6)

for K=2 ⇒ 0 + 5375 + (35x25x15)

= 18500

for K=3 ⇒ 2625 + 3500 + (35x25x5)

= 10500

for K=4 ⇒ 4375 + 5000 + (35x25x10)

= 18125

for K=5 ⇒ 4125 + 0 + (35x25x20)

= 24625

M(2,6) = Min (18500, 10500, 18125, 24625)

M(2,6) = 10500 for K=3

M(1,6)

for K=1 ⇒ 0 + 10500 + (30x25x35)

= 36750

for K=2 ⇒ 15750 + 5375 + (30x25x15)

= 32375

Scanned by CamScanner

for K=3 ⇒ 7875+3500+(30×25×5)

= 15125

for K=4 ⇒ 9375+5000+(30×25×10)

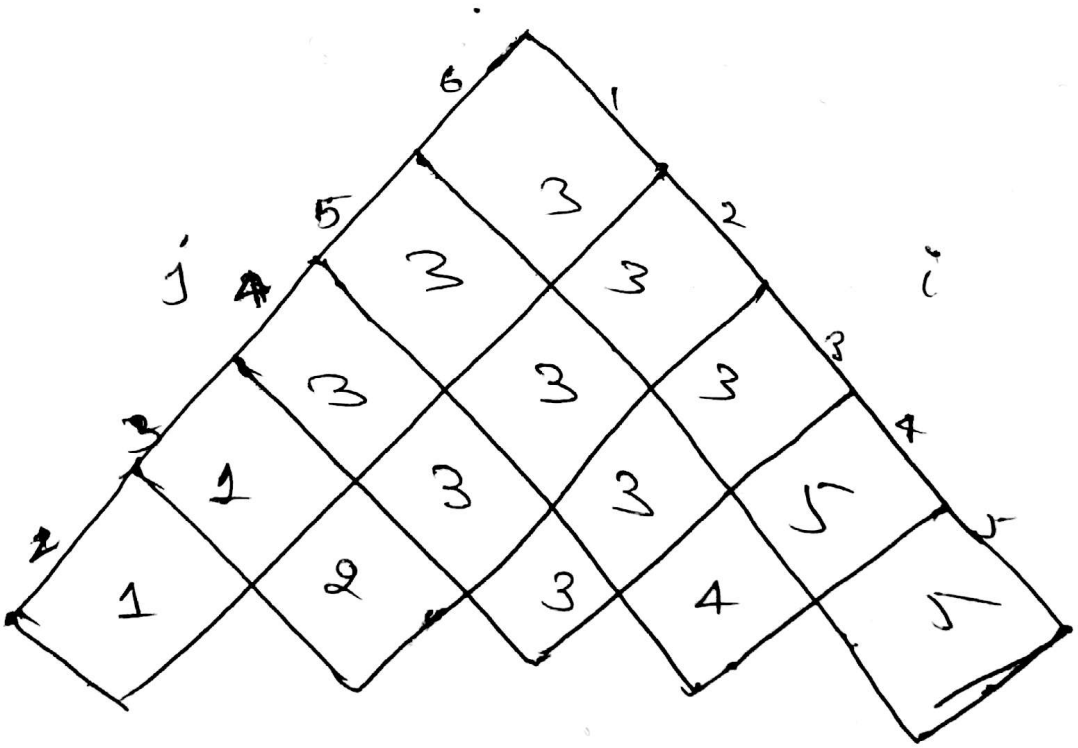
= 21875

for K=5 ⇒ 11875+0+(30×25×20)

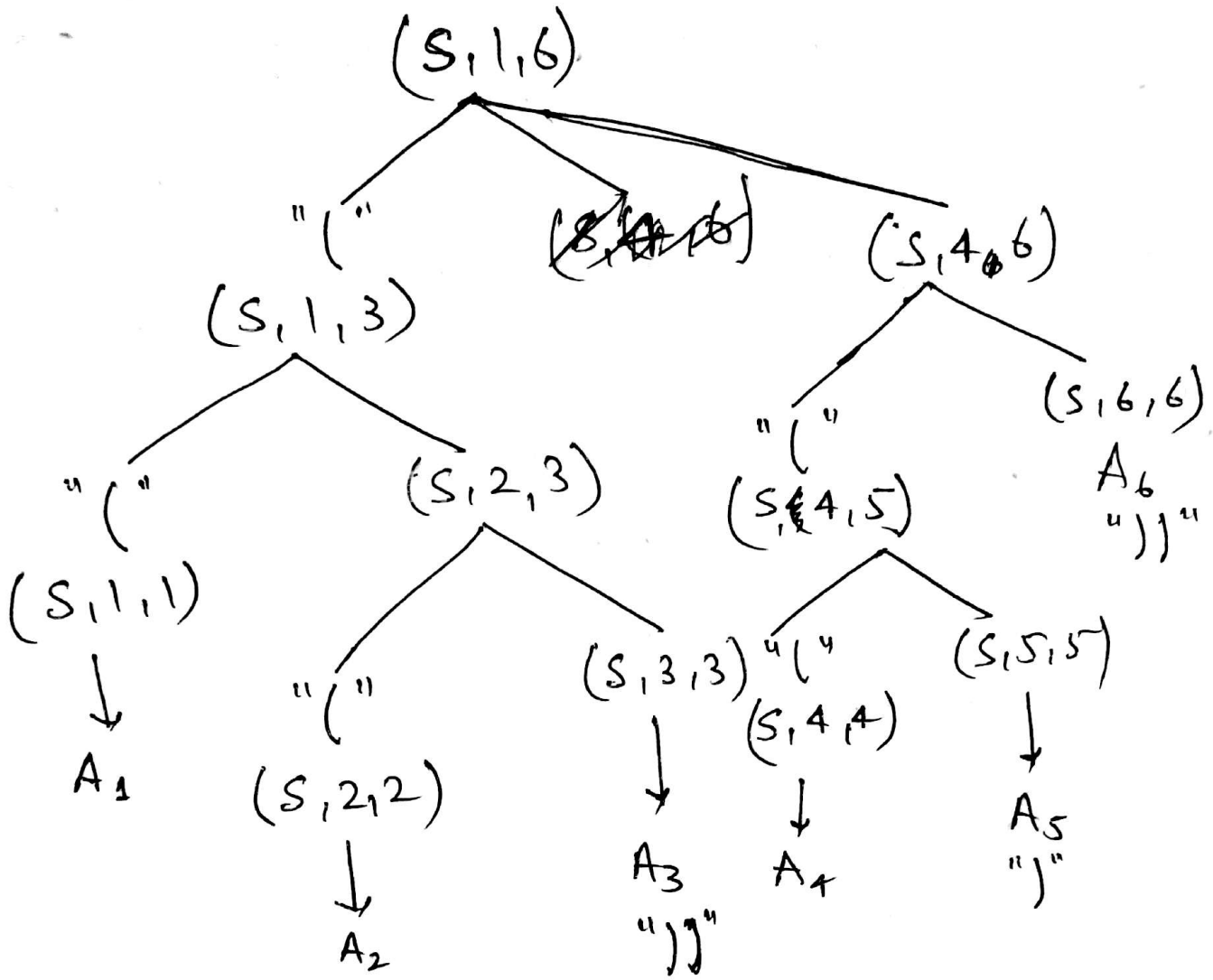
= 26875

M(1,6) = Min(36700, 32375, 15125, 21875,

M(1,6) = 15125 for K=3



Scanned by CamScanner



$((A_1(A_2A_3))((A_4A_5)A_6))$

Scanned by CamScanner

Longest Common Subsequence (LCS)

Longest common subsequence is used in DNA testing and based on dynamic programming.

Step 1) **Characterize the structure:**

Suppose

$X = \{B, C, D\}$

$Y = \{C, B, D\}$

where $Z = \{B, D\}$ or $Z = \{C, D\}$

Step 2) **Recursively define the value of an optimal sol^n**

$$c(i,j) = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c(i-1, j-1) + 1 & \text{if } i,j > 0 \\ & x_i = y_j \\ \max\left(c(i-1,j), c(i,j-1)\right) & x_i \neq y_j \end{cases}$$

Step 3) **LCS-LENGTH (X, Y)**

```
① m ← length[X]
② n ← length[Y]
③ for i ← 1 to m
④     do c[i,0] ← 0
⑤ for j ← 0 to n
⑥     do c[0,j] ← 0
⑦ for i ← 1 to m
⑧     do for (j ← 1 to n)
⑨     do if Xi = Yj
⑩         then c[i,j] ← c[i-1, j-1] + 1
⑪         b[i,j] ← "↖"
```

Scanned by CamScanner

(12) else if $c[i-1, j] \geq c[i, j-1]$

(13) then $c[i,j] \leftarrow c[i-1,j]$

(14) $b[i,j] \leftarrow$ "↑"

(15) else $c[i,j] \leftarrow c[i,j-1]$

(16) $b[i,j] \leftarrow$ "←"

(17) return c and b.

Q

$X = B, A, C, D, B$

$Y = B, D, C, B$

Sol

i \ j		0	B	D	C	B
	0	0	0	0	0	0
✓	B	0	1 ↖	1 ←	1 ←	1 ↖
	A	0	1 ↑	1 ↑	1 ↑	1 ↑
✓	C	0	1 ↑	1 ↑	2 ↖	2 ←
	D	0	1 ↑	2 ↖	2 ↑	2 ↑
✓	B	0	1 ↖	2 ↑	2 ↑	3 ↖

LCS = B, C, B

Scanned by CamScanner

Q. X = A, B, C, B, D, A, B

Y = B, D, C, A, B, A

Sol

① m ← 7

② n ← 6

③ for i = 1 to 7

④ c[1,0] ← 0

④ c[2,0] ← 0

④ c[3,0] ← 0

④ c[4,0] ← 0

④ c[5,0] ← 0

④ c[6,0] ← 0

④ c[7,0] ← 0

⑤ for j = 0 to 6

⑥ c[0,0] ← 0

⑥ c[0,1] ← 0

⑥ c[0,2] ← 0

⑥ c[0,3] ← 0

⑥ c[0,4] ← 0

⑥ c[0,5] ← 0

⑥ c[0,6] ← 0

↑i↓ \ j→	0	1 B	2	3 C	4 A	5 B	6 A
0	0	0	0	0	0	0	0
1 A	0	0 ↑	0 ↑	0 ↑	1 ↖	1 ←	1 ↖
2 B ✓	0	1 ↖	1 ←	1 ←	1 ↑	2 ↖	2 ←
3 C ✓	0	1 ↑	1 ↑	2 ↖	2 ←	2 ↑	2 ↑
4 B ✓	0	1 ↖	1 ↑	2 ↑	2 ↑	3 ↖	3 ←
5 D	0	1 ↑	2 ↖	2 ↑	2 ↑	3 ↑	3 ↑
6 A ✓	0	1 ↑	2 ↑	2 ↑	3 ↖	3 ↑	4 ↖
7 B	0	1 ↖	2 ↑	2 ↑	3 ↑	4 ↖	4 ↑

Scanned by CamScanner

(7) for $i \leftarrow 1$ to 7

(8) for $j \leftarrow 1$ to 6

(9) $x_1 = y_1$ $(A \neq B)$ false.

(12) $c[0,1] \geq c[1,0]$ true $(0=0)$

(13) $c[1,1] \leftarrow c[0,1]$

(14) $b[1,1] \leftarrow$ "↑"

(8) for $j \leftarrow 2$

(9) $x_1 = y_2$ $(A \neq D)$ false

(12) $c[0,2] \geq c[1,1]$ $(0=0)$ true

(13) $c[1,2] \leftarrow c[0,2]$

(14) $b[1,2] \leftarrow$ "↑"

(8) for $j \leftarrow 3$

(9) $x_1 = y_3$ $(A \neq C)$ false

(12) $c[0,3] \geq c[1,2]$

(13) $c[1,3] \leftarrow c[0,3]$

(14) $b[1,3] \leftarrow$ "↑"

(8) for $j \leftarrow 4$

(9) $x_1 = y_4$ $(A = A)$ true

(10) $c[1,4] \leftarrow c[0,3] + 1$ $(0+1=1)$

(11) $b[i,j] \leftarrow$ "↖"

(8) for $j = 5$

(9) $x_1 = y_5$ $(A \neq B)$ false

(12) $c[0,5] \geq c[1,4]$ $(0 \geq 5)$ false

(15) $c[1,5] \leftarrow c[1,4]$

(16) $b[i,j] \leftarrow$ "←"

(8) $j = 6$

(9) $x_1 = y_6$ $(A = A)$ true

(10) $c[1,6] \leftarrow c[0,5] + 1$ $(0+1)=1$

(11) $b[i,j] \leftarrow$ "↖"

Scanned by CamScanner

(7) for i=2

(8) for j=1

c[2,1] ← 1

b[2,1] ← "↖"

(8) for j=2

c[2,2] ← 1

b[2,2] ← "←"

(8) for j=3

c[2,3] ← 1

b[2,3] ← "←"

(8) for j=4

c[2,4] ← 1

b[2,4] ← "↑"

(8) for j=5

c[2,5] ← 2

b[2,5] ← "↖"

(8) for j=6

c[2,6] ← 2

b[2,6] ← "←"

(7) for i=3

(8) for j=1

c[3,1] ← 1 , b[3,1] ← "↑"

(8) for j=2

c[3,2] ← 1 , b[3,2] ← "↑"

(8) for j=3

c[3,3] ← 2 , b[3,3] ← "↖"

(8) for j=4

c[3,4] ← 2 , b[3,4] ← "←"

(8) for j=5

c[3,5] ← 2 , b[3,5] ← "↑"

(8) for j=6

c[3,6] ← 2 , b[3,6] ← "↑"

Scanned by CamScanner

(7) for i = 4

(8) for j = 1

c[4,1] ← 1 , b[4,1] ← "↖"

(8) for j = 2

c[4,2] ← 1 , b[4,2] ← "↑"

(8) for j = 3

c[4,3] ← 2 , b[4,3] ← "↑"

(8) for j = 4

c[4,4] ← 2 , b[4,4] ← "↑"

(8) for j = 5

c[4,5] ← 3 , b[4,5] ← "↖"

(8) for j = 6

c[4,6] ← 3 , b[4,6] ← "←"

(7) for i = 5

(8) for j = 1

c[5,1] ← 1 b[5,1] ← "↑"

(8) for j = 2

c[5,2] ← 2 b[5,2] ← "↖"

(8) for j = 3

c[5,3] ← 2 b[5,3] ← "↑"

(8) for j = 4

c[5,4] ← 2 b[5,4] ← "↑"

(8) for j = 5

c[5,5] ← 3 b[5,5] ← "↑"

(8) for j = 6

c[5,6] ← 3 b[5,6] ← "↑"

Scanned by CamScanner

(7) for $i = 6$

(8) for $j = 1$
$c[6,1] \leftarrow 1$ $\quad b[6,1] \leftarrow$ "↑"

(8) for $j = 2$
$c[6,2] \leftarrow 2$ $\quad b[6,2] \leftarrow$ "↑"

(8) for $j = 3$
$c[6,3] \leftarrow 2$ $\quad b[6,3] \leftarrow$ "↑"

(8) for $j = 4$
$c[6,4] \leftarrow 3$ $\quad b[6,4] \leftarrow$ "↖"

(8) for $j = 5$
$c[6,5] \leftarrow 3$ $\quad b[6,5] \leftarrow$ "↑"

(8) for $j = 6$
$c[6,6] \leftarrow 4$ $\quad b[6,6] \leftarrow$ "↖"

(7) for $i = 7$.

(8) for $j = 1$
$c[7,1] \leftarrow 1$ $\quad b[7,1] \leftarrow$ "↖"

(8) for $j = 2$
$c[7,2] \leftarrow 2$ $\quad b[7,2] \leftarrow$ "↑"

(8) for $j = 3$
$c[7,3] \leftarrow 2$ $\quad b[7,3] \leftarrow$ "↑"

(8) for $j = 4$
$c[7,4] \leftarrow 3$ $\quad b[7,4] \leftarrow$ "↑"

(8) for $j = 5$
$c[7,5] \leftarrow 4$ $\quad b[7,5] \leftarrow$ "↖"

(8) for $j = 6$
$c[7,6] \leftarrow 4$ $\quad b[7,6] \leftarrow$ "↑"

LCS = B, C, B, A

Scanned by CamScanner

GRAPH

1. MST (Minimum Spanning Tree) — Kruskal, Prim's
2. Single Source shortest Path — Dijkstra, Bellman ford
3. All pair shortest path — Floyd warshall algo.

