Dynamic programming

It is used as method for solving certain kind of prob. & can be apply when the solution of the problem include sol^n of sub problem.

Dynamic programming solve each sub problem once & ~~stored~~ stored the result in a table so that it can be rapidly retrieve if needed again.

Dynamic programming used when a prob. break into small sub prob.

Property of dynamic programming

- It simplify sub problem
- It provide optimal sub structure of the problem
- each sol has a value associated in it.
- matrix chain Multiplication

In A matrix chain multiplication. there is a sequence of matrixed in a chain $A_1, A_2, A_3 - - A_n$ & we wish to compute the product of those matrices

Proof :- MATRIX CHAIN MULTIPLICATION (A, B)

(1) If A. column ≠ B. row
(2) error "can't multiply"
(3) else let c be a new matrix
A. row x B. column

```
for i = 1 to A. row
  for j = 1 to B. column
    Cij = 0
    for k = 1 to A column
      Cij = Cij + ai x * bkj
```

Matrix chain multiplication problem :- In matrix chain mul. there are n matrices ($A_1 A_2 - A_n$) where matrix A_i has size $P_{i-1} \times P_i$ Here our aim is fully parenthesize the product $A_1, A_2 - - A_n$ in a way the minimize the no. of salar multiplication

i.m. Imp

Computing Optimal Cost :-

```
MATRIX-CHAIN-ORDER (P)
(1)  n = p.length - 1
(2)  let m[1..n, 1..n] & s[1..n-1,
       2..n] be new table
(3)  for i = 1 to n
(4)      m[i, i] = 0
(5)  for l = 2 to n  // l is the chain length
(6)    for i = 1 to n-l+1
(7)      j = i+l-1
(8)    m[i, j] = ∞
(9)    for k = i to j-1
(10)     q = m[i, k] + m[k+1, j] +
           p_{i-1} p_k p_j
(11)   if  q < m[i, j]
(12)       m[i, j] = q
(13)       s[i, j] = k
(14) return m & s
```

2×3 3×6 6×4 4×5

✱ Matrix chain Multiplication :-

A_1	A_2	A_3	A_4	A_5	A_6
30×35	35×15	15×5	5×10	10×20	20×25
P_0	P_1	P_2	P_3	P_4	P_5 P_6

M =

i \ j	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

S =

i \ j	2	3	4	5	6
1	1	1	3	3	3
2		2	3	3	3
3			3	3	3
4				4	5
5					5

formula :-

$$\min_{i<k\le j} \begin{cases} 0 & i = j \\ \min_{i\le k\le j} \left[m[i,k] + m[k+1, j] + P_{i-1} P_k P_j \right] & i < j \end{cases}$$

$$\boxed{k = i \text{ to } j-1}$$

S Matrix me K ki value put karte rahenge first row to ko first column me jo value hogi

Date : / /etc—

Page No.:

m matrix m pahle diagonal fill karenge

Now $i = 1$ & $j = 2$

then $K = i$ to $j-1$

$K = 1$ to 1

$\boxed{K = 1}$

Using formula $m[i,j] = m[1,1] + m[2,2] + P_0 P_1 P_2$

$30 \times 35 \times 15$

$0 + 0 + 15750 = 15750$

Now $i = 2$ & $j = 3$

$\boxed{K = 2}$

$m[2,2] + m[3,3] + P_1 P_2 P_3$

$0 + 0 + 35 \times 15 \times 5 = 2625$

Now $i = 3$ & $j = 4$

$\boxed{K = 3}$

$m[3,3] + m[4,4] + P_2 P_3 P_4 = 15 \times 5 \times 10 = 750$

Now $i = 4$ & $j = 5$

$\boxed{K = 4}$

$m[4,4] + m[5,5] + P_3 P_4 P_5 = 5 \times 10 \times 20 = 1000$

Now $i = 5$ & $j = 6$

$\boxed{K = 5}$

$m[5,5] + m[6,6] + P_4 P_5 P_6 = 10 \times 20 \times 25 = 5000$

Date : __/__/____ Page No.: ____

first row ka third column = 5 - 1

= Now $i = 1$ & $j = 3$

$k = 1$ to 2

{then dono par check karenge 1 & 2 jiski value min ayegi uski value ko m matrix me put karenge or ji's k ki value par min ayegi us k ki value ko s matrix me put karenge}

for $k = 1$

$m[1,1] + m[2,3] + P_0 P_1 P_3$

$0 + 2625 + 30 \times 35 \times 5 = 7875$ (min valu

for $k = 2$

$m[1,2] + m[3,3] + P_0 P_2 P_3 = 15750 + 30 \times 15 \times 5$

$= 10000$

= Now $i = 2$ & $j = 4$

$k = 2$ to 3

for $k = 2$

$m[2,2] + m[3,4] + P_1 P_2 P_4$

$0 + 750 + 35 \times 15 \times 10 = 6000$

for $k = 3$

$m[2,3] + m[4,4] + P_1 P_3 P_4$

$2625 + 0 + 35 \times 5 \times 10 = 4375$

= Now $i = 3$ & $j = 5$

$k = 3$ to 4

Date : __/__/__ Page No.:

for k=3

$m[3,3] + m[4,5] + P_2 P_3 P_5$

$0 + 1000 + 15 \times 5 \times 20 = 2500$ ✓

for k=4

$m[3,4] + m[5,5] + P_2 P_4 P_5$

$750 + 0 + 15 \times 10 \times 20 = 3750$

Now i=4 & j=6

k = 4 to 5

for k=4

$m[4,4] + m[5,6] + P_3 P_4 P_6$

$0 + 5000 + 5 \times 10 \times 25 = 6,250$

for k=5

$m[4,5] + m[6,6] + P_3 P_5 P_6$

$1000 + 0 + 5 \times 20 \times 25 = 3,500$ ✓

Now i=1 & j=4

k = 1 to 3

for k=1

$m[1,1] + m[2,4] + P_0 P_1 P_4$

$0 + 4375 + 30 \times 35 \times 10 = 14,875$

for k=2

$m[1,2] + m[3,4] + P_0 P_2 P_4$

$1575 + 750 + 35 \times 15 \times 10 = 21,750$

Date: / / Page No.:

for k = 3

m[1,3] + m[4,4] + p0 p3 p4

7875 + 0 + 30 × 35 × 10 = 9375 ✓

Now i = 2 & j = 5

k = 2 to 4

for k = 2

m[2,2] + m[3,5] + p1 p2 p5

0 + 2500 + 35 × 15 × 20 = 13,000

for k = 3

m[2,3] + m[4,5] + p1 p3 p5

2625 + 1000 + 35 × 5 × 20 = 7125 ✓

for k = 4

m[2,4] + m[5,5] + p1 p4 p5

4375 + 0 + 35 × 10 × 20 = 11,375

Now i = 3 & j = 6

k = 3 to 5

for k = 3

m[3,3] + m[4,6] + p2 p3 p6

0 + 3500 + 15 × 5 × 25 = 5,375 ✓

for k = 4

m[3,4] + m[5,6] + p2 p4 p6

750 + 5000 + 15 × 10 × 25 = 9,500

Date : / /

Page No.:

for [k = 5]

$m[3,5] + m[6,6] + p_2 p_5 p_6$

$2500 + 0 + 15 \times 20 \times 25 = 10,000$

Now [i = 4 & j = 5]

[k = 1 to 4]

for [k = 1]

$m[4,4] + m[2,5] + p_0 p_1 p_5$

$0 + 7125 + 30 \times 35 \times 20 = 28,125$

for [k = 2]

$m[1,2] + m[3,5] + p_0 p_2 p_5$

$15750 + 2500 + 30 \times 15 \times 20 = 27,250$

for [k = 3]

$m[1,3] + m[4,5] + p_0 p_3 p_5$

$7875 + 1000 + 30 \times 5 \times 20 = 11,875$ ✓

for [k = 4]

$m[1,4] + m[5,5] + p_0 p_4 p_5$

$9375 + 0 + 30 \times 10 \times 20 = 15,375$

Now [i = 2 & j = 6]

[k = 2 to 5]

for [k = 2]

$m[2,2] + m[3,6] + p_1 p_2 p_6$

$0 + 5375 + 35 \times 15 \times 25 = 18,500$

Date : __/__/____ Page No.: ____

for k = 3

$m[2,3] + m[4,6] + p_1 p_3 p_6$

$262.5 + 3500 + 35 \times 5 \times 25 = 10,500$ ✓

for k = 4

$m[2,4] + m[5,6] + p_1 p_4 p_6$

$4375 + 5000 + 35 \times 10 \times 25 = 18,125$

for k = 5

$m[2,5] + m[6,6] + p_1 p_5 p_6$

$7125 + 0 + 35 \times 20 \times 25 = 24,625$

Now i = 1 & j = 6

k = 1 to 5

for k = 1

$m[1,1] + m[2,6] + p_0 p_1 p_6$

$0 + 10500 + 30 \times 35 \times 25 = 36,750$

for k = 2

$m[1,2] + m[3,6] + p_0 p_2 p_6$

$15750 + 5375 + 30 \times 15 \times 25 = 32,375$

for k = 3

$m[1,3] + m[4,6] + p_0 p_3 p_6$

$7875 + 3500 + 30 \times 5 \times 25 = 15,125$ ✓

for k = 4

$m[1,4] + m[5,6] + p_0 p_4 p_6$

$9375 + 5000 + 30 \times 10 \times 25 = 21,875$

Date :

for K=5

$m[1,5] + m[6,6] + P_0 P_5 P_6$

$11875 + 0 + 30 \times 20 \times 25 = 26,875$

Now $((A_1 (A_2 A_3))((A_4 A_5) A_6)$ Ans

bracket jo lgate hai vo S matrix ka dekhkar lgate hai

jaise 1 row me six/tti to $A_1 + A_2$ par lag jayega

Phir 2 + 3 to $A_2 + A_3$ par lag jayega + so

OPTIMAL SOLUTION :-

PRINT-OPTIMAL PARANTHESIS (s, i, j)

(1) if $i = j$

(2) then Print "A"

(3) else print "(" (Paranthesis)

(4) Print OPTIMAL PARANTHESIS (s, i, s[i,j])

(5) " " " (s, s[i,j] + 1, j)

(6) Print ")"

Date : __/__/____ Page No.: ____

Strassens Matrix Multiplication

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}_{2\times 2} \times B \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}_{2\times 2} = C \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$C_{11} = a_{11} * b_{11} + a_{12} * b_{21}$
$C_{12} = a_{11} * b_{12} + a_{12} * b_{22}$
$C_{21} = a_{21} * b_{11} + a_{22} * b_{21}$
$C_{22} = a_{21} + b_{12} + a_{22} * b_{22}$

$A = [a_{11}]$ $b = [b_{11}]$
$C = [a_{21} + b_{11}]$

If A & B has small dimension then OK if A & B has large dimension we have to divide the problem into sub problem.

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{31} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

A_{11}

$8 * (n/2) * n^2$

$\frac{4}{2} \times \frac{4}{2}$ $\frac{4}{2} \times \frac{4}{2}$

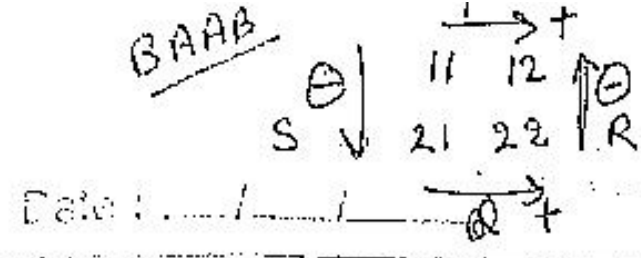
D2 C Algo

$C_{11} = A_{11} * B_{11} + A_{12} * B_{21}$
$C_{12} = A_{11} * B_{12} + A_{12} * B_{22}$
$C_{21} = A_{21} * B_{11} + A_{22} * B_{21}$
$C_{22} = a_{21} * B_{12} + A_{22} * B_{22}$

$O(n^3)$

```
Algo mm (A, B, n)
{ If (n <= 2) { c = 4 formulas }
else
{ mid = n/2                    matrix addition
mm(A11, B11, n/2) + mm(A12, B21, n/2)
mm(A11, B12, n/2) + mm(A12, B22, n/2)
mm(A21, B11, n/2) + mm(A22, B21, n/2)
mm(A21, B12, n/2) + mm(A22, B22, n/2)
```

BAAB



2×2 $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ 2×2 $\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

$P = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) \cdot B_{11}$

$R = A_{11} (B_{12} - B_{22})$

$S = A_{22} (B_{21} - B_{11})$

$T = (A_{11} + A_{12}) B_{22}$

$U = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$

$V = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

BAT (a)

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = B_{11} (A_{21} + A_{22})$

$R = A_{11} (B_{12} - B_{22})$

$S = A_{22} (B_{21} - B_{11})$

$T = B_{22} (A_{11} + A_{12})$

$U = (A_{21} - A_{11}) (B_{11} + B_{12})$

$V = (A_{12} - A_{22}) (B_{21} + B_{22})$

$A = \begin{bmatrix} 5 & 6 \\ -4 & 3 \end{bmatrix}$ $B = \begin{bmatrix} -7 & 6 \\ 5 & 9 \end{bmatrix}$

$P = 16$

$Q = 7$

$R = -15$

$S = 36$

$T = 9(5+6)$

$U = 9$

$V = 42$

$C_{11} = 16 + 36 - 99 + 42 = -5$

$C_{12} = -15 + 99 = 84$

$C_{21} = 7 + 36 = 43$

$C_{22} = 16 - 15 - 7 + 9 = 3$

Date : 23/Sept/2018 Page No.:

★ LCS - Length (X, Y) {Longest Common Subsequence}

(1) m · length (X)
(2) n · length (Y)
(3) for i = 1 to m c[i, 0] = 0
(4) for j = 1 to n c[0, j] = 0
(5) for i = 1 to m
(6) for j = 1 to n
(7) if (Xi == Yj)
(8) c[i, j] = c[i-1, j-1] + 1
(9) else c[i, j] = max (c[i-1, j], c[i, j-1]
(10) Return c

$$c[i,j] = \begin{cases} c[i-1, j-1] + 1 & \text{if } x[i] = y[j] \\ \max [c[i, j-1], c[i-1, j]] & \text{otherwise} \end{cases}$$

String

Ques-1 X = ABCBB
Y = BDCAB
m · length (X) = m = 4
n · length (Y) = n = 5

(vertical) X			B	D	C	A	B
		0	1	2	3	4	5
(horizontal) X	0	0	0	0	0	0	0
A	1	0	0	0	0	1	1
B	2	0	1	1	1	1	2
C	3	0	1	1	2	2	2
B	4	0	1	1	2	2	3

Date : __/__/__

for i = 1 to m
    j = 1 to 4
for i = 1 to m c[i,0] = 0
    i = 1 to 4 c[i,0] = 0
c[1,0], c[2,0], c[3,0], c[4,0] = 0

for j = 1 to n
for y = 1 to 5
for i = 1 to n c[0,j] = 0
    i = 1 to 5 c[0,j] = 0
c[0,1], c[0,2], c[0,3], c[0,4], c[0,5] = 0

Now ° if (xi == yj)
        (A == B) not equal
else i = 1 & j = 1
else c[i,j] = max(c[i-1,j], c[i,j-1])
c[1,1] = max[c[1-1,1], c[1,1-1]]
c[1,1] = max[c[0,1], c[1,0]]
       = max(c[0], c[0])
c[1,1] = 0 ← max value
              late hai

Now ° if (xi == yj)
        (A == D) not equal
    i = 1 & j = 2
else c[i,j] = max[c[i-1,j], c[i,j-1]]
c[1,2] = max(c[0,2], c[1,1])
c[1,2] = max(c[0], c[0])
c[1,2] = 0

Now (A == C) not equal
i = 1 & j = 3
c[i,j] = max[c[i-1, j], c[i, j-1])
c[1,3] = max[c[0,3], c[1,2])
= max(c[0], c[0])
c[1,3] = 0

Now (A == A) equal i = 1, & j = 4
c[i,j] = c[i-1, j-1]+1
c[1,4] = c[0, 3] +1
c[1,4] = 0+1 = 1

Now (A == B) not equal i = 1 & j = 5
c[i,j] = max(c[i-1, j], c[i, j-1])
c[1,5] = max(c[0,5], c[1,4])
= max(c[0], c[1])
c[1,5] = 1 {max value lete hai}

Now (B == B) equal i = 2, & j = 1
c[i,j] = c[i-1, j-1]+1
c[2,1] = c[1, 0]+1
c[2,1] = 0+1 = 1

Now (B == D) not equal i = 2 & j = 2
c[i,j] = max(c[i-1, j], c[i, j-1])
c[2,2] = max(c[1,2], c[2,1])
c[2,2] = max(c[0], c[1])
c[2,2] = 1

Now (B == C) not equal i = 2, & j = 3

c[i, j] = max (c[i-1, j], c[i, j-1])

c[2, 3] = max (c[1, 3], c[2, 2])

c[2, 3] = max (c[0], c[1])

c[2, 3] = 1

Now (B == A) not equal i = 2, & j = 4

c[i, j] = max (c[i-1, j], c[i, j-1])

c[2, 4] = max (c[1, 4], c[2, 3])

c[2, 4] = max (c[1], c[1])

c[2, 4] = 1

Now (B == B) equal i = 2, & j = 5

~~c[i, j] = max (c[i-1,~~

c[i, j] = c[i-1, j-1] + 1

c[2, 5] = c[i, 4] + 1

c[2, 5] = 1 + 1 = 2

Now (C == B) not equal i = 3, & j = 1

c[i, j] = max (c[i-1, j], c[i, j-1])

c[3, 1] = max (c[2, 1], c[3, 0])

c[3, 1] = max (c[1], c[0])

c[3, 1] = 1

Now (C == D) not equal i = 3, & j = 2

c[i, j] = max (c[i-1, j], c[i, j-1])

c[3, 2] = max (c[2, 2], c[2, 1])

c[3, 2] = max (c[1], c[1])

c[3, 2] = 1

Now (C == C) equal i = 3 & j = 3

c[i, j] = c[i-1, j-1] + 1

c[3, 3] = c[2, 2] + 1

c[3, 3] = 1 + 1 = 2

Now (C == A) not equal i = 3, & j = 4

c[i, j] = max(c[i-1, j], c[i, j-1])

c[3, 4] = max(c[2, 4], c[2, 3])

c[3, 4] = max(c[1], c[2])

c[3, 4] = 2

Now (C == B) not equal i = 3, & j = 5

c[i, j] = max(c[i-1, j], c[i, j-1])

c[3, 5] = max(c[2, 5], c[3, 4])

c[3, 5] = max(c[2], c[2])

c[3, 5] = 2

Now (B == B) equal i = 4, & j = 1

c[i, j] = c[i-1, j-1] + 1

c[i, j] = c[3, 0] + 1

c[4, 1] = 0 + 1 = 1

Now (B == D) Not equal i = 4, & j = 2

c[i, j] = max(c[i-1, j], c[i, j-1])

c[4, 2] = max(c[3, 2], c[4, 1])

c[4, 2] = max(c[1], c[1])

c[4, 2] = 1

Now (B == C) not equal $i = 4$, & $j = 3$

$c[i,j] = \max(c[i-1,j], c[i,j-1])$

$c[4,3] = \max(c[3,3], c[4,2])$

$c[4,3] = \max(c[2], c[1])$

$c[4,3] = 2$

Now (B == A) not equal $i = 4$, & $j = 4$

$c[i,j] = \max(c[i-1,j], c[i,j-1])$

$c[4,4] = \max(c[3,4], c[4,3])$

$c[4,4] = \max[c[2], c[2])$

$c[4,4] = 2$

Now (B == B) equal $i = 4$, $j = 5$

$c[i,j] = c[i-1, j-1] + 1$

$c[4,5] = c[3,4] + 1$

$c[4,5] = 2 + 1 = 3$ Ans

Construction an LCS:

(1) Print_LCS (b, X, i, j)
// b is the table used for LCS
(2) If i = 0 or j = 0
(3) then return
(4) If b[i, j] = "↖"
(5) then print_LCS (b, X, i-1, j-1)
(6) Print X.i
(7) Else If b[i, j] = "↑"
(8) then print_LCS (b; X, i-1, j)
(9) Else print_LCS (b, X, i, j-1)

		Q	B 1	D 2	C 3	A 4	B 5
	0	0	0	0	0	0	0
A	1	0	0	0	0	1	1
B	2	0	1	1	1	1	2
C	3	0	1	1	2	2	2
B	4	0	1	1	2	2	3

Pattern BCB Ans
hum niche se check karenge
i.e B - B same = ↑
agar different = ←

Check B - B
" B - A C - A
C - C
B - D
B - B

Page No.:

Date : / /

☆ Knapsack Problem :-

Knapsack problem is divided into two parts

- 0/1 (Dynamic Programming)
- fractional (or A) Greedy approach

There are some given item and we have to pack the knapsack to get the maximize total value each item has some weight & benifit value.
Total weight that we can carry should not more than the fix no. W (weight of knapsack).

In 0/1 knapsack problem we can carry only fixed weight value while in fractional knapsack we can divide the weight value of the item.