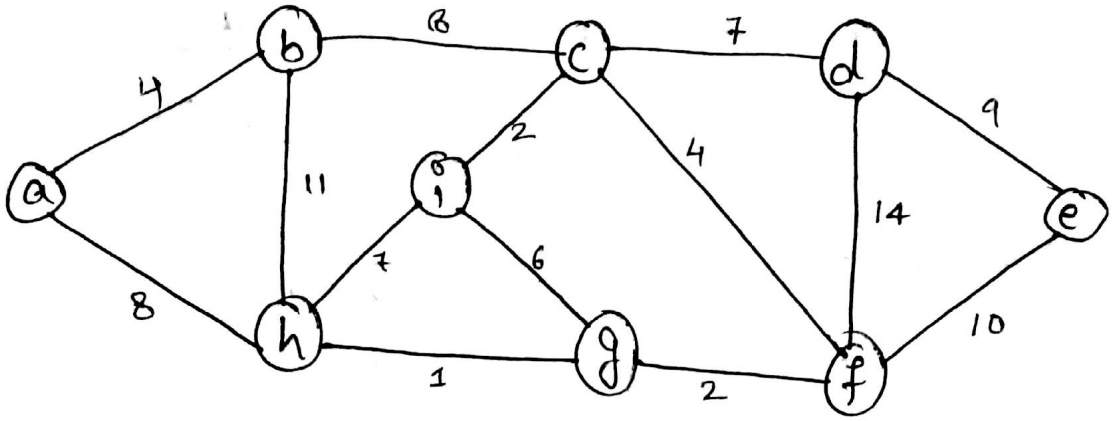
Greedy :

Dynamic	Greedy
1) It gives the optimal sol^n	1) It does not give the guarantee of best sol^n.
2) Sol^n is store in form of table for again use	2) It does not store sol^n.
3) Time Consuming	3) It takes less time as compared to dynamic
4) Dynamic programming is used for small quantity	4) It is used for large quantity.
5) It check for all sol^n	5) It checks at a time best sol^n

Scanned by CamScanner

KRUSKAL :

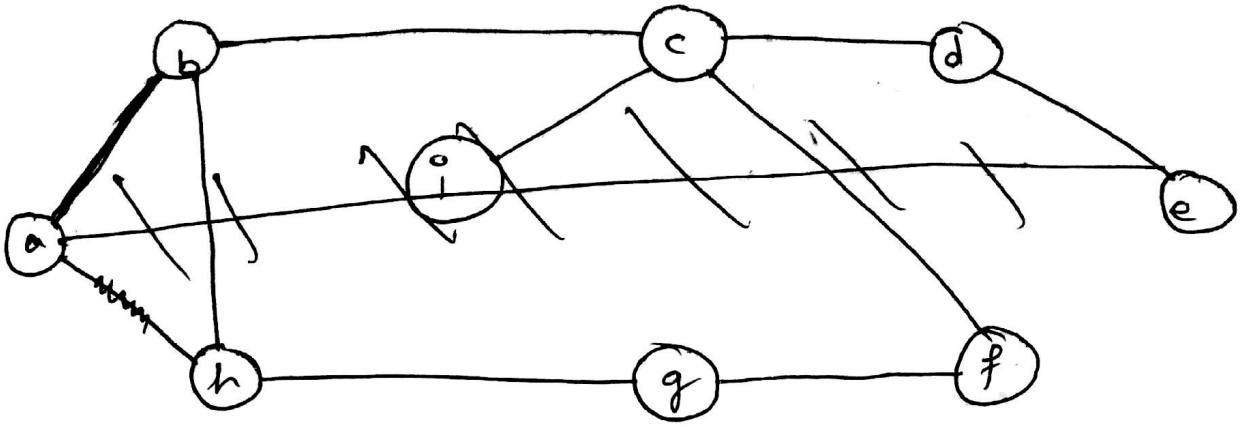
Q

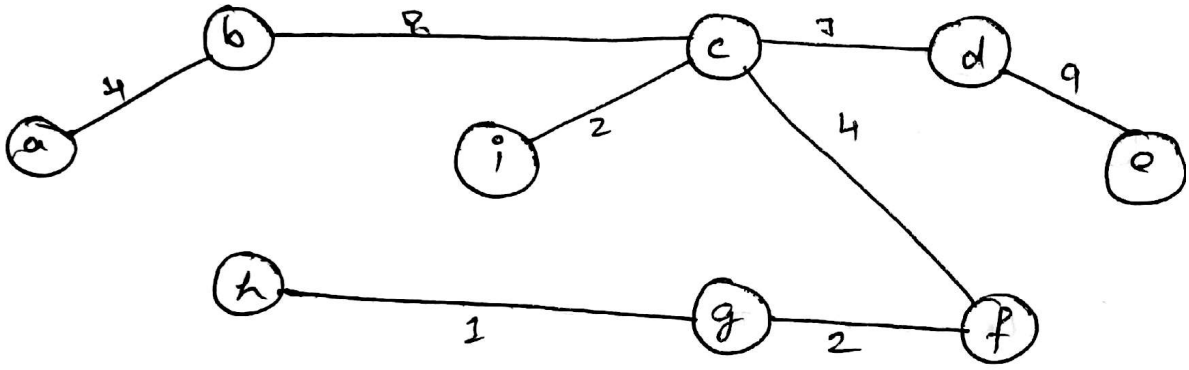


Sol Sort the edge in increasing order

(h, g) → 1
(c, i) → 2
(g, f) → 2
(a, b) → 4
(c, f) → 4
(i, g) → 6
(c, d) → 7
(h, i) → 7
(b, c) → 8
(a, h) → 8
(d, e) → 9
(e, f) → 10
(b, h) → 11
(d, f) → 14

Scanned by CamScanner





Algorithm :

MST Kruskal (G, w)

① A ← ϕ
② for each vertex v ∈ V(G)
③ do make set (v)
④ Sort the edge into non-decreasing order
⑤ For each edge (u, v) ∈ E takes in increasingly order not form the cycle.
⑥ do if find set (u) ≠ find set (v)
⑦ A ← A ∪ (u, v)
⑧ Union (u, v)
⑨ Return A

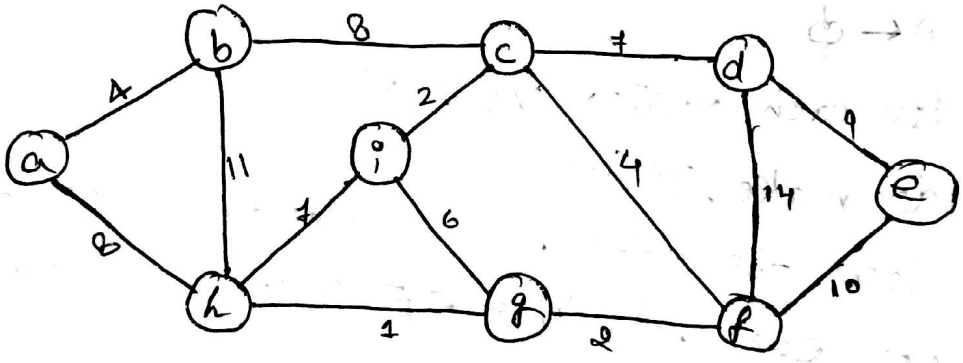
Scanned by CamScanner

PRIM'S Algorithm :

MST(G, w, r)

① for each $u \leftarrow V(G)$
② $Key(u) \leftarrow \infty$
③ $\pi(u) \leftarrow NIL$
④ $Key(r) \leftarrow 0$
⑤ $Q \leftarrow V(G)$
⑥ while $Q \neq \phi$
⑦ $u \leftarrow$ extract min(Q)
⑧ for each $v \in Adj(u)$
⑨ if $v \in Q$ and $w(u,v) < Key(v)$
⑩ $\pi(v) \leftarrow u$
⑪ $Key(v) \leftarrow w(u,v)$

Q



Sol Take an arbitrary vertex

Key	0	∞	∞	∞	∞	∞	∞	∞	∞
vertex	a	b	c	d	e	f	g	h	i
Parent (π)		a	b	c	d	c	f	i	c

Scanned by CamScanner

(5) $Q = \{a, b, c, d, e, f, g, h, i\}$

(6) while $Q \neq \phi$ (true)

(7) $u \leftarrow \{a\}$

(8) for $v \in adj(u) \Rightarrow v \leftarrow \{b, h\}$

(9) if($b \in Q$ and $w(a,b) < key(b)$) true
$(4 < \infty)$

(10) $\pi(b) \leftarrow a$

(11) $key(v) \leftarrow w(a,b)$ ie, 4.

(9) if ($h \in Q$ and $w(a,h) < key(h)$) true

(10) $\pi(h) \leftarrow a$

(11) $key(h) \leftarrow w(a,h)$ ie., 8.

(6) ~~while~~ $Q = \{a, b, c, d, e, f, g, h, i\}$

(6) while $Q \neq \phi$ (true)

(7) $u \leftarrow \{a, b\}$

(8) for $v \in adj(u) \Rightarrow v \leftarrow \{a, h, c\}$

(9) if ($a \in Q$) false.

(9) if($h \in Q$ and $w(b,h) < key(h)$) false

(9) if($c \in Q$ and $w(b,c) < key(c)$) true.

(10) $\pi(c) \leftarrow b$

(11) $key(c) \leftarrow w(b,c)$ ie., 8.

(6) while $Q \neq \phi$ (true)

(7) $u \leftarrow \{c\}$

$Q = \{a, b, d, e, f, g, h, i\}$

(8) for $v \in adj(u) \Rightarrow v \leftarrow \{d, f, i, b\}$

Scanned by CamScanner

~~(9) if (d ∈ Q)~~ true

(9) if (d ∈ Q & w(d,c) < Key(d)) (true)

(10) π(d) ← c

(11) Key(d) ← w(d,c) i.e., 7

~~(8) while Q ≠ ϕ (true)~~

(9) if (f ∈ Q & w(f,c) < Key(f)) (true)

(10) π(f) ← c

(11) Key(f) ← 4

(9) if (i ∈ Q & w(i,c) < Key(i)) (true)

(10) π(i) ← c

(11) Key(i) ← 2

(9) if (b ∈ Q) false

(6) while Q ≠ ϕ (true)

(7) u ← {i}

Q = {a, b, f, d, e, f, g, h, i}

(8) for v ∈ adj(i) v ← {c, g, h}

(9) if (c ∈ Q) false.

(9) if (g ∈ Q & w(g,i) < Key(g))

(10) π(g) ← i

(11) Key(g) ← 6

(9) if (h ∈ Q & w(h,i) < Key(h))

(10) π(h) ← i

(11) Key(h) ← w(h,i) i.e., 7

(6) while Q ≠ ϕ (true)

(7) u ← {f}

Q = {a, b, f, d, e, f, g, h, i}

(8) for v ∈ adj(f) v ← {g, c, d, e}

Scanned by CamScanner

⑨ if ($g \in Q$ & $w(g,f) < Key(g)$) (true)

⑩ $\pi(g) \leftarrow f$

⑪ $Key(g) \leftarrow w(g,f)$; key 2

⑨ if ($c \in Q$) (false)

⑨ if ($d \in Q$ & $w(d,f) < Key(d)$) (false)

⑨ if ($e \in Q$) & $w(e,f) < Key(e)$)

⑩ $\pi(e) \leftarrow f$

⑪ $Key(e) \leftarrow 10$

⑥ while $Q \neq \phi$ (true)

⑦ $u \leftarrow \{g\}$

$Q = \{a, b, c, d, e, f, g, h, i\}$

⑧ for $v \in adj(g)$ $\Rightarrow v \leftarrow h, i, f$

⑨ if ($h \in Q$ & $w(h,g) < Key(h)$)

⑩ $\pi(h) \leftarrow g$

⑪ $Key(h) \leftarrow 1$

⑨ if $i \in Q$ (false)

⑨ if $f \in Q$ (false)

⑥ while $Q \neq \phi$ (true)

⑦ $u \leftarrow \{h\}$

$Q = \{a, b, c, d, e, f, g, h, i\}$

⑧ for $v \in adj(h)$ $v \leftarrow \{a, b, i, g\}$

⑨ if ($a \in Q$) false

⑨ if ($b \in Q$) false.

⑨ if ($i \in Q$) false.

⑨ if ($g \in Q$) false.

⑥ while $Q \neq \phi$ (true

Scanned by CamScanner

(7) $u \leftarrow \{d\}$

$Q = \{a, b, c, d, e, f, g, h, i\}$

(8) for $v \in adj(d)$ $v \leftarrow \{c, f, e\}$

(9) if $(c \in Q)$ false

(9) if $(f \in Q)$ false.

(9) if $(e \in Q \ \& \ w(e,d) < Key(e))$

(10) $\pi(e) \leftarrow d$

(11) $Key(e) \leftarrow w(e,d)$ i.e. 9

(6) while $Q \neq \phi$ (true)

(7) $u \leftarrow \{e\}$

$Q = \{a, b, c, d, f, f, g, h, i\} = \phi$

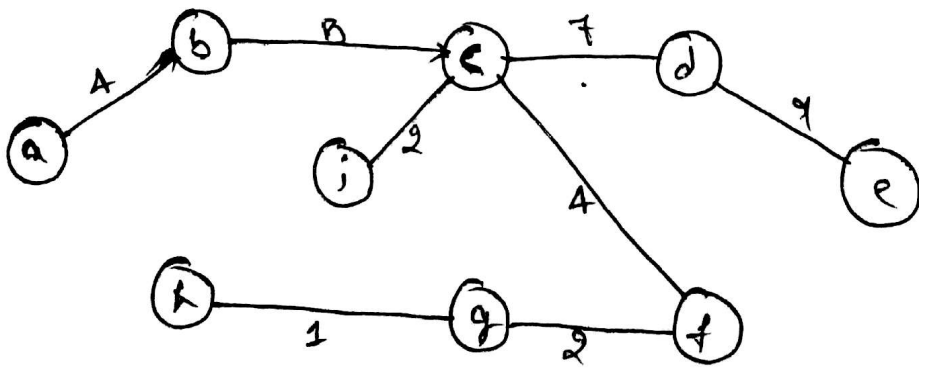
(8) for $v \in adj(e)$ $v \leftarrow \{d, f\}$

(9) if $(d \in Q)$ false

(9) if $(f \in Q)$ false.

(6) while $Q \neq \phi$ (false

Key	0	4	8	7	9	4	2	1	2
vertex	a	b	c	d	e	f	g	h	i
Parent		a	b	c	d	c	f	g	c



Scanned by CamScanner

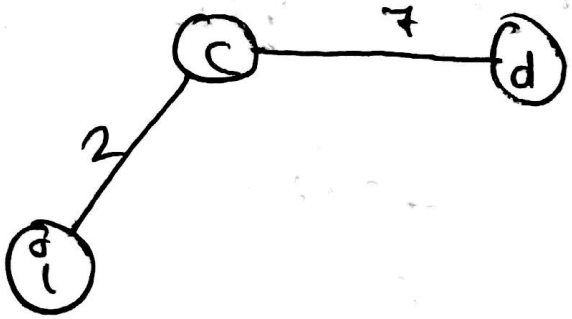
Method - II :

$Q = \{a, b, c, d, e, f, g, h, i\}$

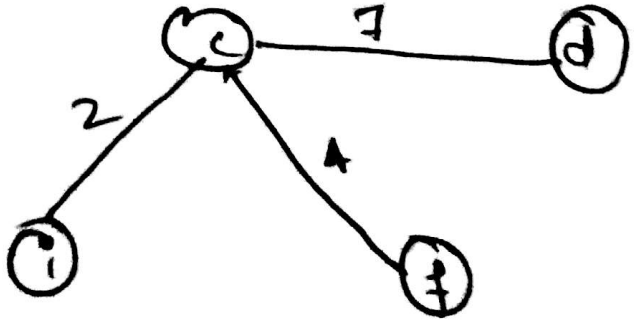
$(d)(a, b, c, e, f, g, h, i)$

* $(d, c)(a, b, e, f, g, h, i)$

* $(d, c, i)(a, b, e, f, g, h)$

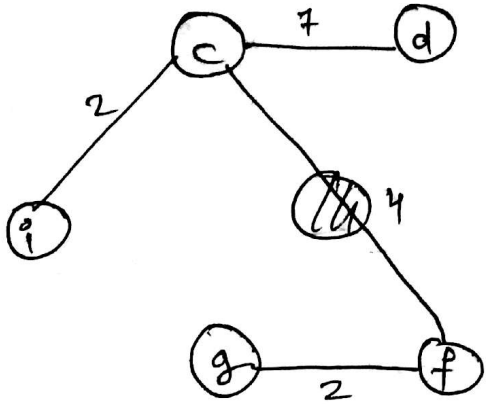


* $(d, c, i, f)(a, b, e, g, h)$

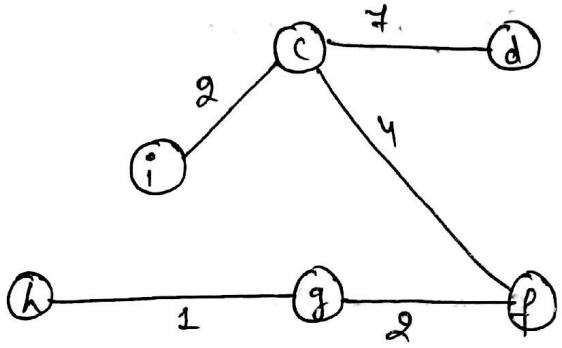


Scanned by CamScanner

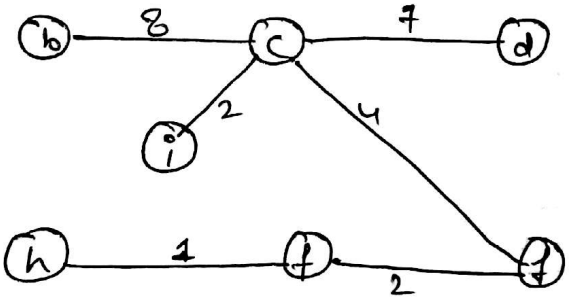
* (d, c, i, f, g) (a, b, e, h)