Ex: items weight & benifit value

given

w_i	b_i
2	3
3	4
4	5
5	8
9	10

w = weight of item
W = (knapsack weight)

given W = 20

we take the value which gives more benefit suppose we take weight -

Date: __/__/____ Page No.: ____

$2+3+4+5 = 3+4+5+8$

$\therefore 14 = 20$

again weight -

$2+4+5+9 = 3+5+8+10$

$20 = 26$ → more than weight

// equal to knapsack

★ 0-1 knapsack algorithm :-

For $w = 0$ to W

$B[0, w] = 0$

for $i = 0$ to n

$B[i, 0] = 0$

for $w = 0$ to W

if $wi <= w$

if $bi + B[i-1, W-wi] > B[i-1, w]$

$B[i, w] = bi + B[i-1, W-wi]$

else

$B[i, w] = B[i-1, w]$

else

$B[i, w] = B[i-1, W]$ // $wi > W$

Ques

i	wt	Bi
1	2	3
2	3	4
3	4	5
4	5	6

Ques- Solve. i

W \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	3	3	3	3
3	0	3	4	4	4
4	0	3	4	5	5
5	0	3	7	7	7

i = column

w = row

n = 4

W = 5 bcz at least weight given ques 5

Date : __/__/____ Page No.: _

0,1 Knapsack (V, w, n, W)

```
for w = 0 to W do
    C[0, w] = 0
for i = 1 to n do
    C[i, 0] = 0
    for w 1 to W do
        If wi < w then
            If [vi + C[i-1, w-wi]] then
                C[i, w] = vi + C[i-1, w-wi]
            else C[i, w] = C[i-1, w].
        else
            C[i, w] = C[i-1, w]
```

Date : __/__/____ Page No.: ____

★ Fractional knapsack:- Consider optimal weight of knapsack is W if we will remove weight w. If we will remove of item I the remaining weight is the optimal weight (W-w)

To solve the fractional problem - we

step1:- compute the value per pound = v_i/w_i

step2:- we can weight as much as possible items with the greatest value per pound.

step3:- If the supply of that item is exhausted we can carry more as much as possible of the item with the next value per pound. Greedy algo takes O(n log n) time

for eg:-

	w	v	P_i
I_1	5	30	6
I_2	10	20	2
I_3	20	100	5
I_4	30	90	3
I_5	40	160	4

W = 60

find value per pound

Solve:- $P_i = v_i/w_i$

6

2

5

3

4

Arrange the value of P_i in decreasing order

Date:

	w	v	Pi
I_1	5	30	6
I_3	20	100	5
I_5	40	160	4
I_4	30	90	3
I_2	10	20	2

$5+20+40 > 60$

then we take fractional part of 40

$5+20+35 = 60$

↓ weight

Benefit of 35 $\Rightarrow \left[\frac{160}{40} \times 35\right] = 140$

Benifit $= 30+100+140 = 270$

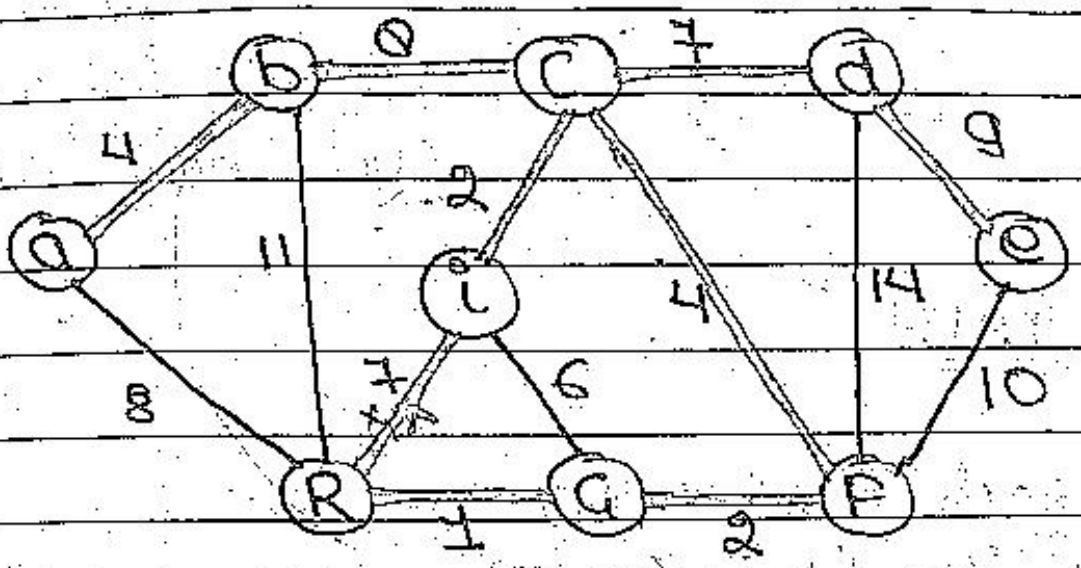
MST {Minimum Spanning Tree}:

MST-KRUSKAL (G, w)

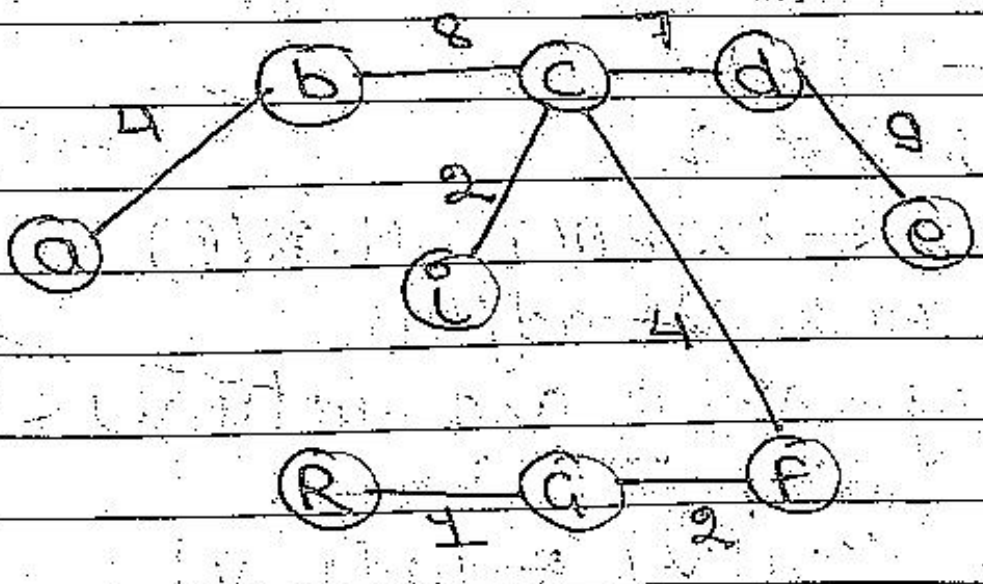
1) $A \leftarrow \phi$
2) for each vertex $v \in V(G)$
3) do make SET (v)
4) Sort the edges E of G into non decreasing order by weight.
5) for each edge $(u,v) \in E$ take in non decreasing order by weight.
6) do if find (u) $\neq$ find-SET(v)
7) then $A \leftarrow A \cup \{(u,v)\}$
8) Union (u,v)
9) Return A

we can't draw cycle.

1, 2, 2, 4, 4, 6, 7, 8, 9, 10, 11, 14

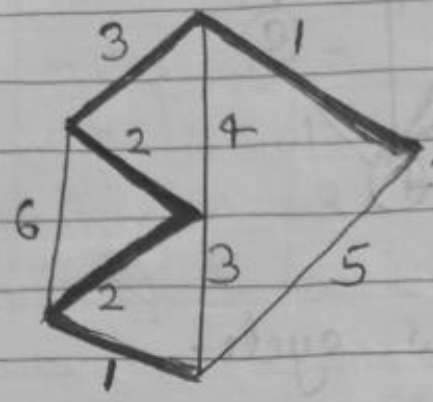


The tree is,



weight = 37

Q-

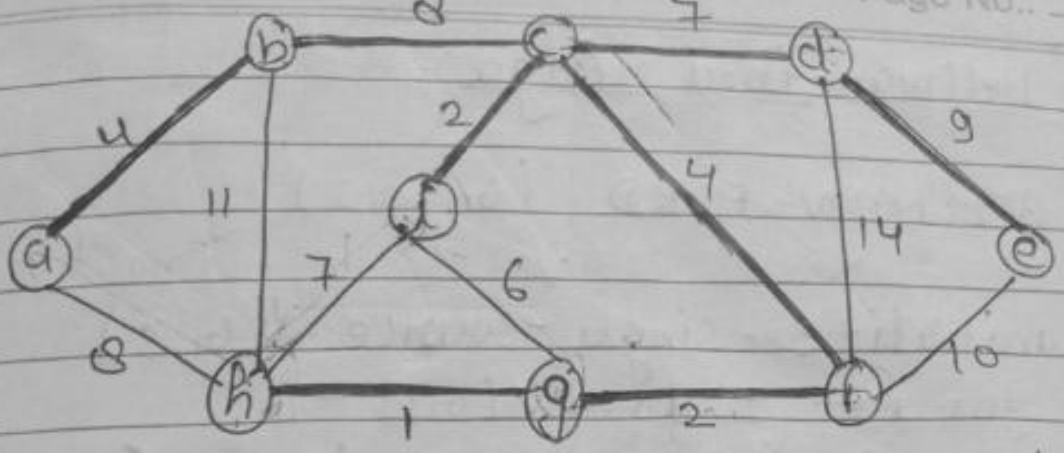


for Prims

usi node se dhekte hai jo sabse pahle minimum aata hai aur usme cycle nahi banni chahiye.

MST-PRIM:-

```
1)  For each u ∈ V(G)
2)      do key[u] ← ∞
3)          π[u] ← NIL
4)  key[r] ← 0
5)  Q ← V[G]
6)  while Q ≠ ϕ
7)  do u ← EXTRACT_MIN(Q)
8)  for each v ∈ adj(u)
9)  do if  v ∈ Q and w(u,v) < key(v)
10)         then π[v] ← u
11)             key[v] ← w[u,v]
```



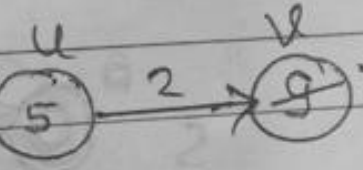
we can't draw a cycle.
weight = 35

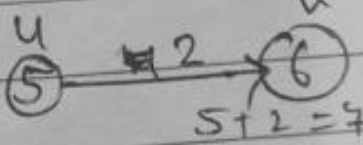
a-b, b-c, c-i, c-f, f-g, g-h, c-d, d-e

Single Source Shortest Path Algo:-

- Bellman ford
- Dijkastra algo

jab hum u aur weight Relax:- ko add karenge agar v se kam aaya add karke aur v jyada aaya to v ke value ko replace kar denge add ki hui value put kar denge agar u aur weight ki value jyada aayi v se to as it is rahne dega





Bellman

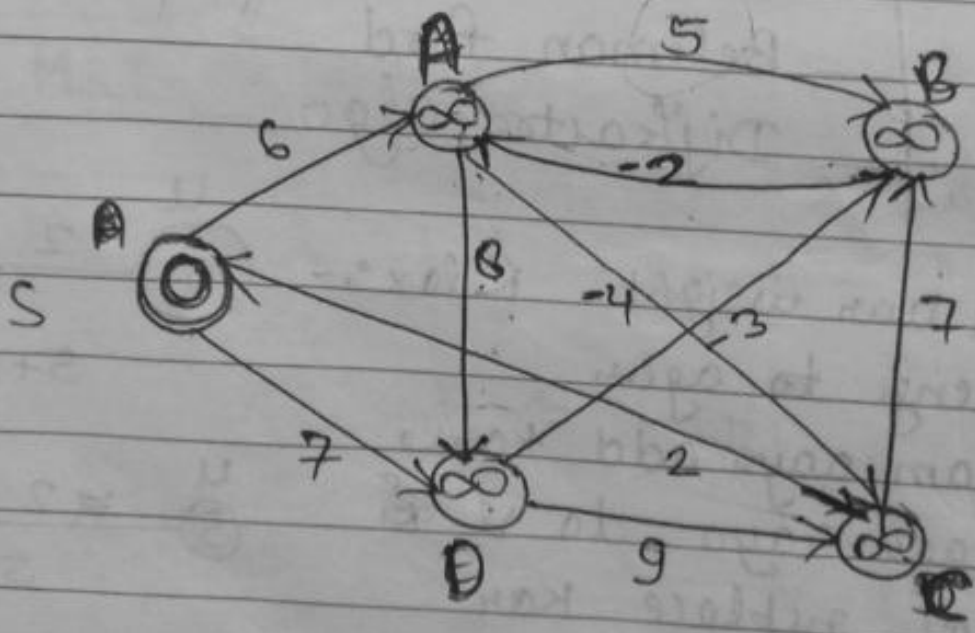
Date : __/__/____ Page No

Bellman-ford Algo :-

BELLMAN-FORD (G, w, s)

1) initialize-single-source (G, s)
2) for i ← 1 to [V[G]] - 1
3) do for each edge (u, v) ∈ E(G)
4) do RELAX (u, v, w)
5) for each edge (u, v) ∈ E(G)
6) do if d[v] > d[u] + w[u, v]
7) then return false
8) Return True.