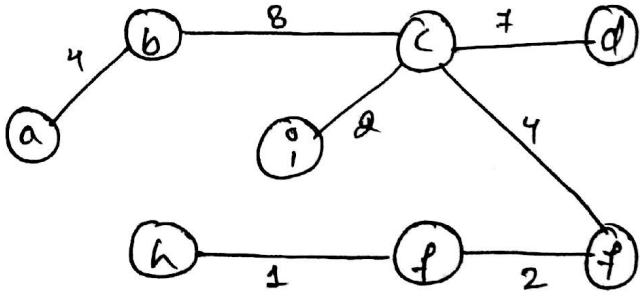
* (d, c, i, f, g, h) (a, b, e)



* (d, c, i, f, g, h, b) (a, e)

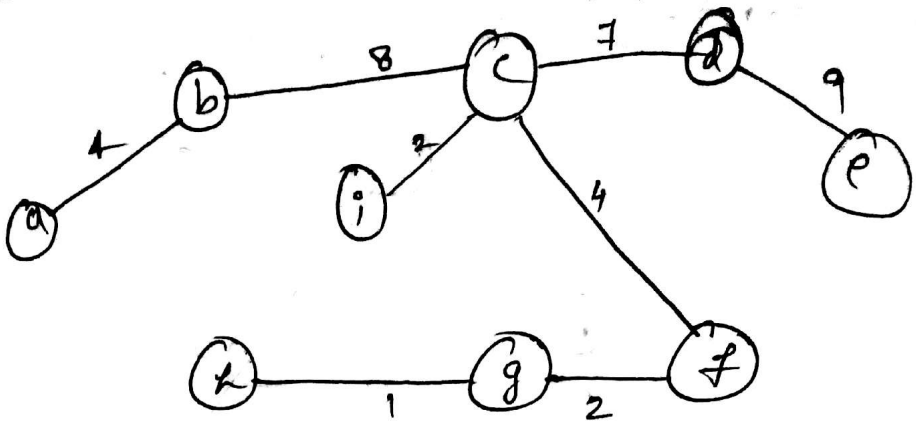


* (d, c, i, f, g, h, b, a) (e)



* (a, e, i, f, g, h, b, a, e)

Scanned by CamScanner



=37

Single Source Shortest Path

1) Dijkstra

Initialization of a single Source

① for each vertex $v \in V(G)$

② $d(v) \leftarrow \infty$

③ $d(s) \leftarrow 0$

④ $\pi(v) \leftarrow NIL$

Relaxation

Relax(u, v, w)

① if $d(v) > d(u) + w(u,v)$

② $d(v) \leftarrow d(u) + w(u,v)$

③ $\pi(v) \leftarrow u$

Dijkstra

Dijkstra(G, w, s)

① Initialize a Single Source

② $S \leftarrow \phi$

③ $Q \leftarrow V(G)$

Scanned by CamScanner

(4) while $Q \neq \phi$

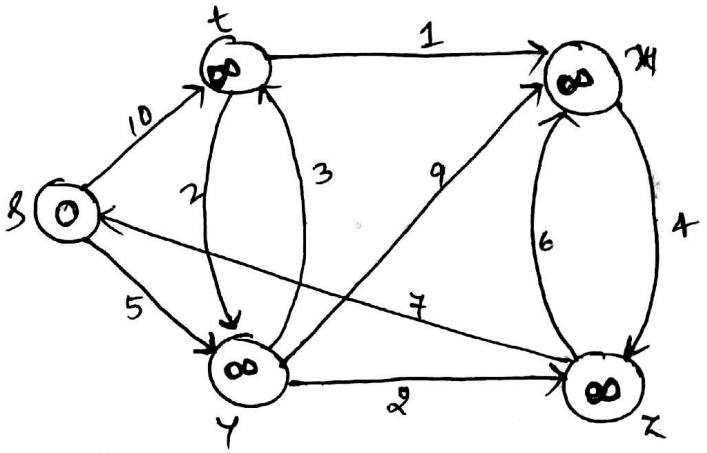
(5) do $u \leftarrow$ extract the min (Q)

(6) $S \leftarrow S \cup \{u\}$

(7) for each vertex $; v \in adj(u)$

(8) Relax (u, v, w)

Q-



(1)

(2) $S = \{\phi\}$

(3) $Q = \{s, t, x, y, z\}$

(4) while $Q \neq \phi$ (true)

(5) $u \leftarrow \{s\}$

$Q = \{s, t, x, y, z\}$

(6) $S \leftarrow \{s\} \cup \phi$

$S \leftarrow \{s\}$

(7) $v \leftarrow \{t, y\}$

(8) Relax for 't'

(1) if $d(t) > d(s) + 10$ (true)

(2) $d(t) \leftarrow 0 + 10$

(3) $\pi(t) \leftarrow s$

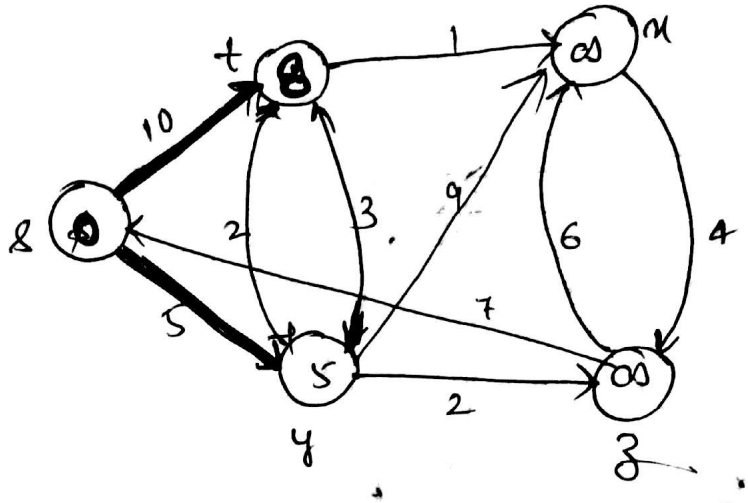
Scanned by CamScanner

⑥ relax for 'y' (false)

① if $d(y) > d(s) + 5$

② $d(y) \leftarrow 5$

③ $\pi(y) \leftarrow s$.



④ while $Q \neq \phi$ (true)

⑤ $u \leftarrow \{y\}$ (as min)

$Q = \{s, t, x, y, z\}$

⑥ $S \leftarrow \{s\} \cup \{y\}$

$S \leftarrow \{s, y\}$

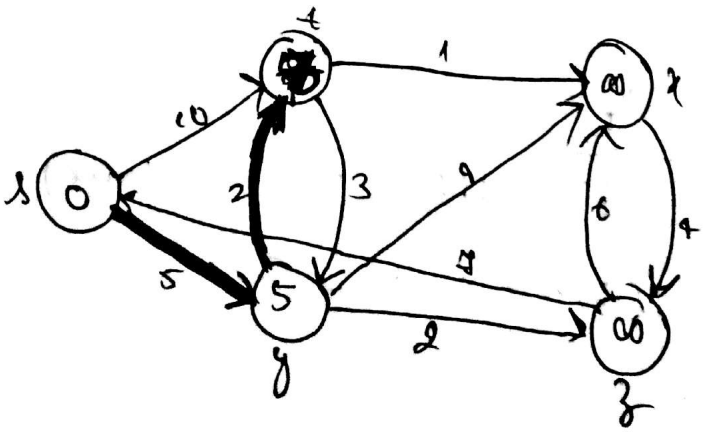
⑦ $V \leftarrow \{t, x, z\}$

⑧ Relax for 't'

① if $d(t) > d(y) + 2$. (true)

② $d(t) \leftarrow 7$

③ $\pi(t) \leftarrow y$.



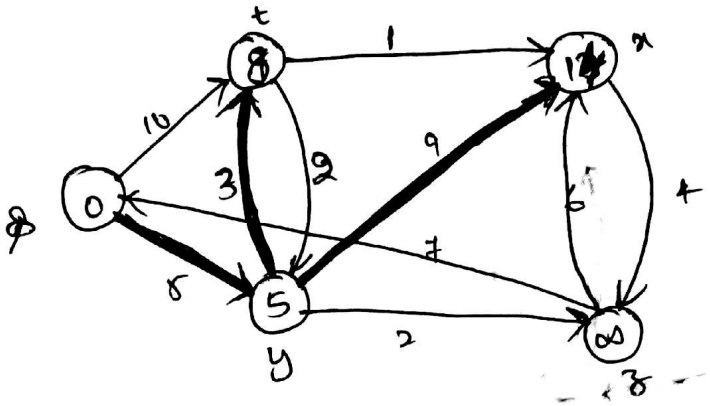
Scanned by CamScanner

(8) Relax for 'x'

(1) if $d(x) > d(y) + 9$ (true)

(2) $d(x) \leftarrow 5 + 9$ ie., 14

(3) $\pi(x) \leftarrow y$

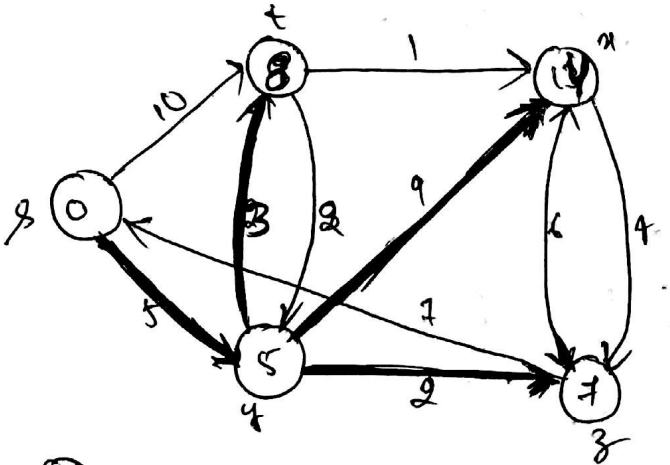


(8) Relax for 'z'

(1) if $d(z) > d(y) + 2$. (true)

(2) $d(z) \leftarrow 5 + 2$ ie., 7.

(3) $\pi(z) \leftarrow y$.



(4) while $Q \neq \phi$

(5) $u \leftarrow \{z\}$ as min.

$Q = \{s, t, x, y, z\}$

(6) $S \leftarrow \{s, y\} \cup \{z\}$

$S \leftarrow \{s, y, z\}$

(7) $v \leftarrow \{x, y, s\}$

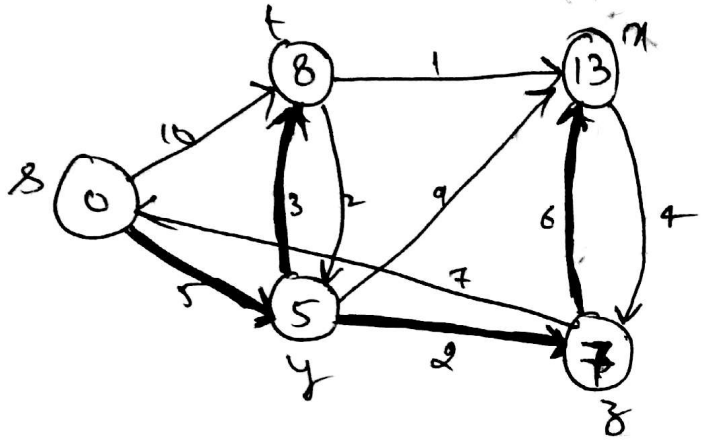
Scanned by CamScanner

(8) Relax for 'x'

① if $d(x) > d(z) + 6$ (true)

② $d(x) \leftarrow 7 + 6$ i.e. 13

③ $d(x) \leftarrow z$.



(8) Relax for 'y'

① if $d(y) > d(z) + 2$ (false)

(8) Relax for 's'

① if $d(s) > d(z) + 7$ (false

(4) while $Q \neq \phi$

(5) $u \leftarrow \{t\}$ as min

$Q = \{s, t, x, y, z\}$

(6) $S \leftarrow \{s, y, z, t\}$

(7) $v \leftarrow \{s, x, y,$

(8) Relax for 's'

① if $d(s) > d(t) + 7$ (false)

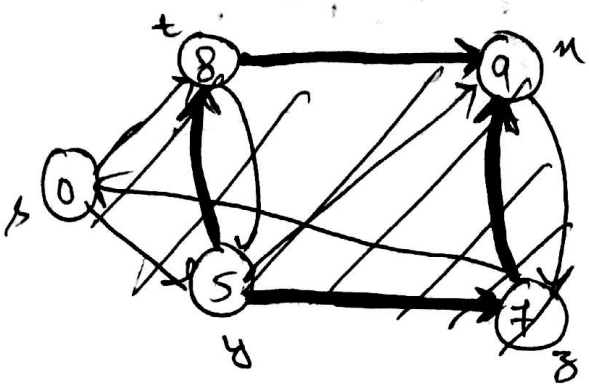
(8) Relax for 'x'

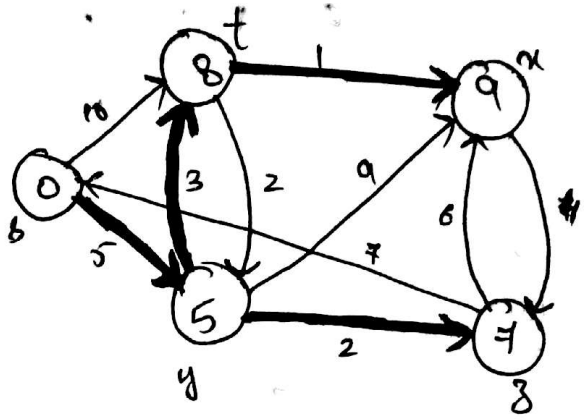
① if $d(x) > d(t) + 1$ (true)

② $d(x) \leftarrow 8 + 1$ i.e. 9

Scanned by CamScanner

(3) $\pi(x) \leftarrow t$





(8) Relax for 'y'

(1) if $d(y) > d(t) + 2$. (false)

~~(9) Relax for 'z'~~

(4) while $Q \neq \phi$

(5) $u \leftarrow \{x\}$

$Q = \{s, t, x, y, z\}$

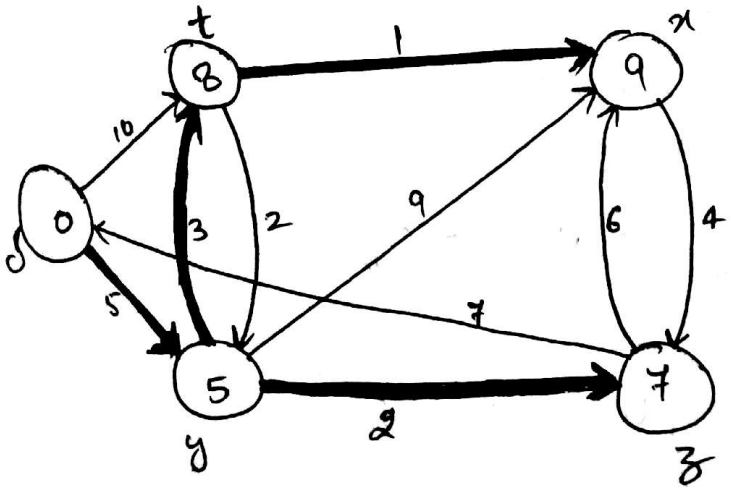
(6) $S \leftarrow \{s, y, z, t\} \cup \{x\}$

$S \leftarrow \{s, x, y, z, t\}$

(7) $v \leftarrow \{z\}$

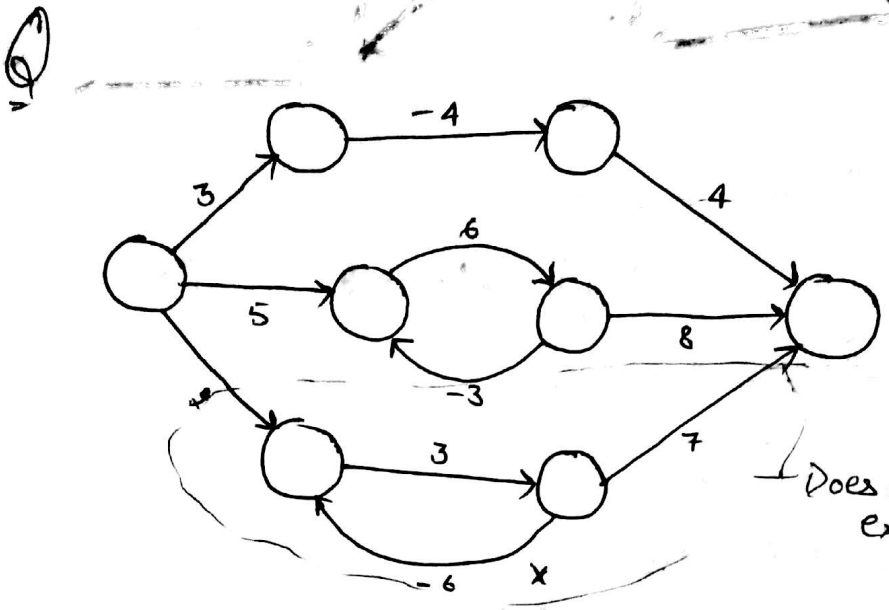
(8) Relax for 'z'

(1) if $d(z) > d(x) + 4$ (false)



Scanned by CamScanner

Bellman ford Algorithm



Bellman ford consider the negative weight but does not consider the negative weight cycle.