Ques-



Solve :-

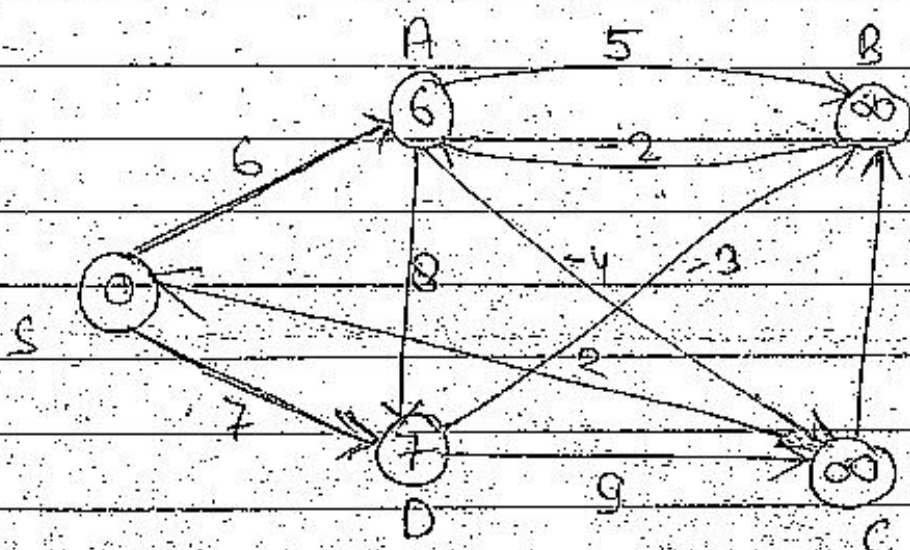
s = so

Solve:

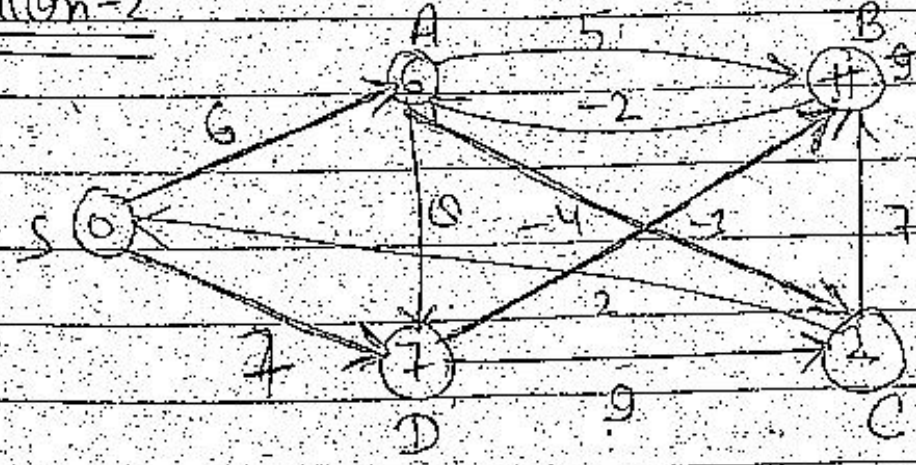
Iteration = V − 1
= 5 − 1 = 4

A − B = 5 ✓
A − C = −4
A − D = 6
B − A = −2
C − B = 7
C − S = 2
D − C = 9
D − B = −3 ✓
C − A = 6
S − B = 7

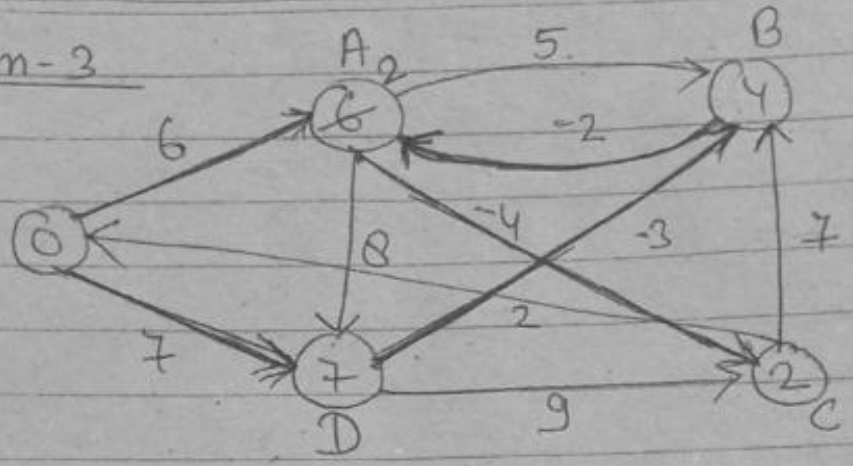
Iteration − 1



Iteration − 2

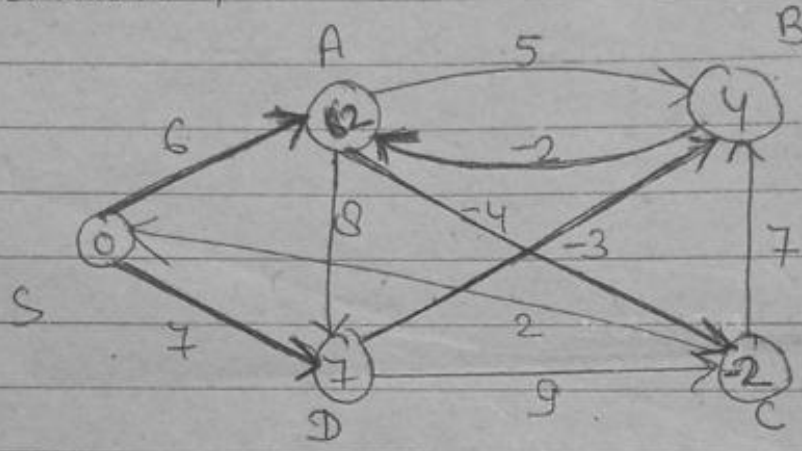


Iteration-3



6-4 = (2)

Iteration-4



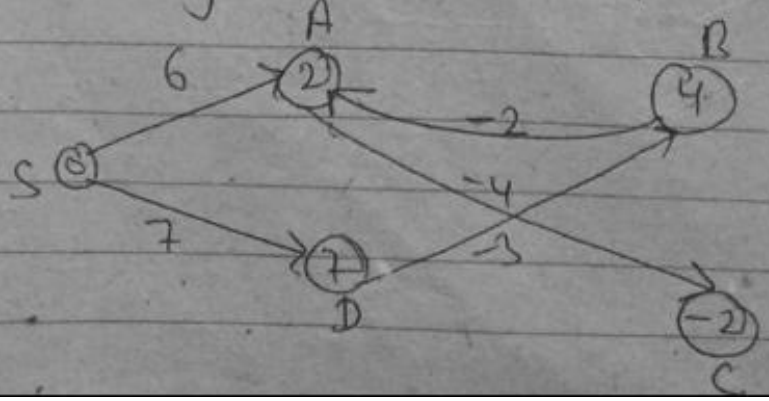
A-C
2-4 = -2

Ans

	S	A	B	C	D
0	0	∞	∞	∞	∞
1	0	6	∞	∞	7
2	0	6	4	2	7
3	0	6	4	2	7
4	0	2	4	-2	7

Ans

finally the shortest path



Ans

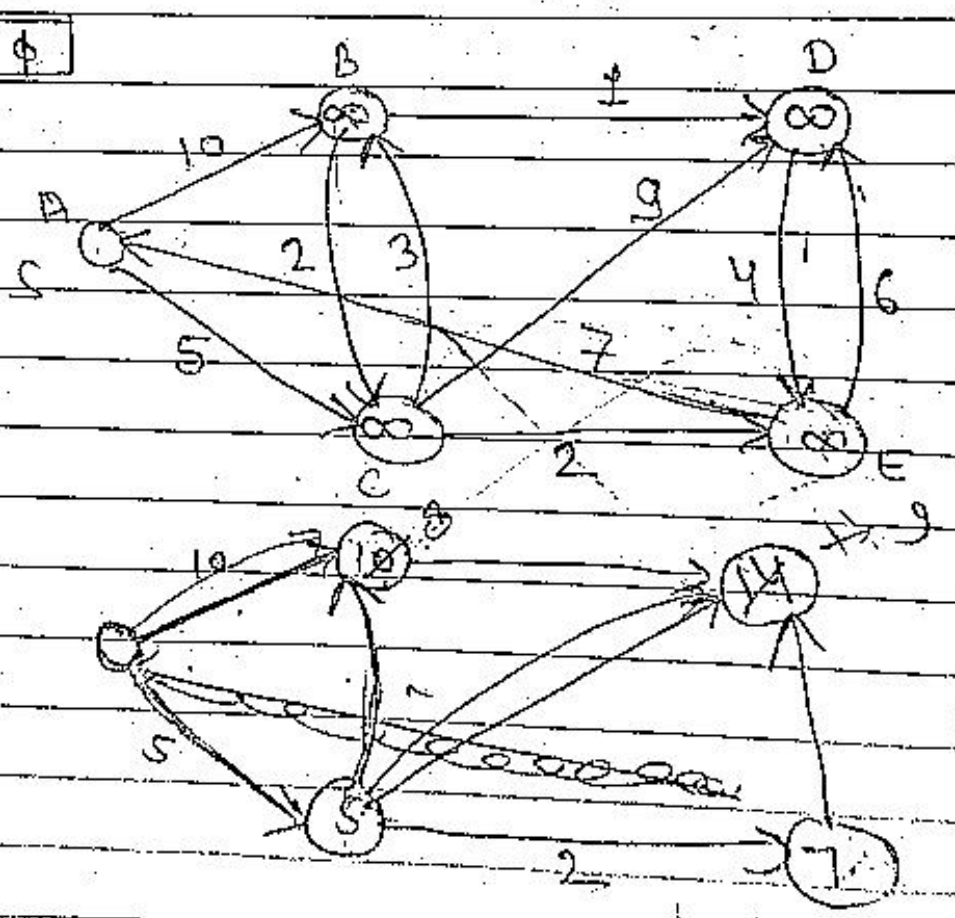
★ Dijkstra's algo :-

shortest path from one node to another node in which weight is non-negative

```
Dijkstra (G)
for each v ∈ V
    d(v) = ∞
d[s] = 0;  S = ϕ,  Q = V
while (Q ≠ ϕ)
    u = ExtractMin (Q)
    S = S ∪ {u}
    for each v ∈ u → adj[ ]
        if (d[v] > d[u] + w[u,v])      ] Relax
            d[v] = d[u] + w[u,v]
```