4th path does not exist as it consists of negative weight cycle.

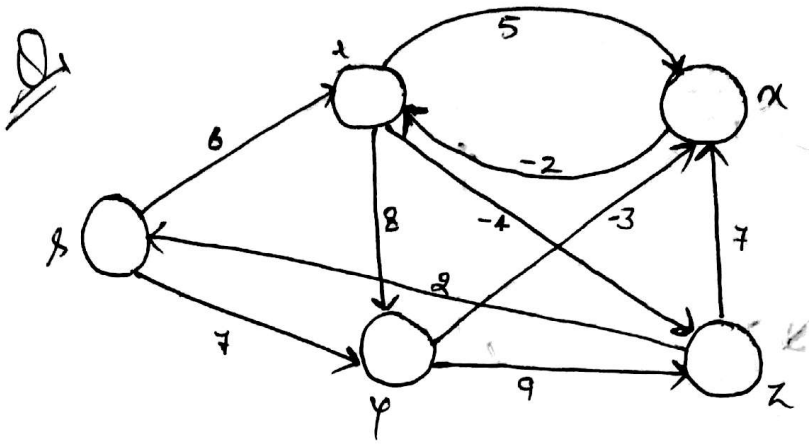
Bellman ford (G, w, s)

1. Initialize a single source
2. for i ← 1 to v(G) - 1
3. do for each edge (u,v) ∈ E(G)
4. Relax (u,v,w)
5. for each edge (u,v) ∈ E(G)
6. if d(v) > d(u) + w(u,v)
7. then return false.
8. Return true.

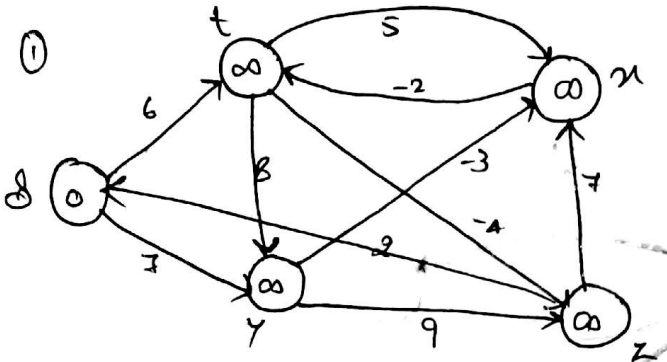
To check weight cyc

Scanned by CamScanner

Q.



Sol ①



② for $i = 1$ to 4.

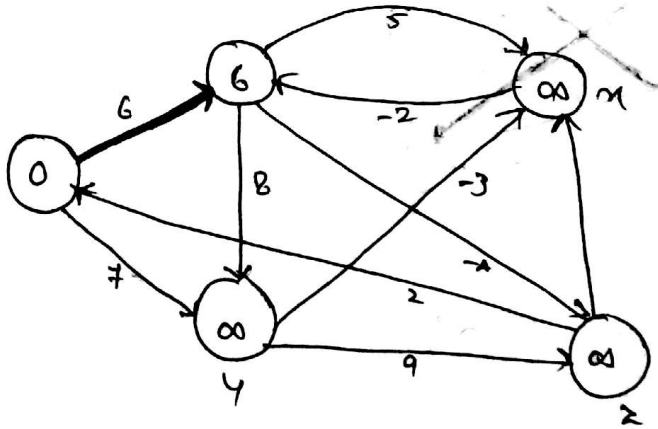
③ for edge (s,t) as adj.

④ Relax (s,t,w)

① if $d(t) > d(s) + 6$.

② $d(t) \leftarrow 6$

③ $\pi(t) \leftarrow s$.



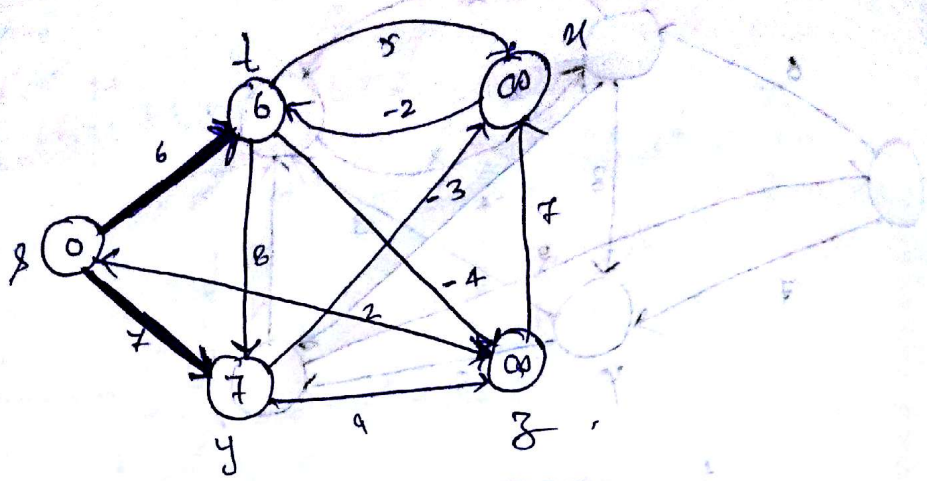
⑤ for edge (s,y) as adj

④ Relax (s,y,w)

① if $d(y) > d(s) + 7$.

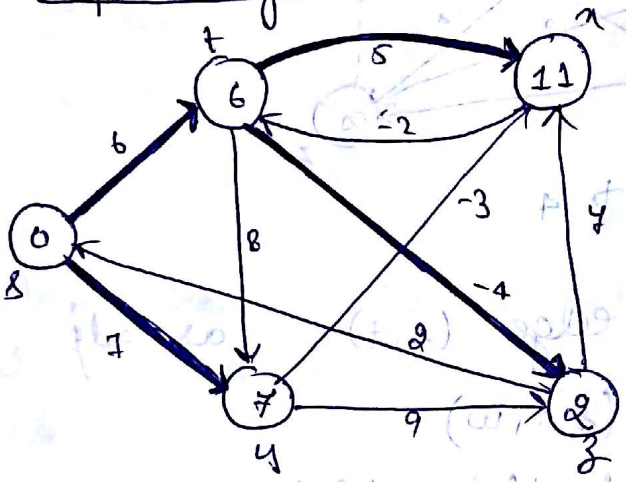
② $d(y) \leftarrow$ [illegible]

Scanned by CamScanner

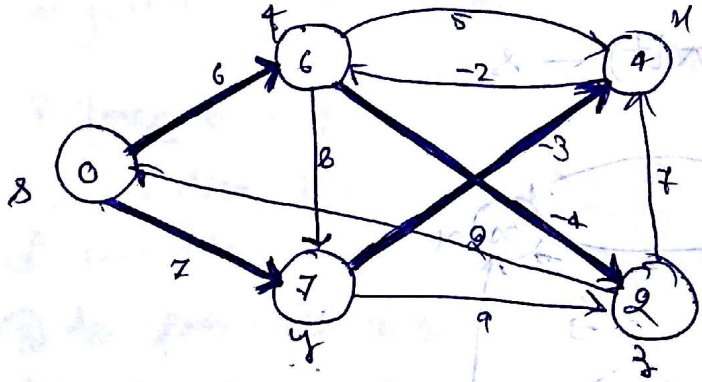


(2) for i=2.

Adj. to edge, (s,t)

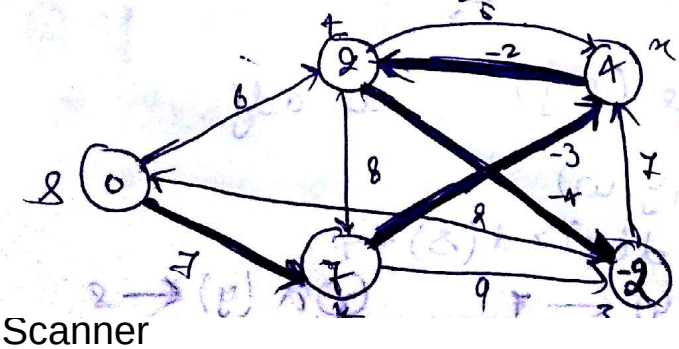


Adj to edge (s, y)



for i=3

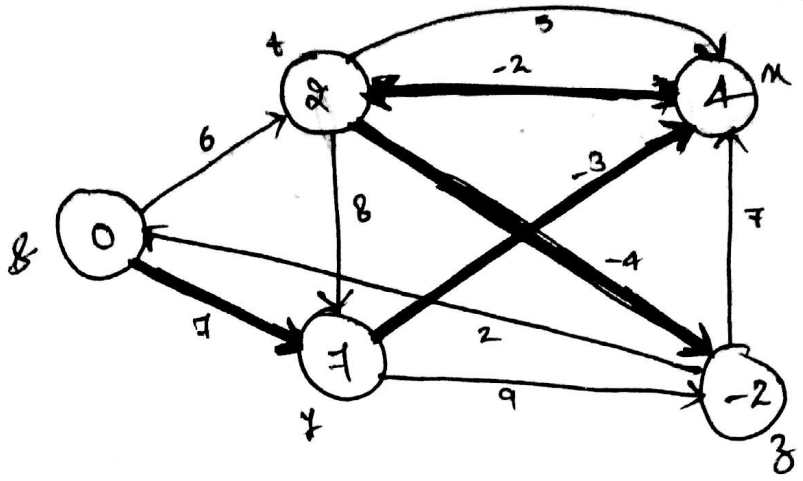
Adj to edge (y, x)



remember: change this

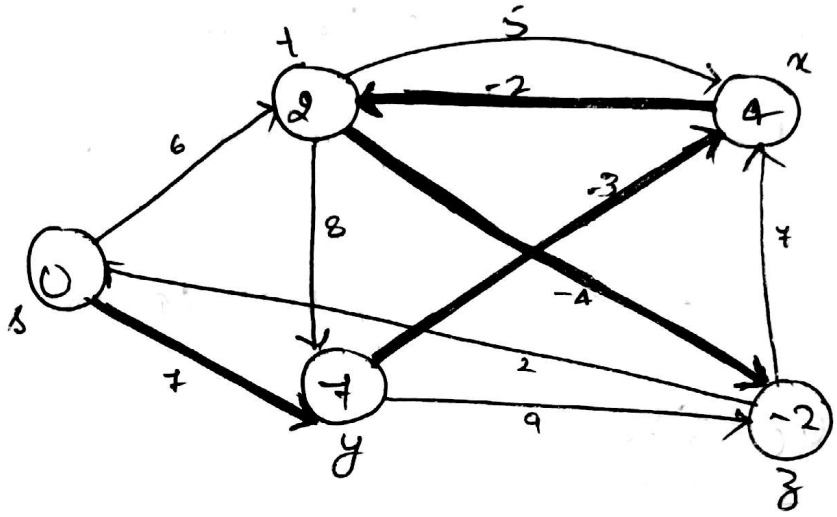
Scanned by CamScanner

Adj to edge (t, z)



for, i = 4

Adj to edge. (x, t)



Scanned by CamScanner

Knapsack Problem

It can be divided into two categories—

① - 0 & 1

② - fractional

0/1 Knapsack problem can not be solved by greedy approach because it does not give the best sol^n so 0/1 knapsack problem can be solved by dynamic approach

fractional can be solved by greedy Approach

Greedy Approach

Knapsack problem

↳ fractional Knapsack problem

Q there are 5 items, w = 50kg

Items	w	v	$P_i = V_i / w_i$
1	5	30	6
2	10	20	2
3	20	100	5
4	30	90	3
5	40	160	4

Find out P_i

Scanned by CamScanner

801

Step-① Find out P_i

$$P_i = \frac{V_i}{W_i}$$

Step② set the P_i into decreasing order.

Items	w	V	P_i
1	5	30	6
3	20	100	5
5	40	160	4
4	30	90	3
2	10	20	2

Step ③

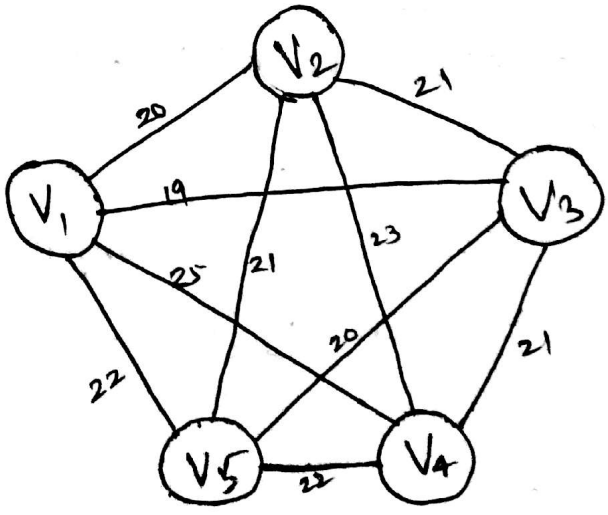
$$w = 50 - 5 = 45 - 20 = 25 - 25 = 0$$

$$v = 30 + 100 + 100 = 230$$

Travelling Salesman problem

A salesman travels from one city to another and return to the same city, it means all cities should be covered with minimum distance & starting and distance should be same.

Q)



Scanned by CamScanner

Sol

	v_1	v_2	v_3	v_4	v_5	
start → v_1	0	20	(19)	25	22	$v_1 \to v_3$
v_2	20	0	21	(23)	21	$v_2 \to v_4$
v_3	19	21	0	21	(20)	$v_3 \to v_5$
v_4	(25)	23	21	0	22	$v_5 \to v_1$
v_5	22	(21)	20	22	0	$v_5 \to v_2$

$\Rightarrow v_1 \to v_3 \to v_5 \to v_2 \to v_4 \to v_1$

$= 108$

Suppose

$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_1$

$= 106$

So, travelling salesman problem can not be solved by greedy approach with optimal sol^n.

Huffman Code (Greedy)

It is used to compress the data based on greedy approach.

There are two techniques to compress the data by Huffman code.

1) Fixed length
2) Variable length

Scanned by CamScanner

Fixed length

	a	b	c	d	e	f
frequency	45000	13000	12000	16000	9000	5000
let.	000	001	010	011	100	101

total bit = 11akh x 3

= 3 lakh

Variable length:

a	b	c	d	e	f
↓	↓	↓	↓	↓	↓
0	001	100	111	1101	1100
frequency 45000	13000	12000	16000	9000	5000

total bit : 45000 + 39000 + 36000 + 48000 + 36000 + 20000

= 224,000

Huffman code

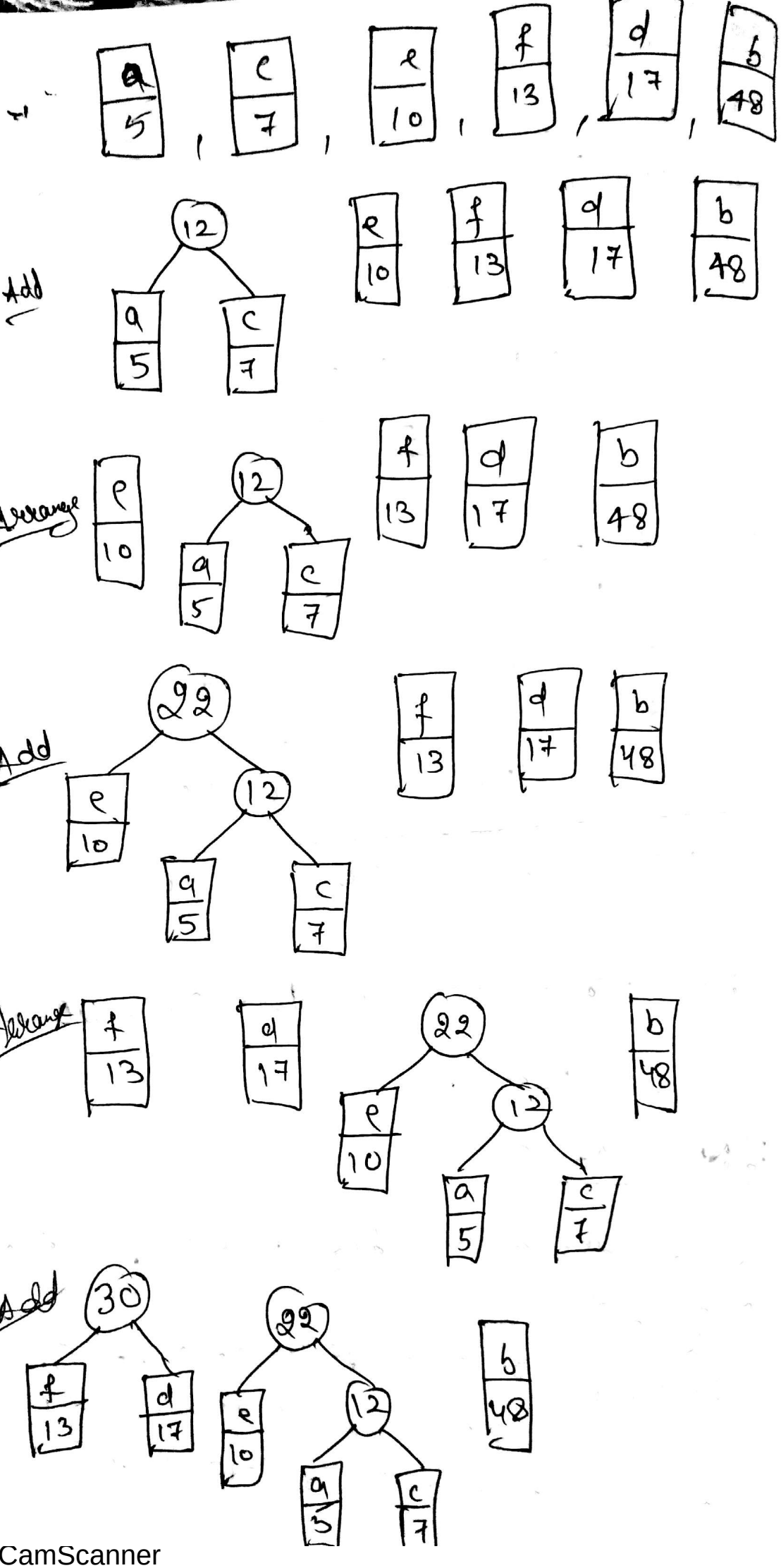
Q what are the optimal huffman code for the set of frequency.

a-5, b-48, c-7, d-17, e-10, f-13.

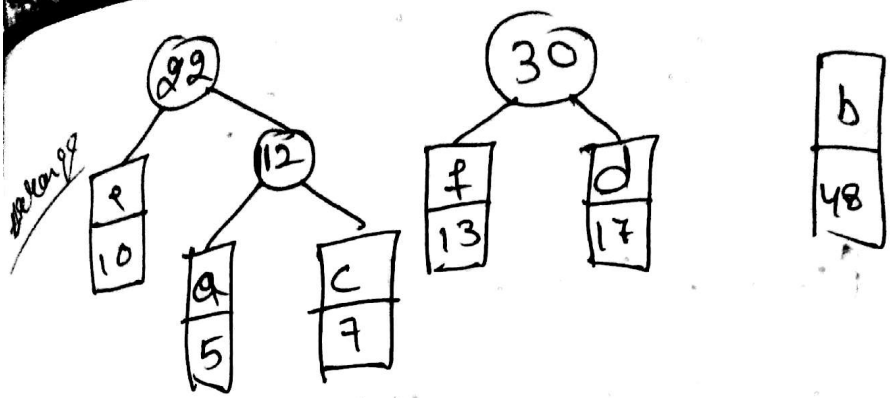
step(i) Set the frequency into increasing order

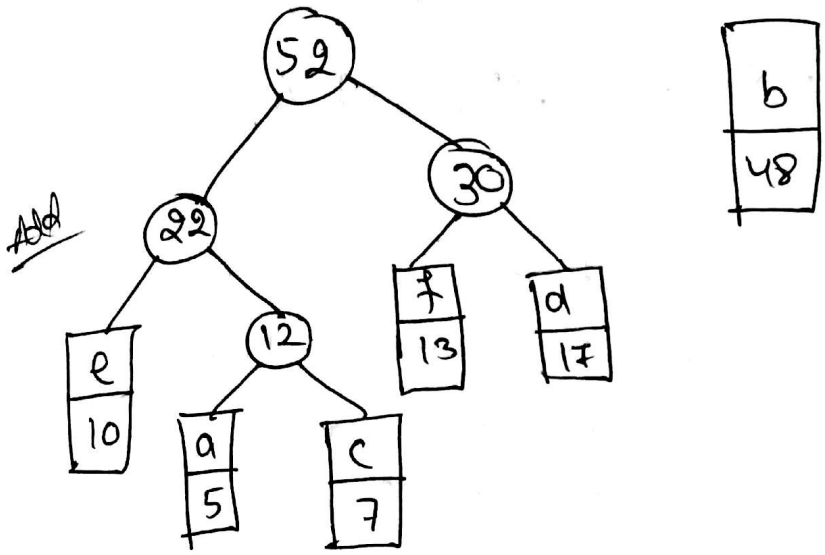
a-5, c-7, e-10, f-13, d-17, b-48

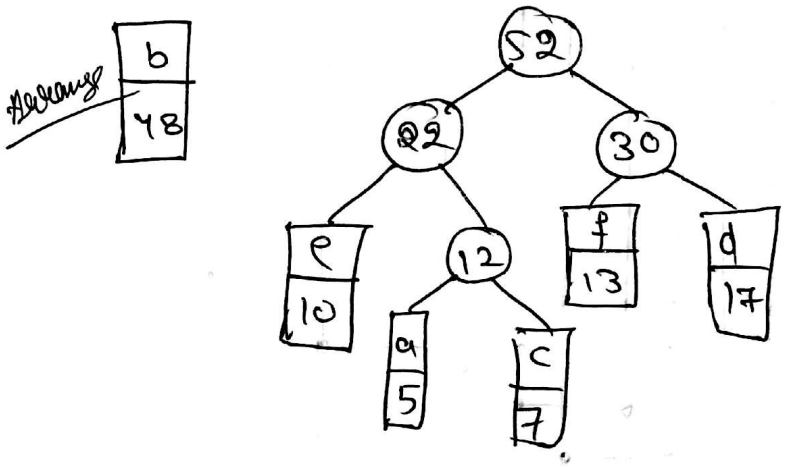
Scanned by CamScanner

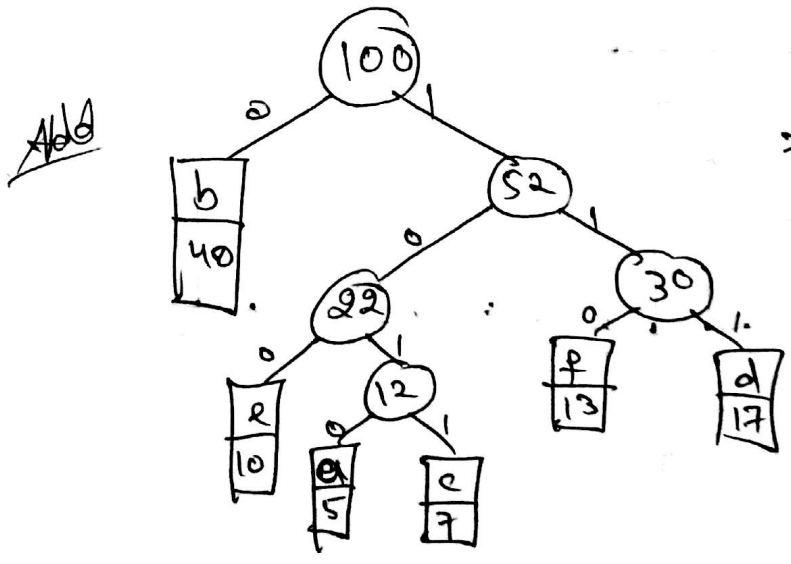


Scanned by CamScanner









a = 1010
b = 0
c = 1011
d = 111
e = 100
f = 110

Scanned by CamScanner

Activity Selection Problem

Q.

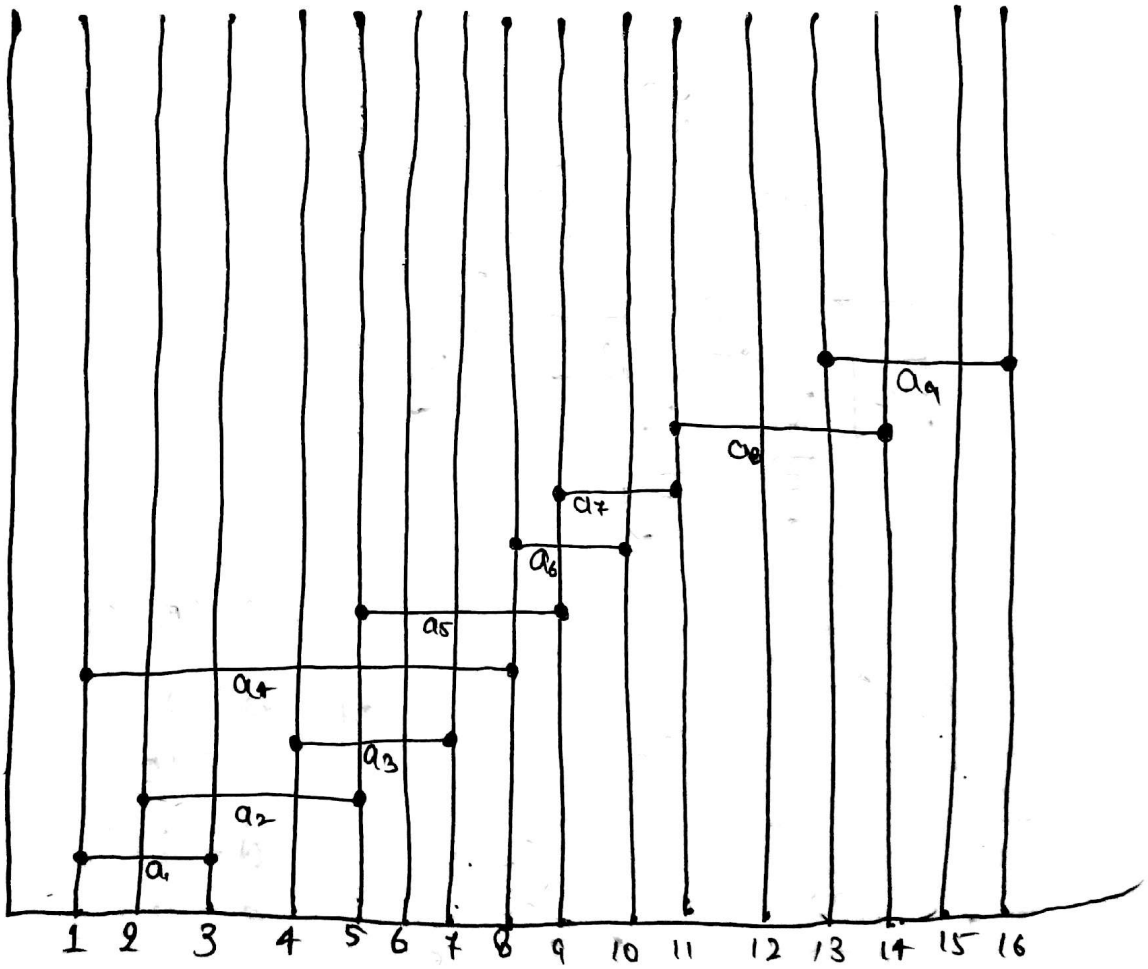
$S = (1, 2, 3, 4, 5, 6, 7, 8, 9)$

$S_i = (1, 2, 4, 1, 5, 8, 9, 11, 13)$

$f_i = (3, 5, 7, 8, 9, 10, 11, 14, 16)$

Sol^n

Step (1) Arrange the activities in increasing order of finishing time and motive is that where the max. no. of activities can take place



→ a_1, a_3, a_6, a_8

Scanned by CamScanner

BACK TRACKING APPROACH :

It works on intelligency suppose we have to make a series of decision among various choice where we don't have enough information to know which choice is the best.

Some sequence of choice may be a solution to our problem.

Backtracking is a mathematical approach to try out various sequence of decision until we find out one sol?.

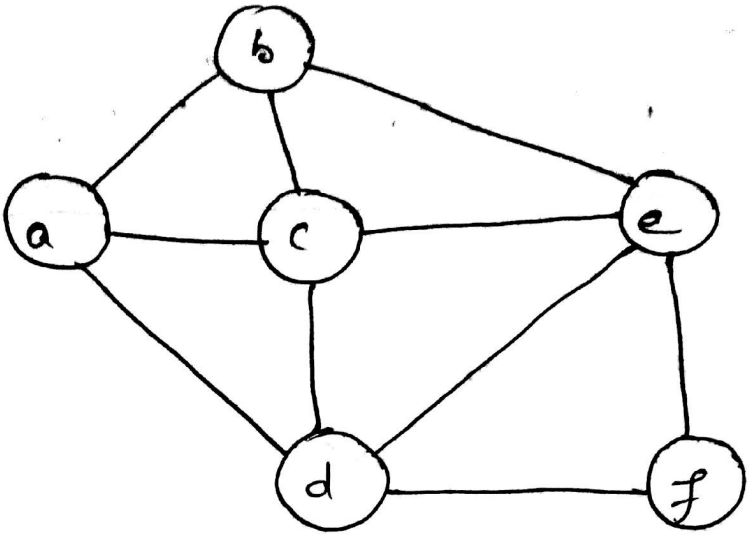
There are different examples of backtracking
- Hamiltonian Circuit problem
- Graph Colouring
- n-queen problem
- Sum of subset

Hamiltonian Circuit Problem:

All vertex should be covered & starting and ending vertex should be same. we will find out hamiltonian circuit with the help of back-tracking.

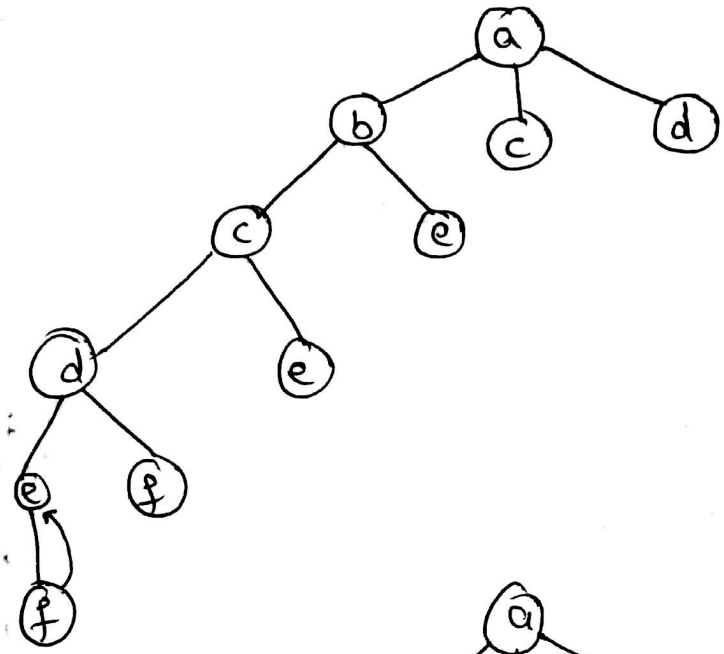
Scanned by CamScanner

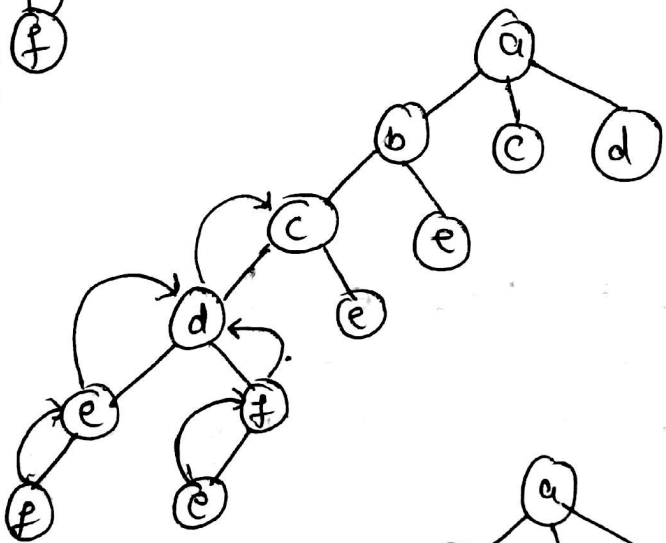
Q.