$S = \phi$

Queue-

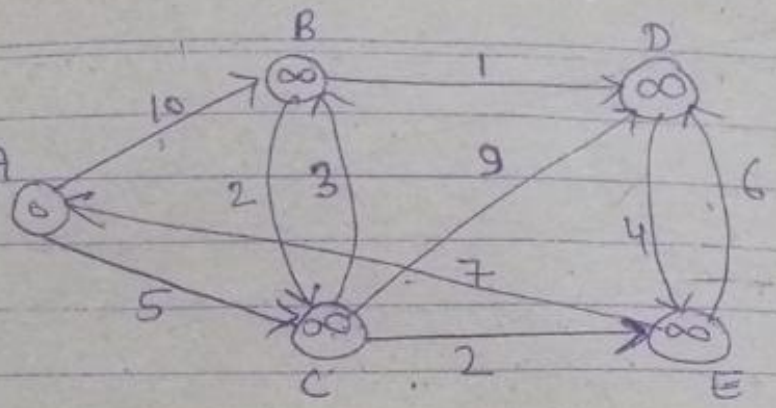


(ii)

dark-blue
light-blue

ab isme se smallest se dhekhenge means click do 5

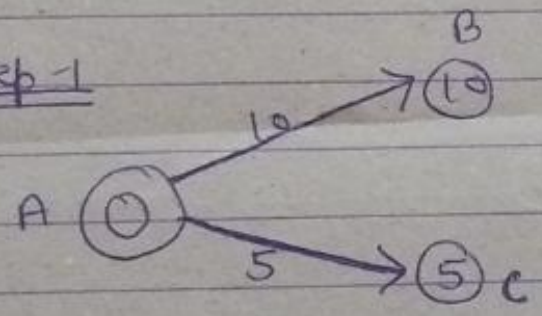
Q



A	B	C	D	E
0	∞	∞	∞	∞
0	10	5	∞	∞
0	8	5	14	7
0	8	5	13	7
0	8	5	9	7

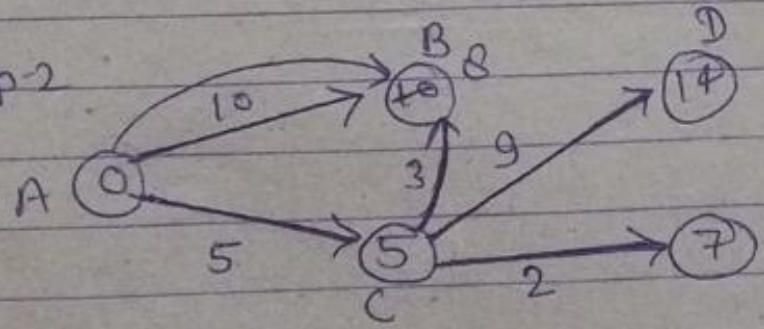
→ Puri row me sabse smallest then use mark de mark check ko badd use check kro

step-1



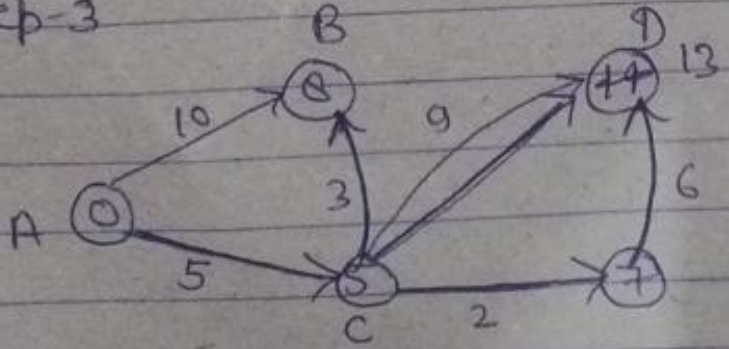
check smallest se jaisi is fig me 5 to 5 se check

Step-2

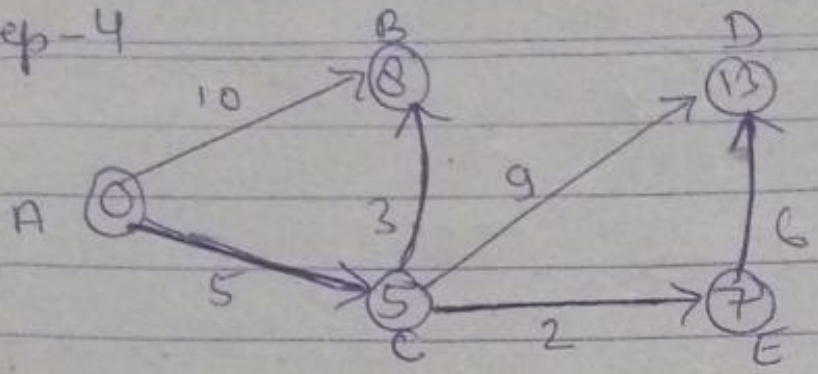


Check 7

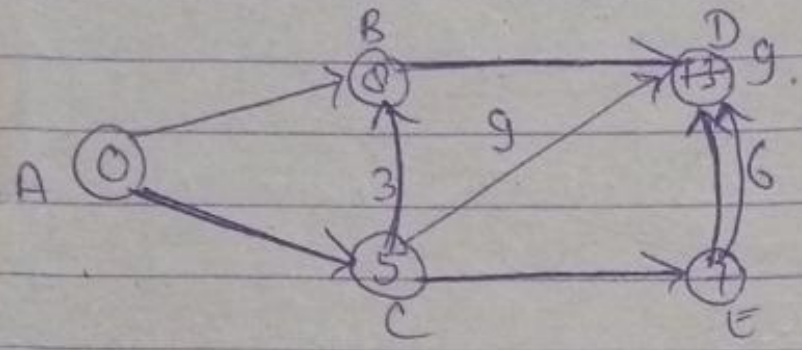
Step-3



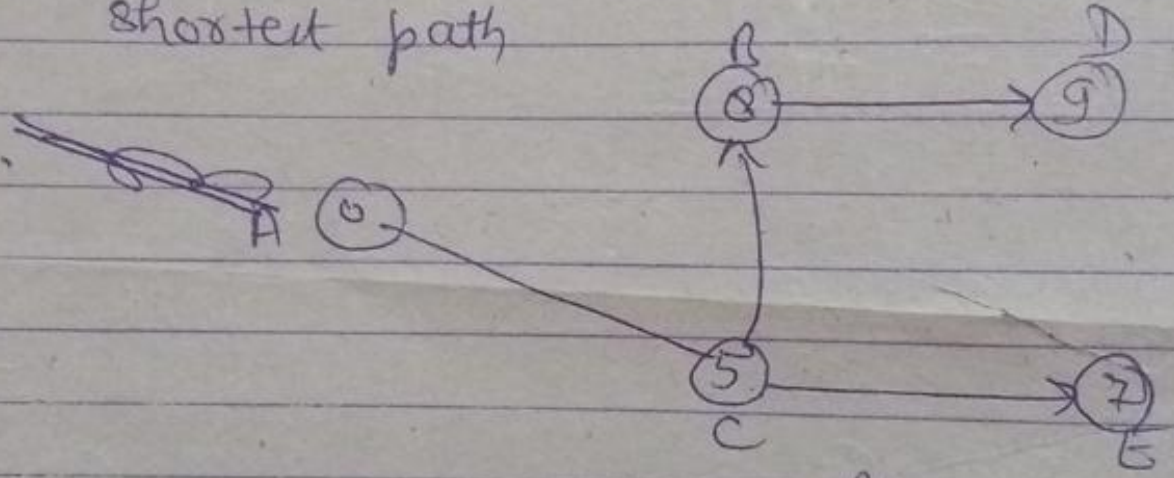
Step-4



Check-8



Shortest path

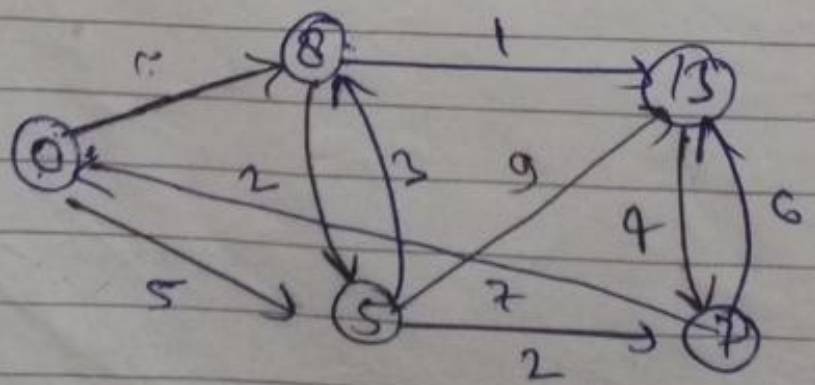


Ans

Date : __/__/____ Page No.:

* Diff. b/w Prims & Krushkal algo:-

Prims	Krushkal
① select any vertex as source.	① select shortest edge in the N/w.
② select the shortest edge connected to that vertex.	② select the next shortest edge which doesn't create a cycle.
③ Prims Select the shortest edge connected to any vertex that may be already connected & repeat step 3 until all vertex have been connected	③ Repeat step 2 untill all vertex have been connected.
④ Prims algo have been time complexity order of $O(V^2)$	④ Krushkal algo have time complexity order of $O(\log V)$



★ Bellman-ford algo	Dijkstra
① Bellman-ford is a single shortest path algo which allow (-ve) edge & can detect (-ve) cycle in a graph.	① It is also single source shortest path algo it doesn't allow (-ve) edge.
② Bellman-ford algo node contain only the information that are related to	② In dijkstra algo whole information about the N/w
① It can be easily implemented & distributed n/w.	③ Dijkstra can't

★ All pairs shortest path algo :-

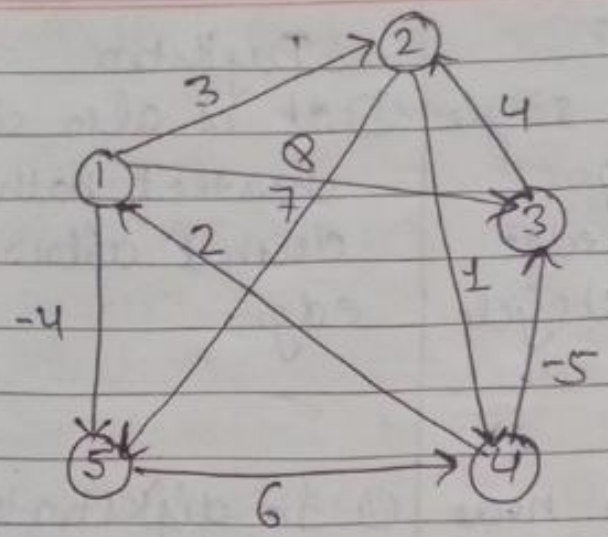
FLOYD - WARSHAL (W)

1) $n \leftarrow$ rows [W]
2) $D^{(0)} \leftarrow W$
3) for $K \leftarrow 1$ to n
4) do for $i \leftarrow 1$ to n
5) do for $j \leftarrow 1$ to n
6) do $d_{ij}^{(K)} \leftarrow \min(d_{ij}^{(K-1)}, d_{ik}^{(K-1)} + d_{kj}^{(K-1)})$
return $D^{(n)}$

In case $D^{(0)}$:- $d_{ij}^{(0)}$ $\begin{cases} \text{if } i = j = 0 \\ \text{if } i \neq j \text{ & No connection for } i \to j = \infty \end{cases}$

Date : __/__/____ Page No.: ____

Q-



1→1 = 0
1→2 ≠3

Solve :-

$$D^{(0)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{array}$$

~~For $\Pi^{(0)}$~~

$$\pi_{ij} \quad k=0 \begin{bmatrix} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty \end{bmatrix}$$

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

Date : __/__/____ Page No.: ____

$\pi^{(0)}$ =

	1	2	3	4	5
1	NIL	1	1	NIL	1
2	NIL	NIL	NIL	2	2
3	NIL	3	NIL	NIL	NIL
4	4	NIL	4	NIL	NIL
5	NIL	NIL	NIL	5	NIL

$d_{ij} = \min(d_{ij}, d_{ir} + d_{rj})$

See $D^{(0)}$ for $D^{(1)}$ matrix

$D^{(1)}$ =

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

$i=1, j=1, k=1$

$d_{11} \leftarrow \min(d_{11}, d_{11} + d_{11})$

$\min(0, 0+0)$

0

$i=1, j=2, k=1$

$d_{12} \leftarrow \min(d_{12}, d_{11} + d_{12})$

$d_{12} \leftarrow \min(3, 0+3)$

$d_{12} \leftarrow 3$

$i=1, j=3, k=1$

$d_{13} \leftarrow \min(d_{13}, d_{11} + d_{13})$

$d_{13} \leftarrow \min(\infty, 0+\infty)$

$d_{13} \leftarrow \infty$

$d_{14} \leftarrow \min(d_{14}, d_{11} + d_{14})$ $d_{14} \leftarrow \min(\infty, 0+\infty) = \infty$

$d_{15} \leftarrow \min(d_{15}, d_{11} + d$

$D^{(2)}$ solve

$D^{(3)}$ "

$D^{(4)}$ "

$D^{(5)}$ "

Date : / / Page No.:

	1	2	3	4	5
$\pi^{(1)}$ = 1	NIL	1	1	NIL	1
2	NIL	NIL	NIL	2	2
3	NIL	3	NIL	NIL	NIL
4	2	1	4	NIL	1
5	NIL	NIL	NIL	NIL	NIL

Do & D_1 are the same rows same column so write $\pi^{(1)}$ values are same.