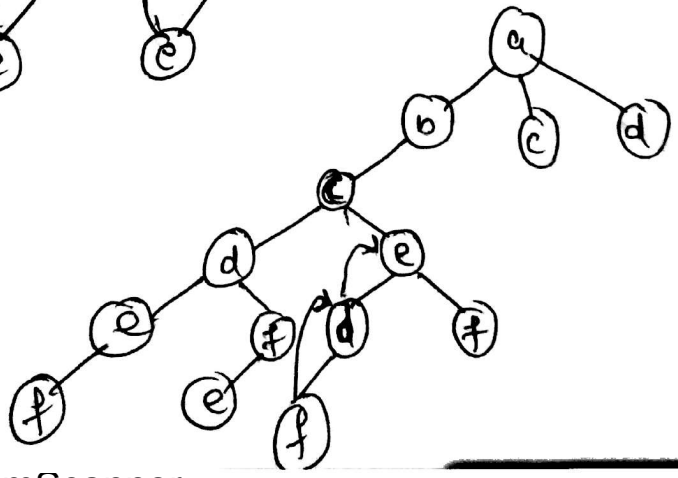


Sol

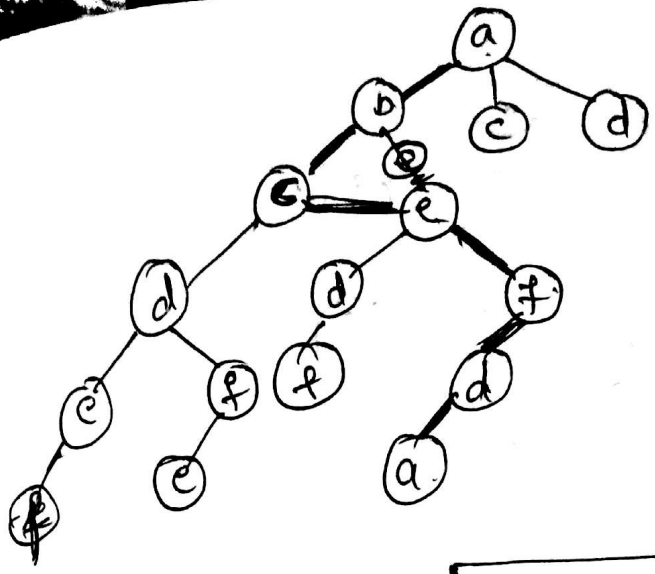
State space diagram







Scanned by CamScanner



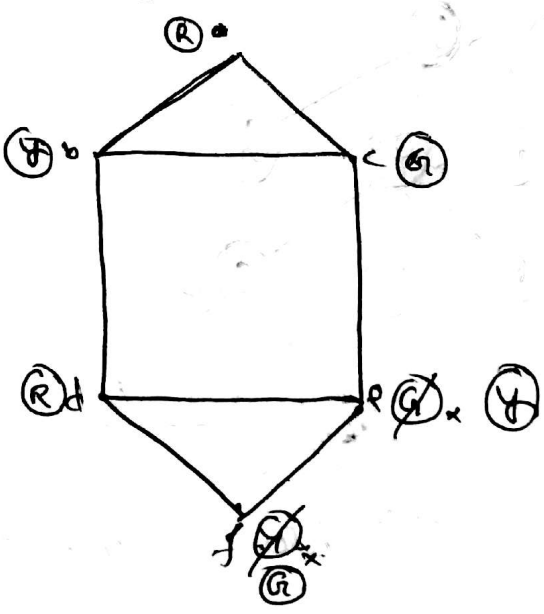
⇒ a, b, c, e, f, d, a

Graph Colouring:

It means all vertex should be coloured and no adjacent vertex should be of same colour.

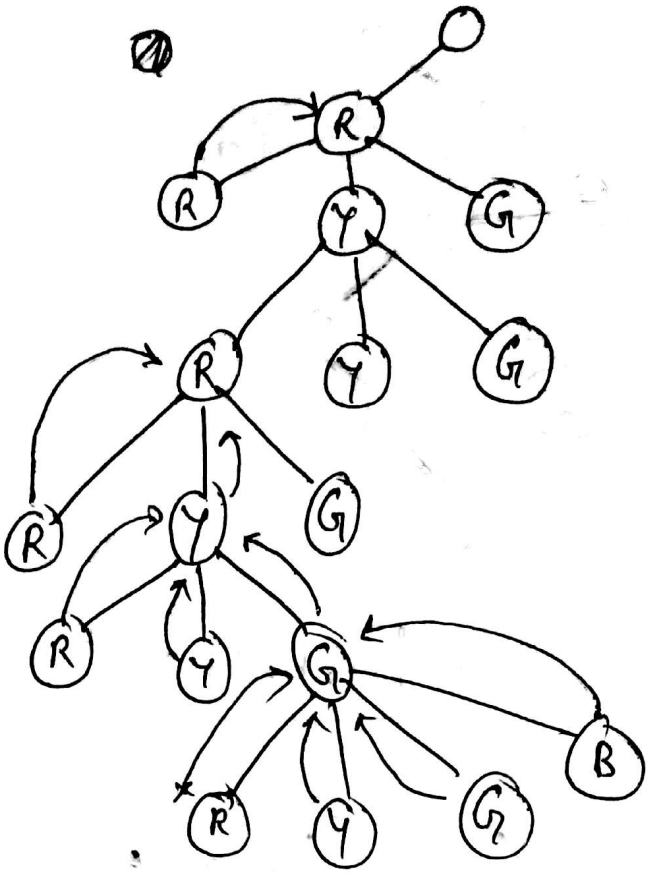
Chromatic Number: It means minimum no. of colour are required to colour any graph.

Q

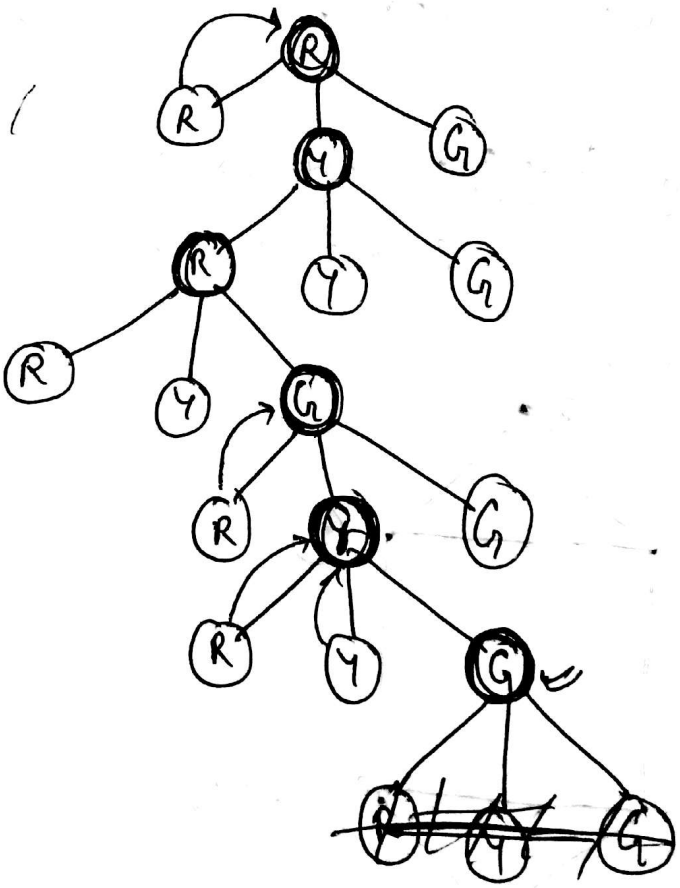


Chromatic NO = 3.

Scanned by CamScanner



∴ Chromatic No = 4 (Wrong)



∴ Chromatic No = 3 (correct)

Scanned by CamScanner

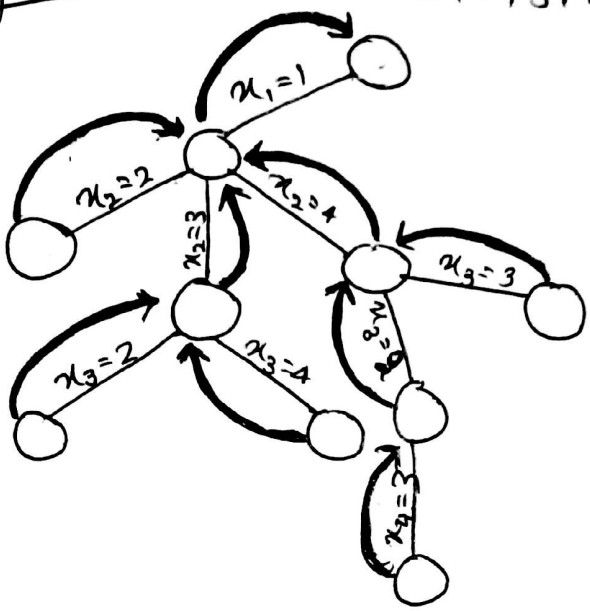
n-Queen Problem

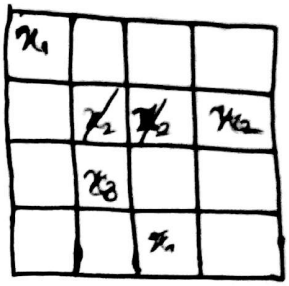
1-queen, 2-queen, 3-queen problem cannot exist

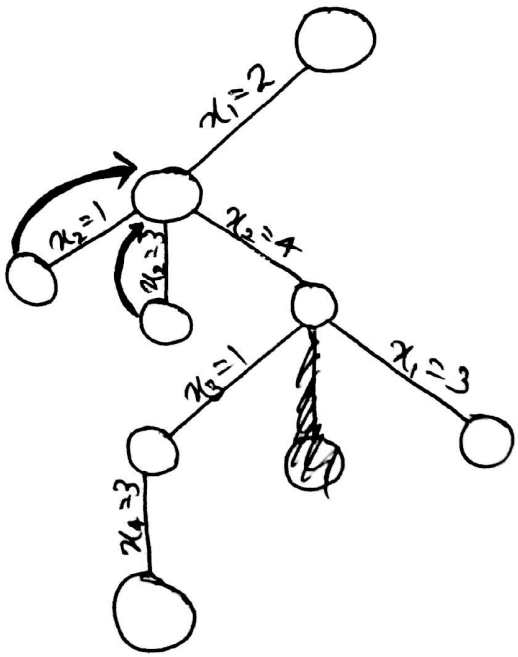
Queens can attack each other when they are in same row, same column & same diagonal.

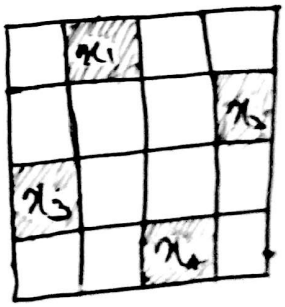
4-queen problem

x_1, x_2, x_3, x_4 = queen
1, 2, 3, 4 = row number.









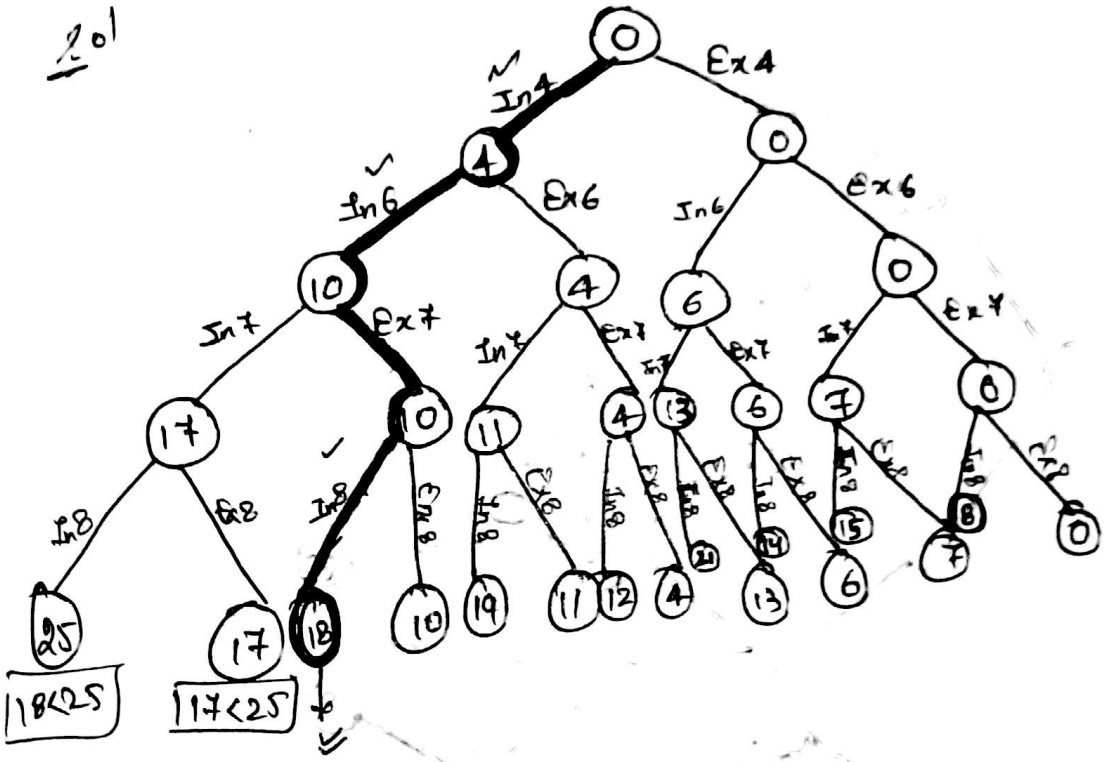
Scanned by CamScanner

Sum of Subsets

Q Find all possible subsets of S that sum to m and draw the state space diagram or back-tracking tree

$S = \{4, 6, 7, 8\}$ & $m = 18$.

Sol



→ 4, 6, 8.

Q $S = \{5, 7, 10, 12, 15, 18, 20\}$ $M = 35$

Sol

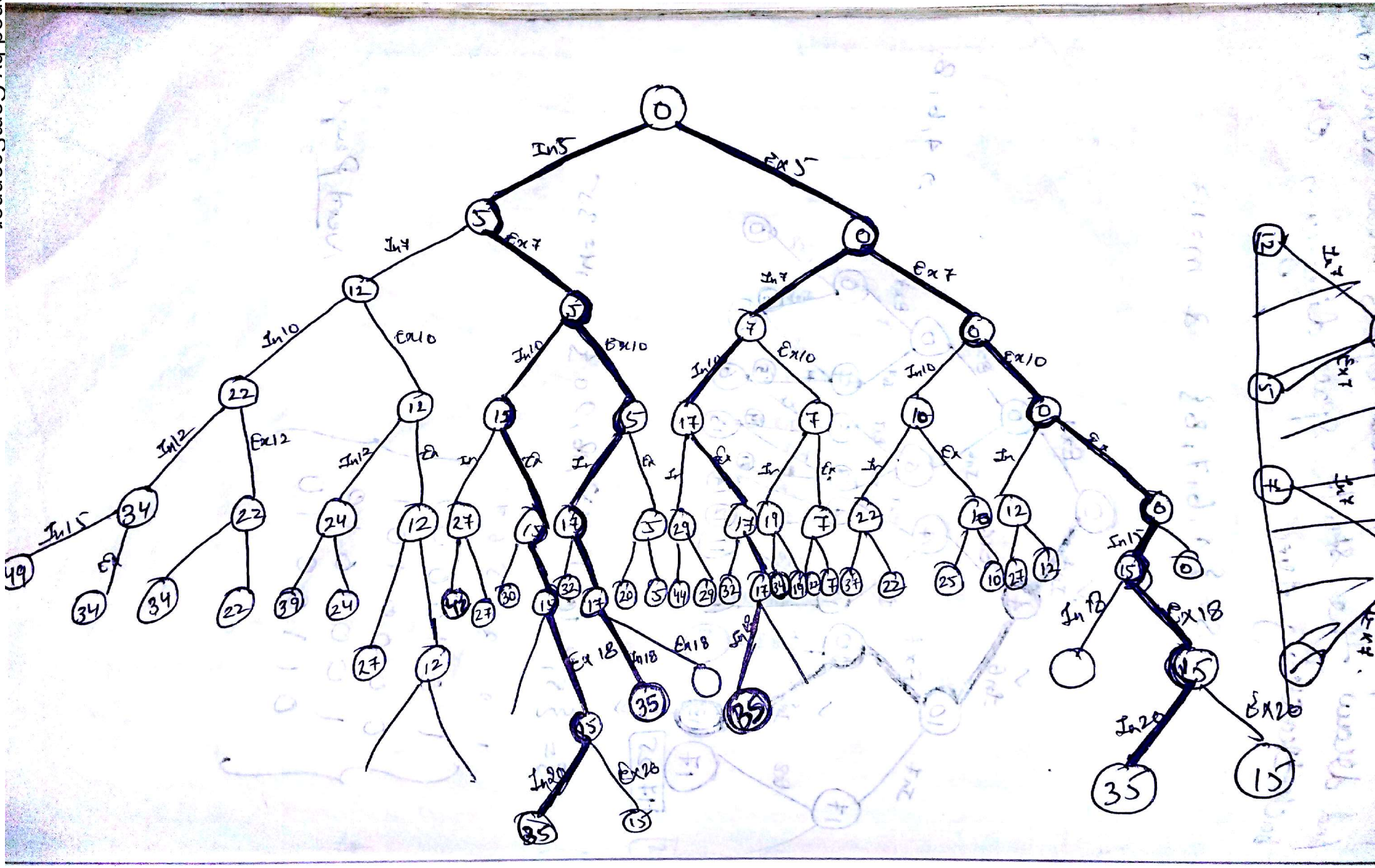
A

1	1	3	4	5	6	7
1	0	1	0	0	0	1
0	0	0	0	1	0	1
1	0	0	1	0	1	0
0	1	1	0	0	1	0

Next Page

Scanned by CamScanner

Scanned by CamScanner



0-1 Knapsack problem [Dynamic]

S.no. n	value P	weight w
1	1	2
2	2	3
3	5	4

Knapsack = 6

Sol) Total no. of ways = $2^n = 2^3 = 8$

for 1st item: $S_0^0 = (0,0)$ (value, weight) neglect; $S_1^0 = (1,2)$ consider

for 2nd item: $S_0^1 = ((0,0)(1,2))$ neglect; $S_1^1 = ((2,3),(3,5))$ consider

for 3rd item: $S_0^2 = ((0,0)(1,2)(2,3)(3,5))$ neglect; $S_1^2 = ((5,4)(6,6)(7,7)(8,9))$ consider

$S^3 = ((0,0)(1,2)(2,3)(3,5)(5,4)(6,6)(7,7)(8,9))$

$S^4 = ((0,0)(1,2)(2,3)(3,5)(5,4)(6,6))$ as weight is greater than Knapsack.