$\pi_{42} = \pi_2$

5 to 5 min

$D^{(5)}$	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\pi^{(5)}$	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

Via through

In N-Queen problem the idea is to place queens one by one in different columns, starting from the left most column.

When we place a queen in column, we check for clashes with already placed queens.

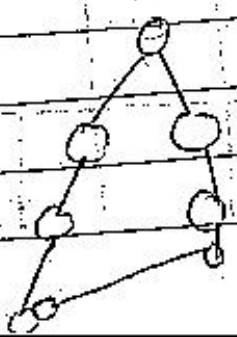
- Starting from the left most column.
- If all queens are placed return true.
- Try all rows in the current column.

* Backtracking :- In backtracking method, the desire solution is expressible has an n tuple $x_1, x_2, x_3 \cdots x_n$ where x_i is selected from some finite set S_i.
The basic idea of backtracking is to build a vector, one component at a time and to test whether the vector being ~~for~~ form have any chance of success.
Backtracking determine the solution by systematically solution space, set of all possible sol^n for the given problem.
Backtracking is a depth first search with some bounding function.

Examples:-

(i) N-Queen Problem:-
(ii) Hamiltonian cycle.
(iii) Sum of subset Problem
(iv) Graph coloring Problem
(v) Travelling Salesman problem.

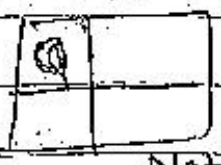
* Branch & Bound :- To find the multiple solution of a problem.



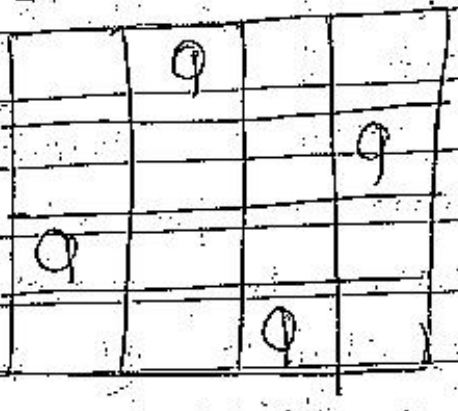
Date : __/__/__ Page No.:

★ N-Queen Problem :-

It can't place the 2 Q vertical, horizontal & diagonal



Ex- we use back track one soln is wrong then it is correct soln



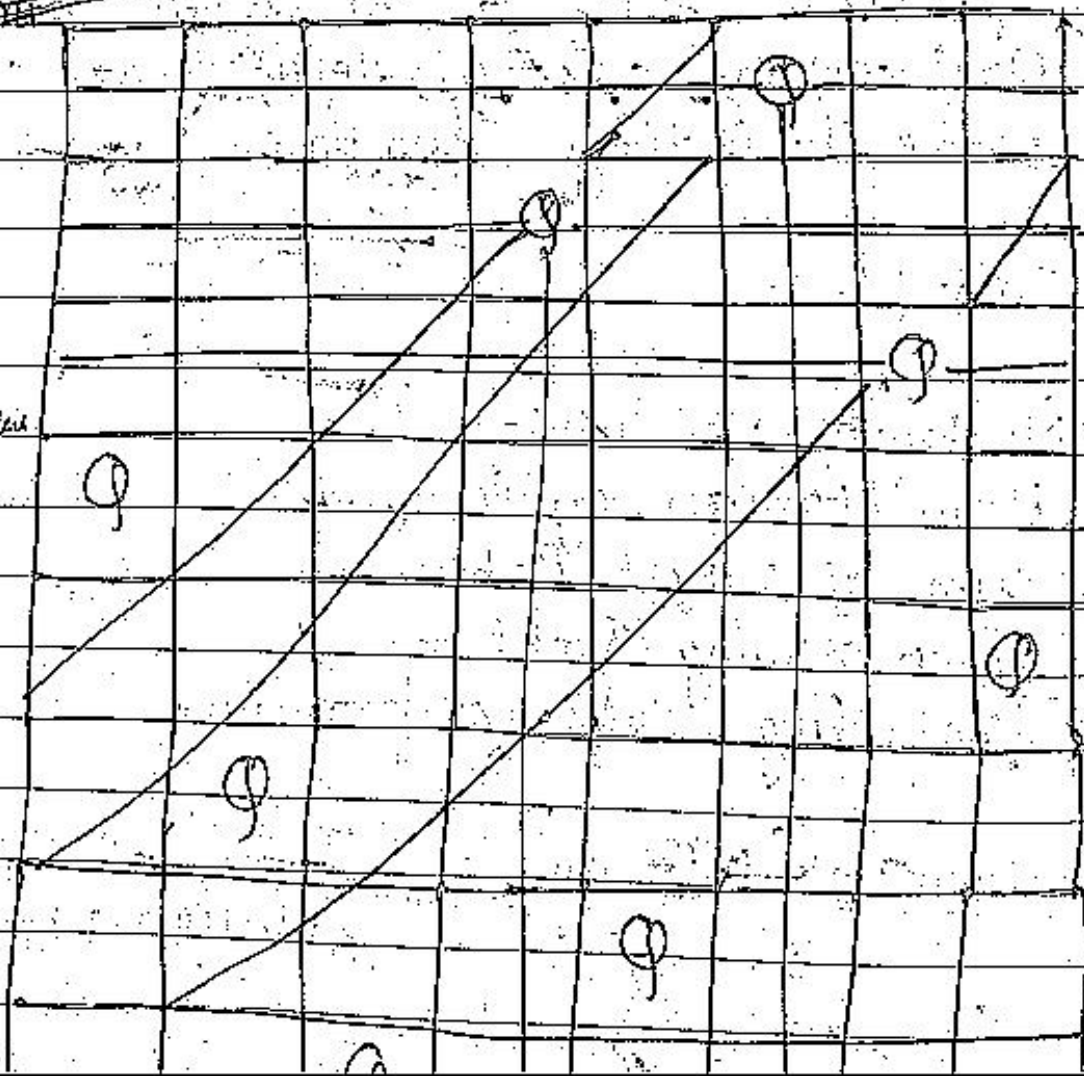
It place the 4 queens 4x4 This is a solution.

It is write the 8x8 soln

Graph

Hamil

6, 4, 7, 1, 8, 2, 5, 3



It place the 8 queen

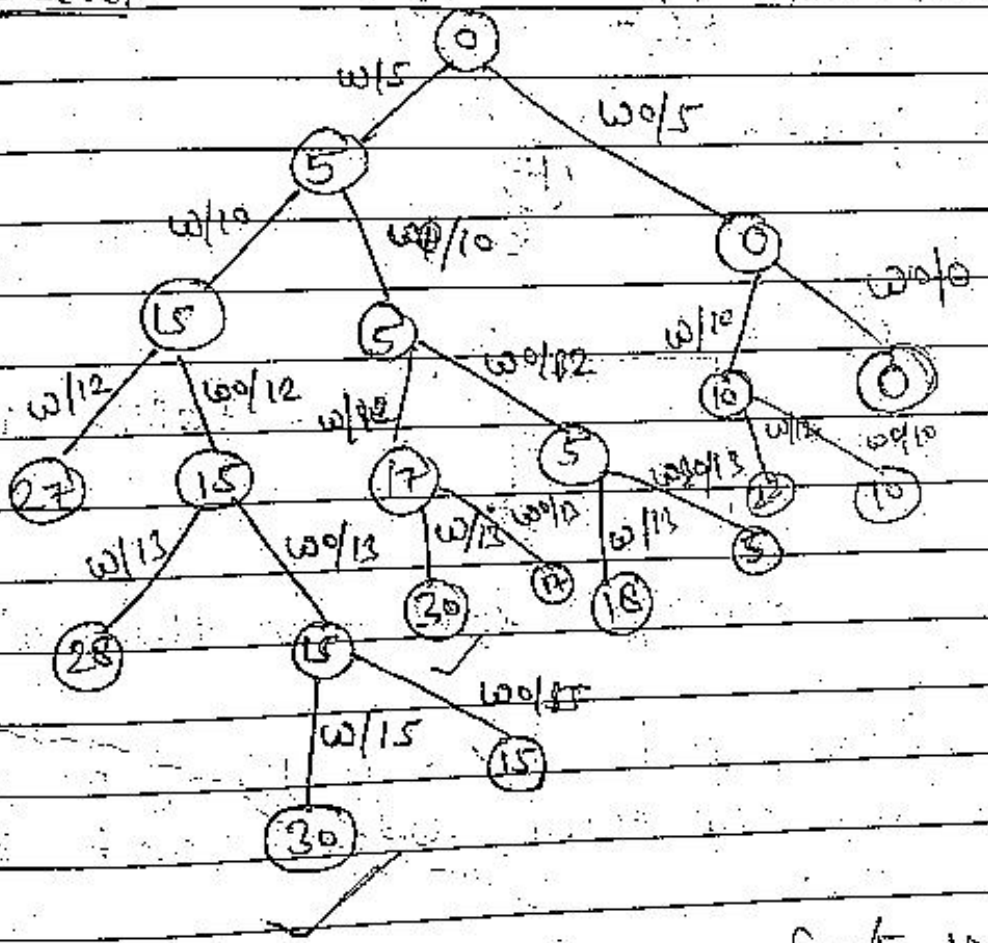
Sum of Subset Problem :- Let S be a set of numbers $(S_1, S_2, S_3 \cdots S_n)$ and we have to find a subset whose sum is equal to a given positive integer. The problem is solved by branch & bound problem but there is a multiple solution

Eg: Consider a set $S = (5, 10, 12, 13, 15)$ and $d = 30$ (positive integer)

w/s → with add 5

wo/s → without adding 5

Solve:-



S = (5, 10, 12, 13, 15)

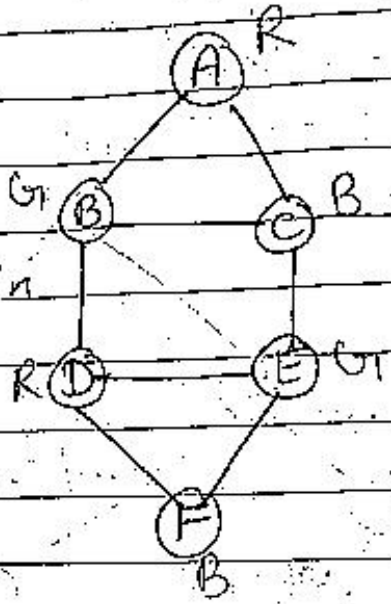
5 + 10 + 15 → 1st

12 + 13 + 5 → 2nd

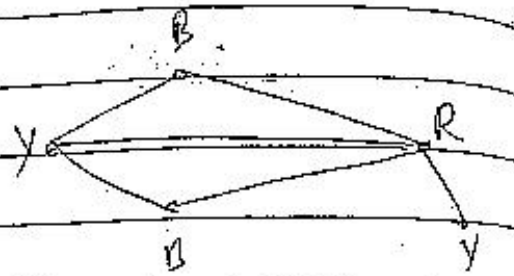
★ Graph coloring :-

Red, Green, blue color are given
· do color edge par ek sath nahi
aa sakte.