Step 2 By Purging rule: suppose (P_i, w_i) & (P_j, w_j)

1st rule ~~Suppose two~~ if $P_j > P_i$ and $w_j < w_i$ then discard (P_i, w_i)

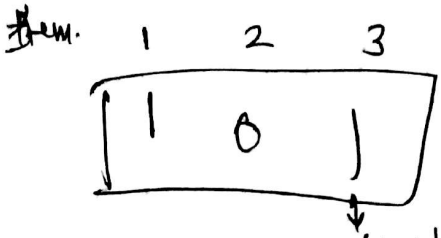
2nd rule if $P_j < P_i$ and $w_j > w_i$ then discard (P_j, w_j)

Scanned by CamScanner

So, neglect (3,5)

$S_3^5 = \{(0,0)\ (1,2)\ (2,3)\ (5,4)\ (6,6)\}$.

∴ $S = (6,6)$ from S^0, S^1, S^2



as, (6,6) is from S^2 (Consider)
(1,2) is from S^1 (neglect)
(1,2) is from S^0 (Consider)

66 (wt is selected)
− 54 (weight of item 3)
(1,2)

Branch & Bound:

* Complexity of back tracking & branch & bound contain the exponential time.
* Back tracking always prefer DFS (depth first search) while branch & bound prefer both (DFS or BFS)
* Like back tracking, branch & bound contains all possible solutions and all possible solutions are display with the help of state space tree.
* There are three difference b/w back tracking and branch & bound —

1) Branch & bound is used for optimization.
2) A back tracking search the preorder DFS. branch & bound has no restrictions for the searching.
3) Branch & bound must completely search the state space tree because if entire state space tree is not searched, possibilit

Scanned by CamScanner

exist that optimal sol:- is exist, on unsearched tree.

* In this technique, we consider the bound means value: we calculate the bound at each node. It can be lower bound or upper bound.

Knapsack problem (Branch & Bound)

The knapsack problem where we have to fill the knapsack of capacity 'W' with given 'n' items which have weight 'w_i' and value 'v_i' in such a manner, that total weight of items should not exceed from the knapsack and obtain the maximum value.

$$\text{upper bound } (UB) = v_i + (W - w_i) * \frac{v_{i+1}}{w_{i+1}}$$

Q:

Item	weight (w_i)	value (v_i)
I_1	6	50
I_2	8	48
I_3	6	30
I_4	4	16

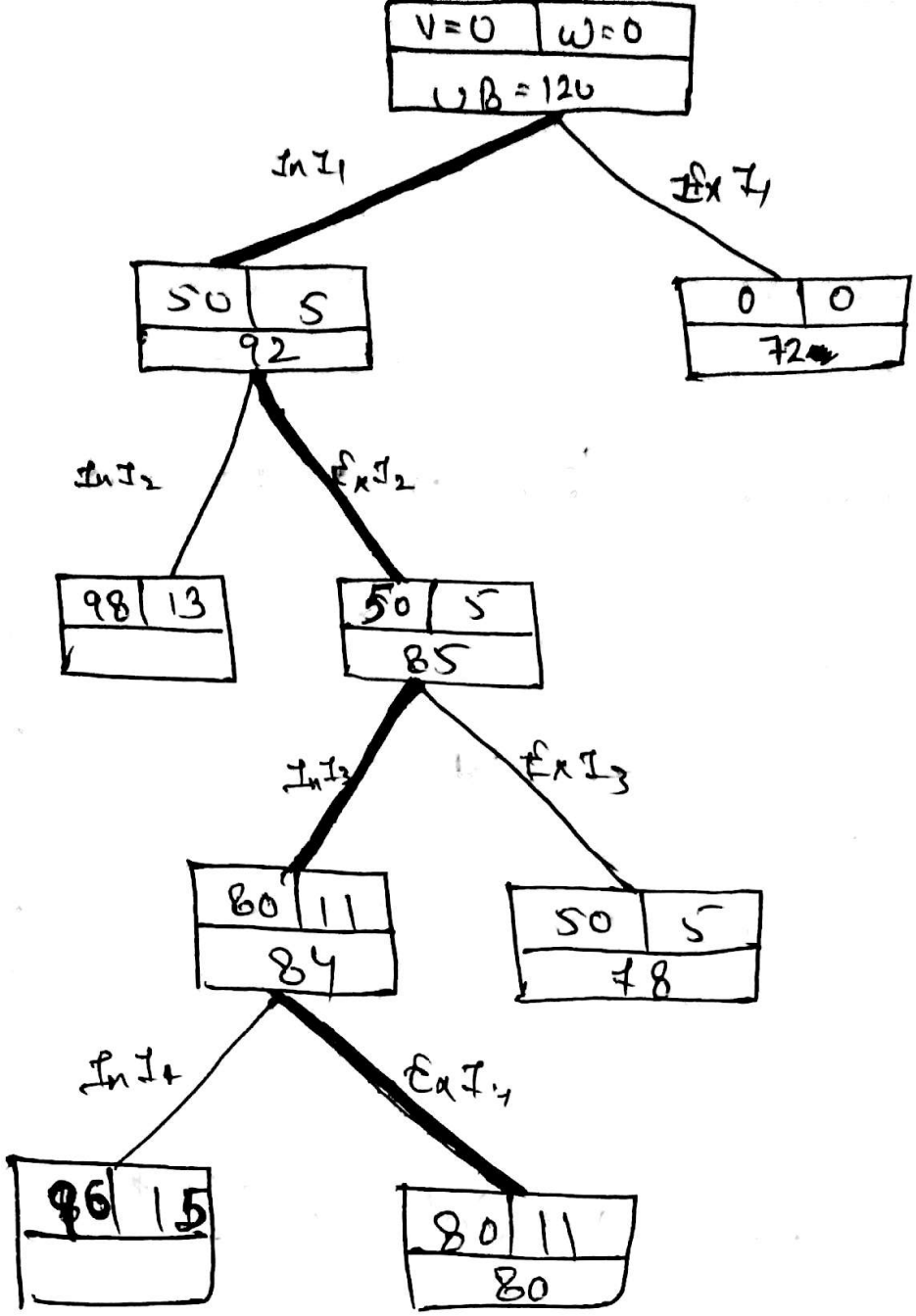
Knapsack = 12

Sol

Step. 1: Find the P_i & set the 'P_i' into decreasing order.

Item	weight	value	P_i
I_1	5	50	10
I_2	8	48	6
I_3	6	30	5
I_4	4	16	4

Scanned by CamScanner



1	2	3	4
1	0	1	0

⇒ value (v) = 80

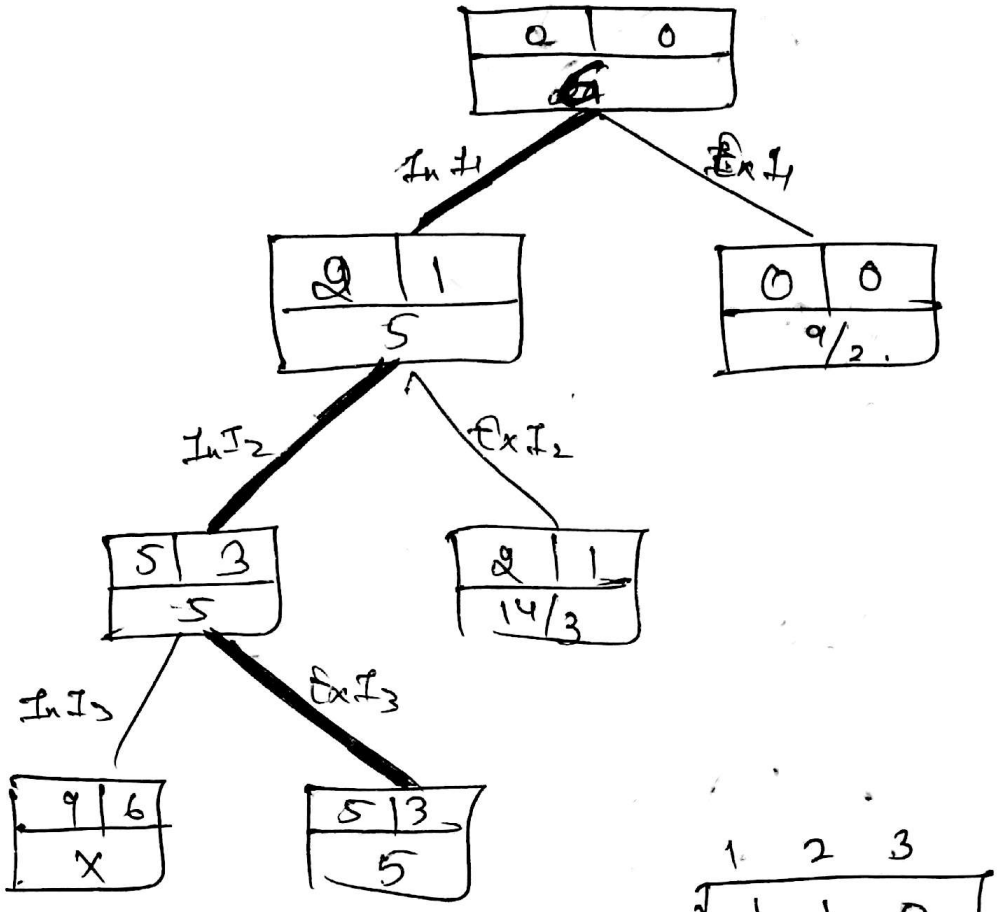
Q

Item	weight	value
1	1	2
2	2	3
3	3	4

Knapsack = 3.

Scanned by CamScanner

Items	weight	value	p_i
1	1	2	2
2	2	3	3/2
3	3	4	4/3.



1	2	3
1	1	0

value (v) = 5

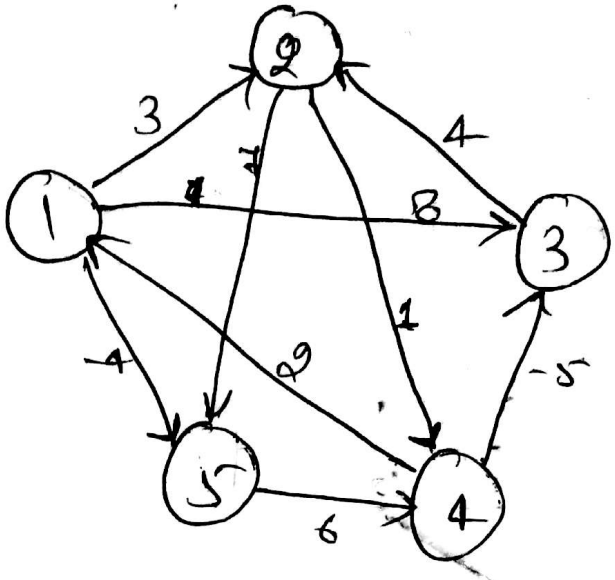
~~Travelling Salesman Problem~~

Floyd warshall : (All pair shortest path)

All pair shortest path can be solved by single source shortest path by considering each vertex as a source but complexity will be increases

Scanned by CamScanner

Q =



Sol:

$D_0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

$D_1 =$

0	3	8	∞	-4
∞	0	∞	1	7
∞	4	0	∞	∞
2	5	-5	0	-2
∞	∞	∞	6	0

$D_2 =$

0	3	8	4	-4
∞	0	∞	1	7
∞	4	0	5	11
2	5	-5	0	-2
∞	∞	∞	6	0

Scanned by CamScanner

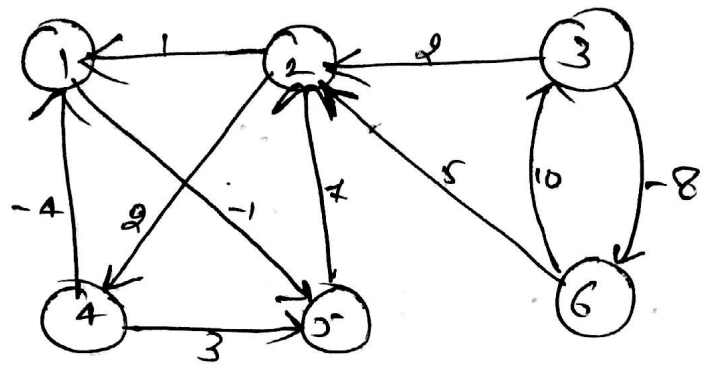
$$D_3 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

$$D_5 = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

Scanned by CamScanner

Q



$D_0 =$

	1	2	3	4	5	6
1	0	∞	∞	∞	-1	∞
2	1	0	∞	2	∞	∞
3	∞	2	0	∞	∞	-8
4	-4	∞	∞	0	3	∞
5	∞	7	∞	∞	0	∞
6	∞	5	10	∞	∞	0

$D_1 =$

	1	2	3	4	5	6
1	0	∞	∞	∞	-1	∞
2	1	0	∞	2	0	∞
3	∞	2	0	∞	∞	-8
4	-4	∞	∞	0	-5	∞
5	∞	7	∞	∞	0	∞
6	∞	∞	10	∞	∞	0

$D_2 =$

0	∞	∞	∞	-1	∞
1	0	∞	2	0	∞
3	2	0	4	2	-8
-4	∞	∞	0	-5	∞
8	7	∞	9	0	∞
6	5	10	7	5	0

Scanned by CamScanner

D_3^2

$$\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{bmatrix}$$

D_4^2

$$\begin{bmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

D_5^2

$$\begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

Scanned by CamScanner

$$D_6 = \begin{bmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{bmatrix}$$

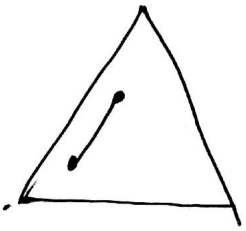
Scanned by CamScanner

Convex Hull:

A polygon is a collection of line segment forming a cycle and not crossing each other

Convex polygon: If every line segment drawn b/w ~~the~~ ~~two~~ any two points inside the polygon lies entirely inside the polygon.

eg)



Graham Scan:

to find out the extreme points.

Step 1: Let P_0 the point in queue with the minimum angle from x-axis.

Step 2: Let the remaining queue in queue and sort by angle around P_0.

Step 3: top (S) ← 0

Step 4: PUSH (P_0, S)

Step 5: PUSH (P_1, S)

Step 6: PUSH (P_2, S)

Step 7: for (i=3 to m)

Scanned by CamScanner

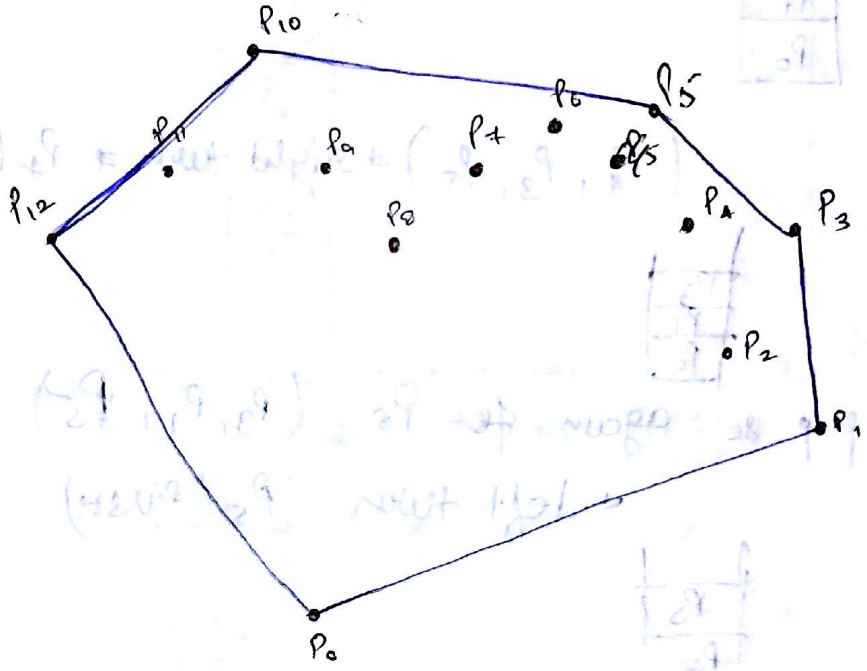
Step 8 : do while the angle formed by points next to top, top(s), top(s-1) then P_i, makes a non-left turn

Step 9 : pop(s)

Step 10 : else PUSH (S_i, P_i)

Step 11 : Return s

Q



Sol

for P_3 (P_1, P_2, P_3) ⇒ right turn ⇒ P_2 (Pop)

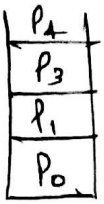
as pop so again for P_3.

(P_0, P_1, P_3) ⇒ left turn ⇒ P_3 (PUSH)

Scanned by CamScanner

for P_4 (P_1, P_3, P_4)

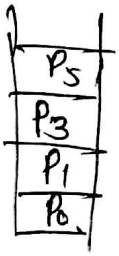
left turn ⇒ Push (P_4)



for P_5 (P_4, P_3, P_5) ⇒ right turn ⇒ P_4 (Pop

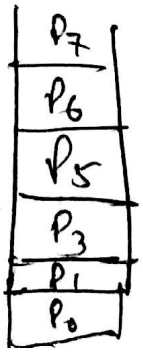
as pop so again for P_5 (P_3, P_1, P_5)

⇒ left turn P_5 (PUSH)



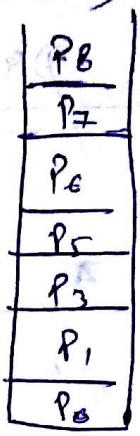
for P_6 (P_5, P_3, P_6) ⇒ left turn ⇒ P_6 (push)

for P_7 (P_6, P_5, P_7) ⇒ left turn ⇒ P_7 (Push)

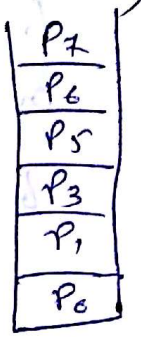


Scanned by CamScanner

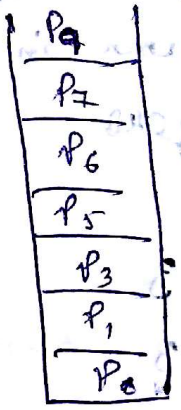
for P_8 (P_7, P_6, P_8) ⇒ left turn ⇒ P_8 (PUSH)



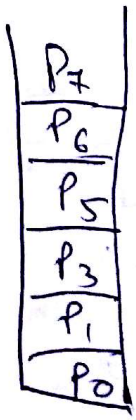
for P_9 (P_9, P_8, P_7) ⇒ right turn ⇒ P_8 (POP)



as pop so again for P_9 (P_7, P_6, P_9) ⇒ left turn P_9 (PUSH)

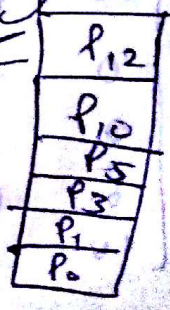


for P_{10} (P_7, P_9, P_{10}) ⇒ right turn P_9 (POP)



again (P_7, P_6, P_{10}) ⇒

Final



Scanned by CamScanner

Travelling Salesman problem (Branch & Bound)

~~In this~~

Q.

	1	2	3	4	5
1	∞	7	3	12	8
2	3	∞	6	14	9
3	5	8	∞	6	18
4	9	3	5	∞	11
5	18	14	9	8	∞

Sol:

① Find out reduced cost matrix by subtracting from each row & column in that way one entry should be zero.

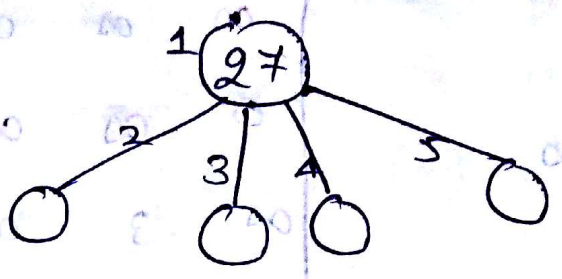
$$\begin{bmatrix} \infty & 4 & 0 & 9 & 5 \\ 0 & \infty & 3 & 11 & 6 \\ 0 & 3 & \infty & 1 & 13 \\ 6 & 0 & 9 & \infty & 8 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix} \begin{matrix} 3 \\ 3 \\ 5 \\ 3 \\ 8 \end{matrix}$$

$= 3+3+5+3+8 = 22.$

$$\begin{bmatrix} \infty & 4 & 0 & 9 & 0 \\ 0 & \infty & 3 & 11 & 1 \\ 0 & 3 & \infty & 1 & 8 \\ 6 & 0 & 9 & \infty & 3 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix} \quad 5$$

Scanned by CamScanner

$= 22 + 5 = 27$



(2) for $i = 1$ & $j = 2, 3, 4, 5$.

i.e., for (1, 2.) (row, column)

$A(2,1) = \infty$

$A(j,1) = \infty$.

$$\begin{array}{c} \\ 1 \\ \\ \\ 2 \end{array} \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 3 & 11 & 1 \\ 0 & \infty & \infty & 1 & 8 \\ 6 & \infty & 2 & \infty & 3 \\ 10 & \infty & 1 & 0 & \infty \end{bmatrix}$$

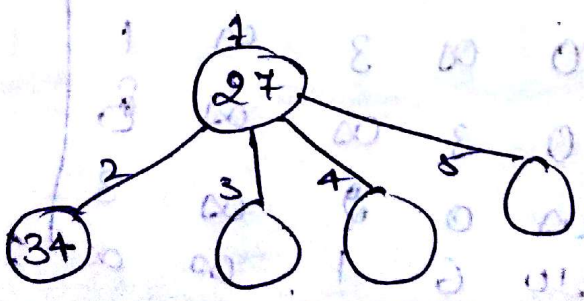
Reduce

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 10 & 0 \\ 0 & \infty & \infty & 1 & 8 \\ 4 & \infty & 0 & \infty & 1 \\ 10 & \infty & 1 & 0 & \infty \end{bmatrix} + 1 + 2 = 3.$$

(3) Reduced Cost = Previous Reduced Cost + A(i,j) + Current reduced Cost

Reduced Cost = $27 + 4 + 3$

$= 34$



Scanned by CamScanner

(2) for i=1 & j=3.

ie.; (1,3)

$A(3,1) = \infty$

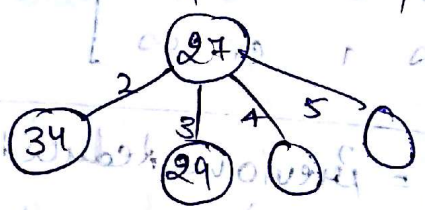
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 1 \\ \infty & 3 & \infty & 1 & 8 \\ 6 & 0 & \infty & \infty & 3 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 1 \\ \infty & 2 & \infty & 0 & 7 \\ 6 & 0 & \infty & \infty & 3 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix} \leftrightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 0 \\ \infty & 2 & \infty & 0 & 6 \\ 6 & 0 & \infty & \infty & 2 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix}$$

1

(3) Reduced cost = 24 + 0 + 2

= 29



(9) for i=1 & j=4

ie.; (1,4)

$A(4,1) = \infty$

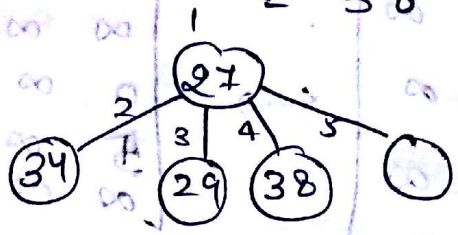
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ 10 & 6 & 1 & \infty & \infty \end{bmatrix} 1$$

$$= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ 9 & 5 & 0 & \infty & \infty \end{bmatrix} \leftrightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 0 \\ 0 & 3 & \infty & \infty & 7 \\ \infty & 0 & 2 & \infty & 2 \\ 9 & 5 & 0 & \infty & \infty \end{bmatrix}$$

1

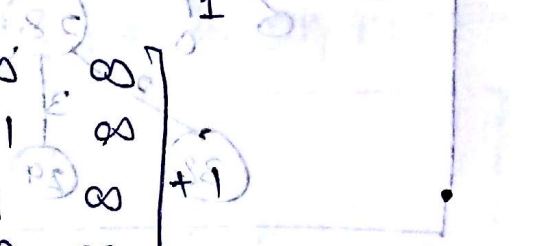
Scanned by CamScanner

③ Reduced cost = 27 + 9 + 2
= 38



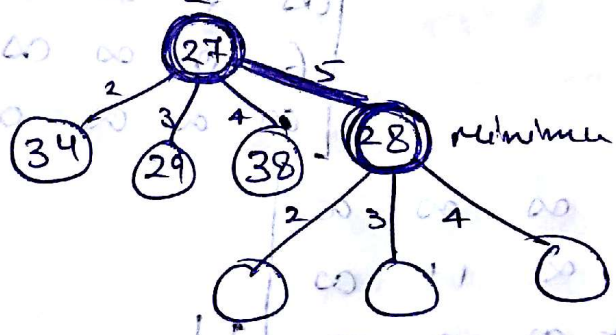
② for i = 1 & j = 5
i.e., (1, 5)
A(5,1) = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 2 & \infty & \infty \\ \infty & 6 & 1 & 0 & \infty \end{bmatrix}$$



$$= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 1 & \infty & \infty \\ \infty & 6 & 0 & 0 & \infty \end{bmatrix} + 1$$

③ Reduced cost = 27 + 0 + 1
= 28



② for i = 5 & j = 2, 3, 4

i.e., (5, 2)

A(2, 1) = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 11 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 6 & \infty & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 2 \\ \\ \\ 1 \\ \\ \end{matrix}$$

Matrix is made from the matrix which have reduced cost = 28. i.e., where i = 1, j = 5

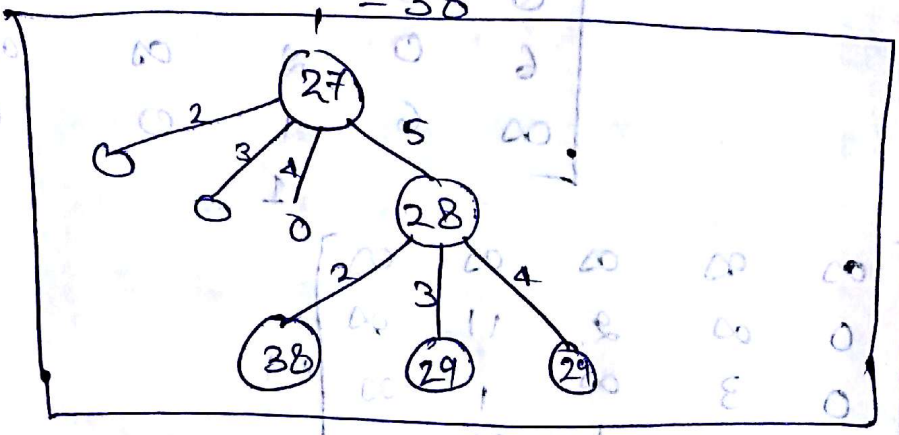
Scanned by CamScanner

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 9 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \longleftrightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 9 & \infty \\ 0 & \infty & \infty & 0 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

1

(3) Reduced cost = 28 + 6 + 4

= 38



(2) for i = 5 & j = 3

ie., (5,3)

$\Rightarrow A(3,1) = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & \infty & \infty & 1 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} 1$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & \infty & \infty & 0 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} +1$$

(3) Reduced cost = 28 + 0 + 1

= 29

Scanned by CamScanner

② for $i = 5$ & $j = 4$

i.e., (5,4)

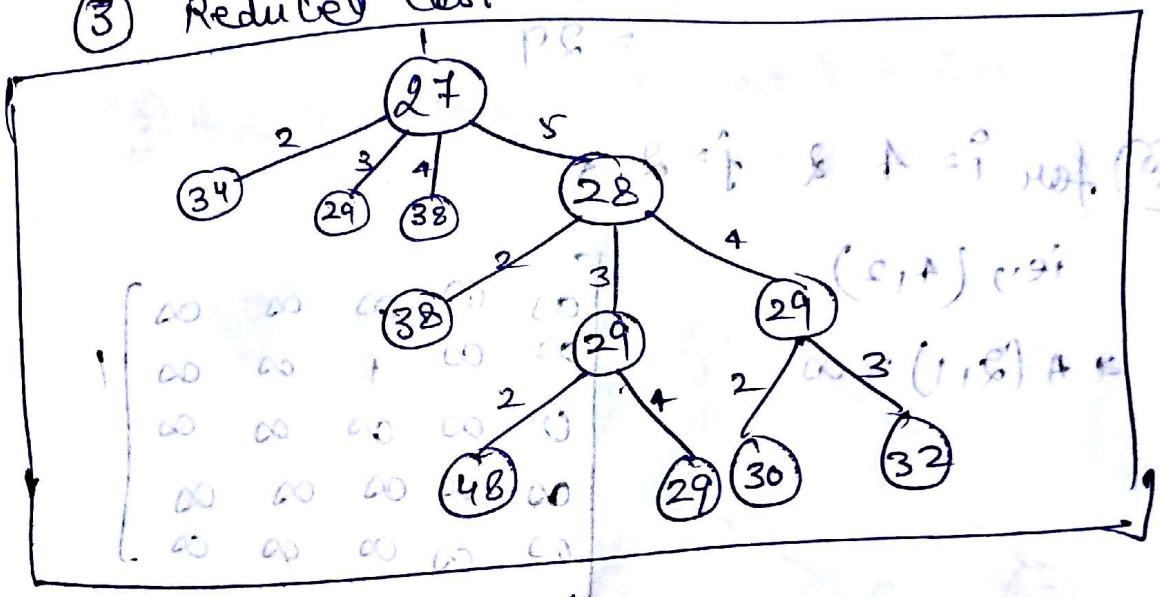
$\Rightarrow A(4,1) = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

1

$$\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 1 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} + 1$$

③ Reduced cost = 28 + 0 + 1 = 29



② for $i = 3$, $j = 2, 4$.

i.e., (3,2)

$A(2,1) = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 11 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 11 \\ \\ \\ 6 \\ \\ \end{matrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Scanned by CamScanner

(3) Reduced cost = 29 + 2 + 17

= 48

(2) for i = 3 & j = 4

i.e., (3, 4)

⇒ A(4,1) = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

(3) Reduced cost = 29 + 0 + 0

= 29

(2) for i = 4 & j = 2, 3

i.e., (4, 2)

⇒ A(2,1) = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 1 & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

(3) Reduced cost = 29 + 0 + 1

= 30

Scanned by CamScanner

(2) for i = 4 & j = 3.

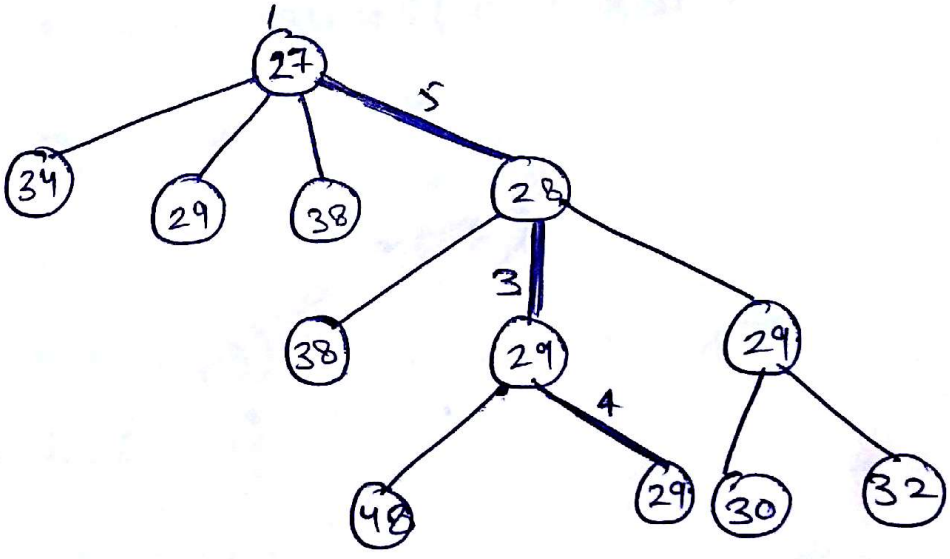
i.e., (4, 3)

$A(3,1) = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} 3$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

(3) Reduced cost = 29 + 0 + 3 = 32



⇒ $1 \to 5 \to 3 \to 4 \to 2 \to 1$ path

Scanned by CamScanner