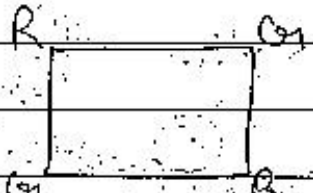
In this problem ①
it is to assign colours
to certain elements of
a graph subject to certain
constraints. vertex color
is the most common
graph coloring problem.



Solve to R G B



Properly coloured graph

② 

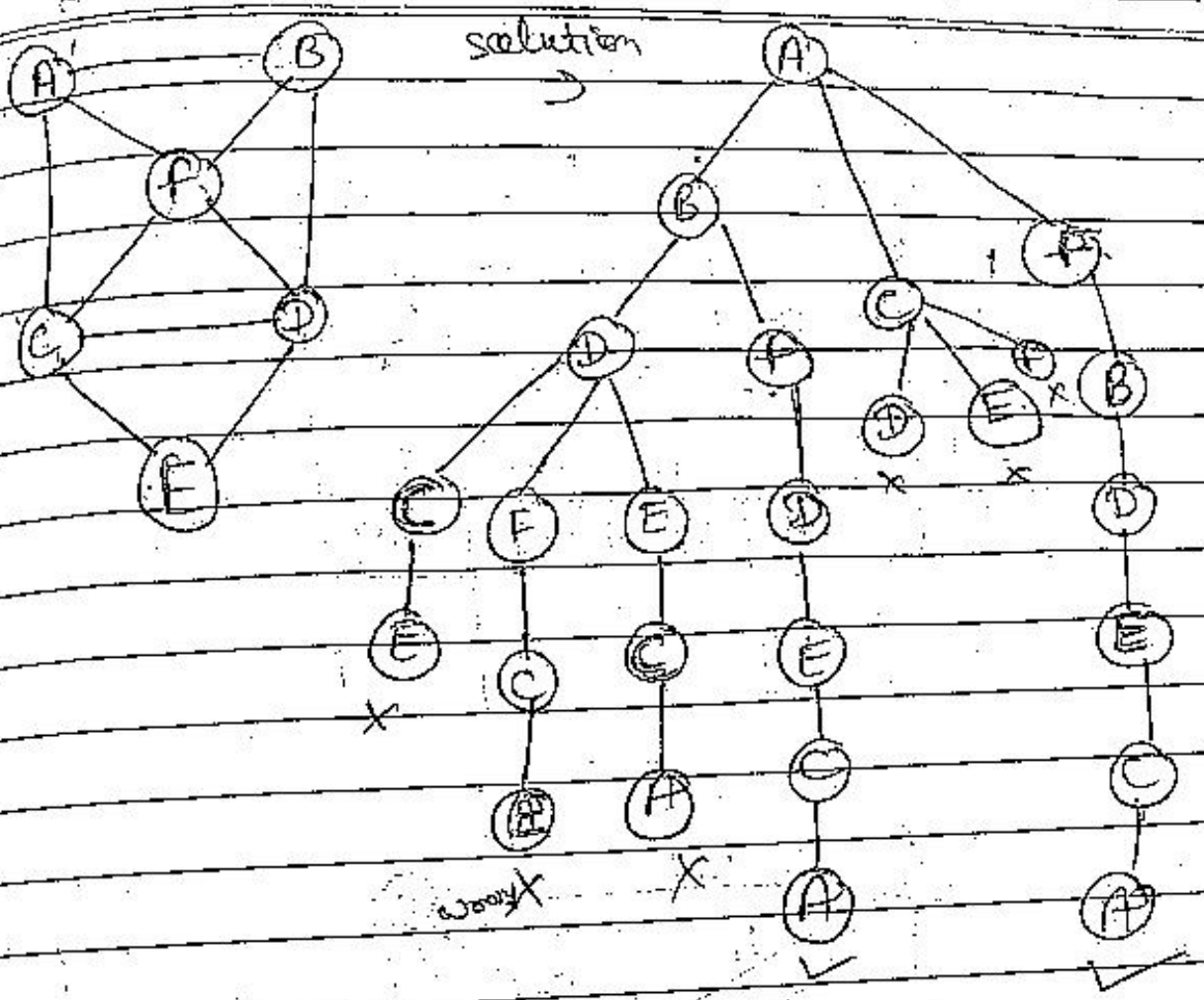
Solve to R G B

multiple soln in this given ques.

we have fill this graph from these color
· too connected vertex can't be same
color.

★ Hamiltonian Cycle :- Are given A undirected graph and two nodes x & y and find a path from x to y visiting each node in the graph exactly once.

Date : __/__/____ Node repeat nahi hone chahiye.



* Branch & Bound TSP (Travelling salesman Problem)

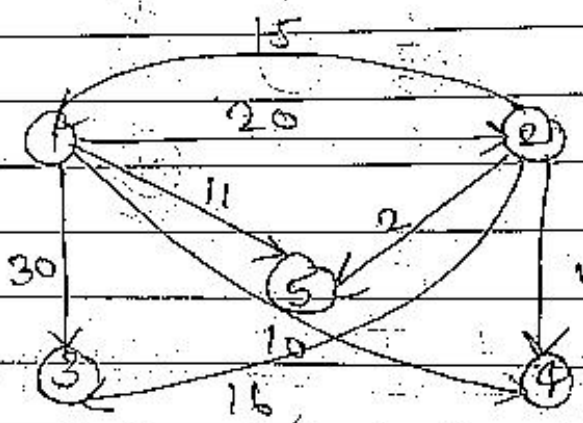
There are n cities and cost of travelling from one city to another city is given. we have to obtain the cheapest cost of a round. In such that each city visited ones and returning to that city and complete the tour. Generally TSP is represented by weighted graph.

Date : / /

Suppose we have 5 cities -

Given

	1	2	3	4	5
1	∞	20	30	10	11
2	15	∞	16	4	2
3	3	5	∞	2	4
4	19	6	18	∞	3
5	16	4	7	16	∞



There are n nodes $n = 5$ and we need to draw a state space tree down (x) in the path that begins at the root and reaching node (x) in state space tree & return to root node in branch & bound strategy.

Cost of each node is computed in TSP by selecting the node with optimum cost i.e $C(x)$ is the approximation cost along the path from node 2 (x).

for TSP there are two method finding the cost
(i) row minimization
(ii) column "

using row minimization,

	1	2	3	4	5	
1	∞	20	30	10	11	→ 10
2	15	∞	16	4	2	→ 2
3	3	5	∞	2	4	→ 2
4	19	6	18	∞	3	→ 3
5	16	4	7	16	∞	→ 4

minimize value of row

21 Total cost = 21

using row minimization, 3 + 4 + 10 + 2 + 2
= 21

update matrix using row minimization.

	1	2	3	4	5
1	∞	10	20	0	1
2	13	∞	14	2	0
3	1	3	∞	0	2
4	16	3	15	∞	0
5	12	0	3	12	∞

minimum value we subtracted from this

This is also known as Red-Row

Each value of row is substracted the minimum value of the row

Using Update row matrix;

Solve column minimization → 1 + 0 + 2 + 0 + 0

Total cost = 4

Red - column matrix:-

	1	2	3	4	5
1	∞	16	17	0	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞

each value of column is subtracted of column minimization.

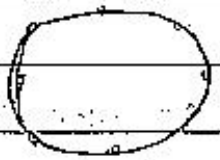
Total reduce cost → 21 + 4 = 25

put the Red column matrix:-

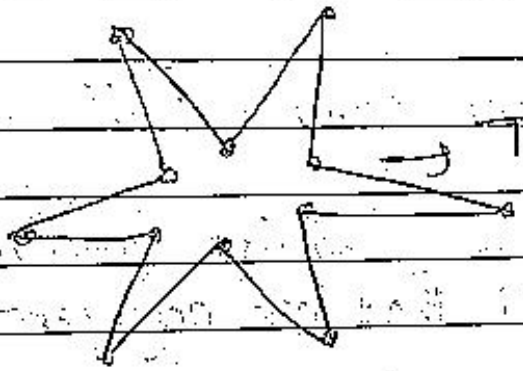
take min cost 1 - 4 - 5 - 2 - 5 - 2 - 3 → wrong, bcz repeat

1 - 4 - 2 - 5 - 3 - 1 Any

Convex Hull :- There exist a set of points on a plane which is said to be convex. If for any two point A and B in the set, the entire line segment with the end points at a and B belongs to be set



→ This is a convex hull.



→ This is not a convex hull

Convex hull of any set S of $n \geq 2$ points which are not on the same line is a convex polygon with the vertices at some point of S. Only the one way to visualize the convex hull is to put a rubberband around all points and let it wrape as tight it can.
The resultant polygon is called convex hull. Convex hull problem of finding the smallest convex polygon that contains given points in a plaine can be solved using devide & conquer policy or method. The version of solving convex hull problem is called quick hull. that is based on quick short technique.

☆ Algorithm :-

Step 1: Sort points ($P_1, P_2, P_3 \dots P_n$) by their x coordinates.

Step-2: repeatedly find the convex hull through points P_1 to $P_{n/2}$.

Step 3: Repeatdly find convex hull through point $P_{n/2+1}$ to P_n.

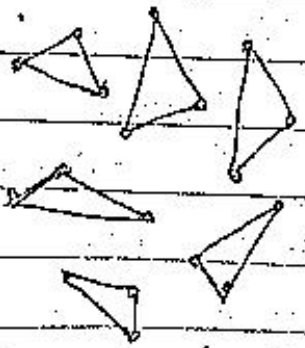
Step-4: Merge the two convex hull

Step 5: Merge procedure require finding a bridge blw two hull that are adjacent to each other.

Step 6: Concatinate left part of left hull and right part of right hull.

Step 7: They algorithm has same efficiency as quick Sort as order of $O(n \log n)$ in average case & ~~worst~~ $O(n^2)$ in worst case.

Example:

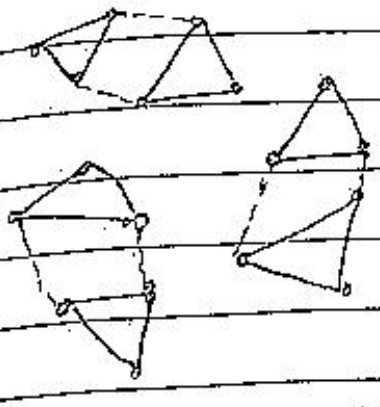


add to near by points

Convex hull of adjacent point

Date : __/__/____ Page No.: ____

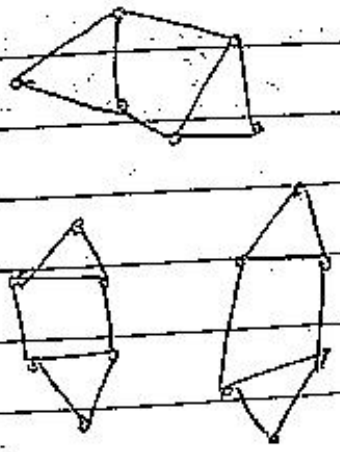
6. Concatenate near by hull to dotted



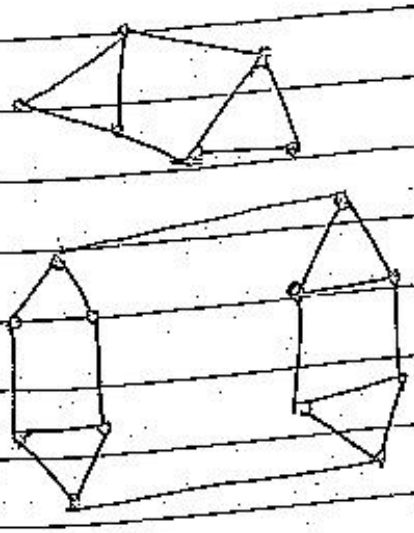
Merge the lower & upper convex hull by dotted line.

upper & lower

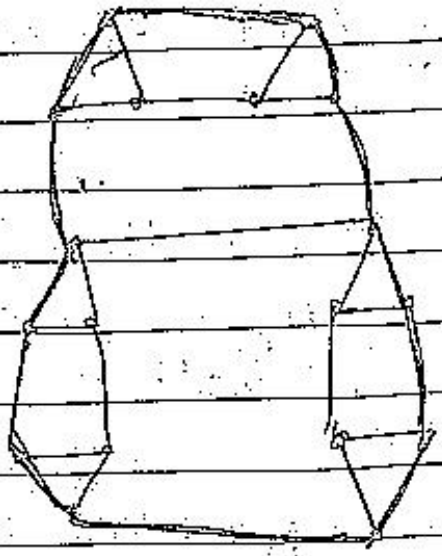
→ ~~cont~~ concatinate these ^ hull by permanent line segments



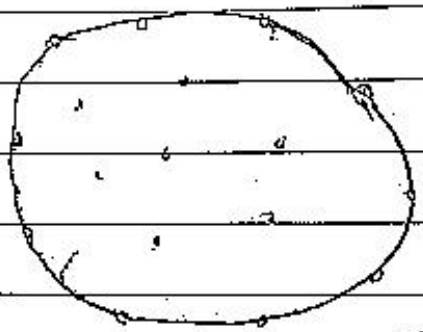
→ merge left & right convex hull.



→



→

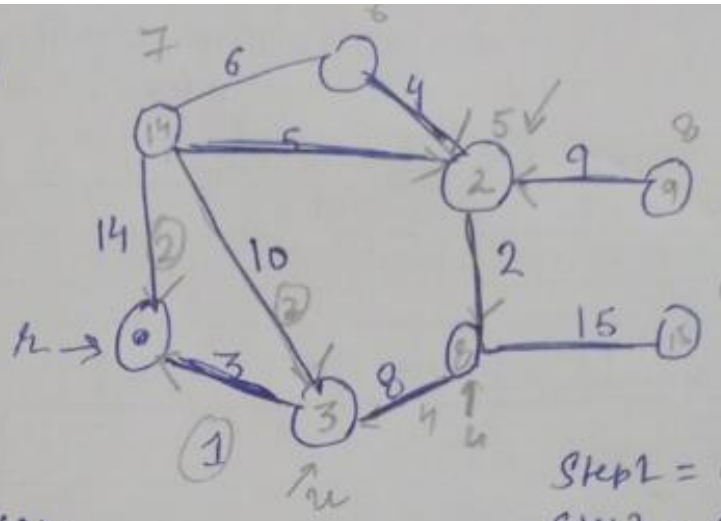


now we find the convex hull.

Ans

8/1

```
MST-Prim (G, w, r)
  Q = V[G];
  for each u ∈ Q
      Key[u] = ∞;
  Key[r] = 0;
  P[r] = null;
  While (Q not empty)
      u = ExtractMin(Q);
      for each v ∈ adj[u]
          If (v ∈ Q and w(u,v) < Key[v])
              P[v] = u;
              Key[v] = w(u,v);
```

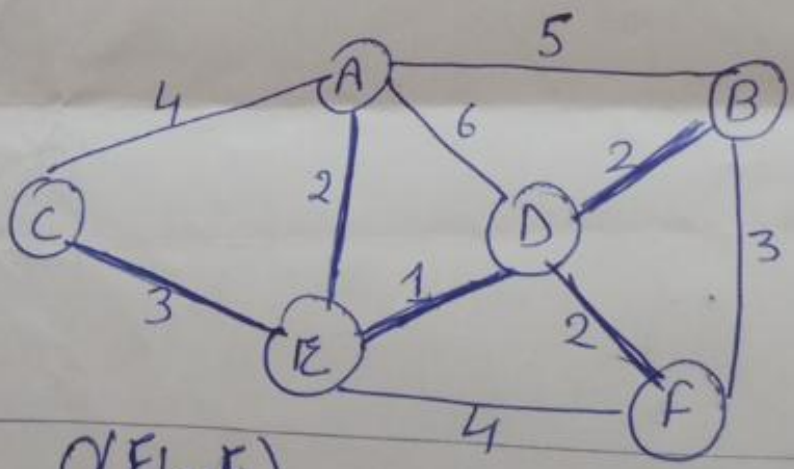


Step 2 = ∞

Step 2 = ⊙ r

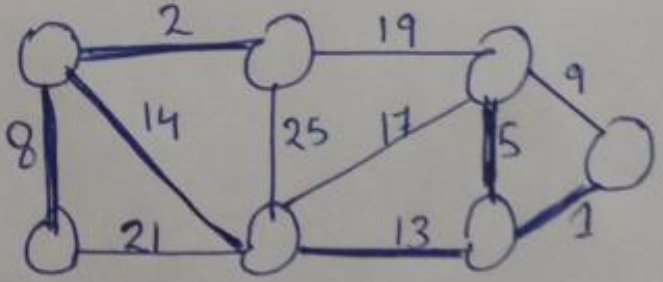
8] 11 11
3 18
5] 9 15
4 4
9) 9
15 15

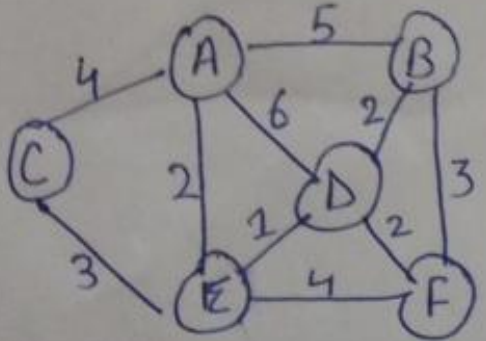
O(ElogV)



O(ElogE)

```
Kruskal ()
{
    T = Q
    for each v ∈ V
        Makeset (v);
    Sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        If FindSet (u) ≠ FindSet (v)
            T = T ∪ {{u,v}};
            Union (FindSet (u), FindSet (v));
}
```





$D \rightarrow \Sigma \mid aSc$

```
*  Fractional Knapsack (Array V, Array, w, int W)
1)  for i = 1 to size(V)
2)      do P[i] = V[i]/w[i]
3)  Sort- decending (P)
4)  i = 1
5)   while (W>0)
6)      do amount = min(W, w[i])
7)          Solution[i] = amount
8)          W = W - amount
9)          i = i + 1
10)  return Solution
```

ml	
4	12
8	32
2	40
6	30
1	50

10ml

1 ml of 5
2 ml of 3
6 ml of 4
1 ml of 2

Date : ___/___/___ Page No.: _

0, 1 Knapsack (v, w, n, W)

for w = 0 to W do
    c[0, w] = 0
for i = 1 to n do
    c[i, 0] = 0
    for w 1 to W do
        If wi < w then
            If [vi + c[i-1, w-wi]. then
                c[i, w] = vi + c[i-1, w-wi]
        else c[i, w] = c[i-1, w].
    else
        c[i, w] = c[i-1, w]

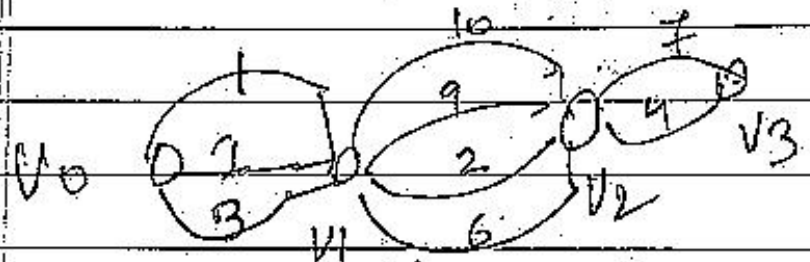
Date : [Optimal Reliability] Page No.:

Suppose that a problem can be solved by a sequence of decisions. The greedy method has that each decision is locally optimal

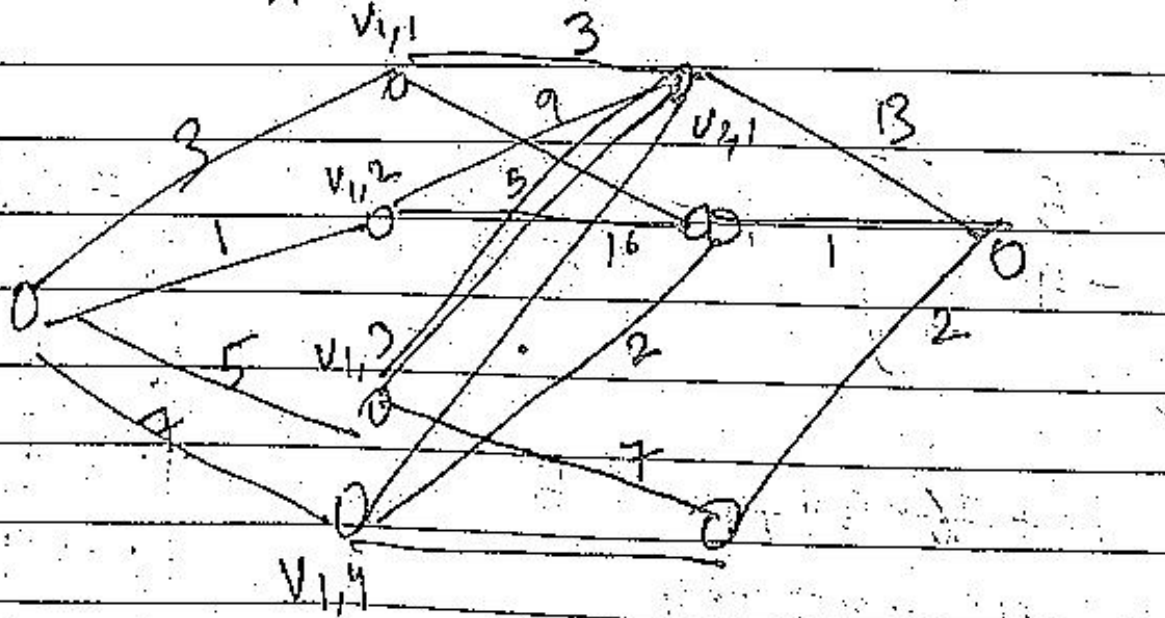
these locally optimal solutions will finally add up to a global optimal solution.

only a few optimization problem can be solved using greedy approach

Shortest path



1+2+4+7



V0 V12 V21 V3 = 23