Date :

1.a)

Greedy Approach	Dynamic Programming
(i) Generates a single decision sequence	(i) Many decision sequence may be generated.
(ii) It is less reliable.	(ii) It is highly reliable.
(iii) It follows Top-down approach	(iii) It follows Bottom-up Approach.
(iv) Efficiency is more.	(iv) Efficiency is less.
(v) Overlapping subproblems cannot be handled.	(v) Chooses the optimal solution to the subproblem
(vi) Contain a particular set of feasible set of solutions.	(vi) There is no special set of feasible set of solution.
(vii) It is easier to engage in greedy procedures than to use Dijkstras shortest path algorithm which takes O(ElogV + VLog V) time.	(vii) Bellman Ford algorithm, which is based on dynamic programming takes O(V) time.
(viii) Fractional knapsack is an example of greedy algorithms.	(viii) 0/1 knapsack problem is an example of greedy algorithms.

optimal

Page No.

Date :

(ix) No memorization is required	(ix) memorization is required
(x) A greedy strategy is faster than dynamic one.	(x) Compared to greedy programming it is slower.
(xi) Fast results	(xi) Slow results comparatively
(xii) Each step is locally optimal	(xii) Past solutions are used to create new ones.
The process of looking back and revisiting previous choices is more costly in terms of memory.	The memorization solution requires a DP table which increases memory complexity.

Page No.
Date :

1.b.

		0	1	2	3	4	5	6	7
		x_i	a	b	a	b	a	a	b
0	y_i	0	0	0	0	0	0	0	0
1	a	0	↖ 1	← 1	↖ 1	← 1	↖ 1	↑ 1	← 1
2	b	0	↑ 1	↖ 2	← 2	↖ 2	← 2	← 2	↖ 2
3	a	0	↖ 1	↑ 2	↖ 3	← 3	↖ 3	↖ 3	← 3
4	b	0	↑ 1	↖ 2	↑ 3	↖ 4	← 4	← 4	↖ 4
5	b	0	↑ 1	↑ 2	↑ 3	↖ 4	↑ 4	↑ 4	↖ 5
6	a	0	↖ 1	↑ 2	↖ 3	↑ 4	↖ 5	↖ 5	↑ 5
7	a	0	↖ 1	↑ 2	↖ 3	↑ 4	↖ 5	↖ 6	← 6

LCS(x,y) = (a,b,a,b,a,a)

~~a b a b a a~~

2.a) Find the order of parenthesization for the optimal chain multiplication, where sequence of dimension d = (15, 5, 10, 20, 25)

→ Step 1 :- Compute First Diagonal of Matrix m Given that product vector P is

P =	15	5	10	20	25
	P_0	P_1	P_2	P_3	P_4

Therefore matrices and their orders are as follows:

$A_1 = 15 \times 5$

$A_2 = 5 \times 10$

$A_3 = 10 \times 20$

$A_4 = 20 \times 25$

Date :

Now for first diagonal :
$m[i,j] = 0$ if $i=j$

$m[1,1] = 0$
$m[2,2] = 0$
$m[3,3] = 0$
$m[4,4] = 0$

Step 2 : Compute Second Diagonal of matrix m and s.
Now for second diagonal :
$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1} p_k p_j$ for $i \le k < j$

For $m[1,2]$ only one value of k is possible : $1 \le 1 < 2$
Therefore we have :
$m[1,2] = m[1,1] + m[2,2] + p_0 p_1 p_2$
$= 0 + 0 + 15 \times 5 \times 10$
$m[1,2] = 750$ for $k=1$
$s[1,2] = 1$

For $m[2,3]$ only one value of k is possible:
$2 \le 2 < 3$
Therefore we have :
$m[2,3] = m[2,2] + m[3,3] + p_1 p_2 p_3$
$= 0 + 0 + 5 \times 10 \times 20$
$m[2,3] = 1000$ for $k=2$
$s[2,3] = 2$

For $m[3,4]$ only one value of k is possible:
$3 \le 3 < 4$
Therefore we have :
~~$m[3,4]$~~

Date :

$m[3,4] = m[3,3] + m[4,4] + P_2 \times P_3 \times P_4$
$= 0 + 0 + 10 \times 20 \times 25$
$m[3,4] = 5000$ for $k=3$
$s[3,4] = 3$

Step3 : Compute Third Diagonal of Matrix m ands
Now for Third diagonal :

$m[i,j] = \min(m(i,k) + m[k+1,j) + P_{i-1} P_k P_j$ for $i \leq k < j$

For $m[1,3]$ two values of k are possible as $k=1,2$ due to $1 \leq 1,2 < 3$
Therefore we have :
When $k=1$
$m[1,3] = m[1,1] + m[2,3] + P_0 P_1 P_3$
$= 0 + 1000 + 15 \times 5 \times 20$
$m[1,3] = 2500$ for $k=1$

When $k=2$
$m[1,3] = m[1,2] + m[3,3] + P_0 P_2 P_3$
$= 750 + 0 + 15 \times 10 \times 20$
$m[1,3] = 3750$ for $k=2$

Now we have :
$m[1,3] = 2500$ for $k=1$
$m[1,3] = 3750$ for $k=2$

Therefore
$m[1,3] = \min(2500, 3750)$
$m[1,3] = 2500$ for $k=1$
$s[1,3] = 1$

Page No.

Date :

For m[2,4] two values of k are possible as k = 2,3 due to $2 \le 2,3 < 4$
Therefore we have
When k=2
$m[2,4] = m[2,2] + m[3,4] + P_1 P_2 P_4$
$= 0 + 5000 + 5 \times 10 \times 25$
$m[2,4] = 6250$ for k=2

When k=3
$m[2,4] = m[2,3] + m[4,4] + P_1 P_3 P_4$
$= 1000 + 0 + 5 \times 20 \times 25$
$m[2,4] = 3500$ for k=3

Now we have,
$m[2,4] = 6250$ for k=2
$m[2,4] = 3500$ for k=3

Therefore
$m[2,4] = \min(6250, 3500)$
$m[2,4] = 3500$ for k=3
$s[2,4] = 3$

Step 4 : Compute 4th diagonal of matrix m and s
Now for 4th diagonal

$m[i,j] = m[i,k] + m[k+1,j] + P_{i-1} P_k P_j$ for $i \le k < j$
For [1,4] three values of k are possible.
as k = 1,2,3.
Therefore we have
when k = 1

Date :

$m[1,4] = m[1,1] + m[2,4] + P_0P_1P_4$
$= 0 + 3500 + 15 \times 5 \times 25$
$m[1,4] = 5375$ for $k = 1$

when $k=2$, $m[1,4] = m[1,2] + m[3,4] + P_0P_2P_4$
$= 750 + 5000 + 15 \times 10 \times 25$
$m[1,4] = 9500$ for $k=2$

when $k=3$, $m[1,4] = m[1,3] + [4,4] + P_0P_3P_4$
$= 2500 + 0 + 15 \times 20 \times 25$
$m[1,4] = 10,000$ for $k=3$

$m[1,4] = \min(5375, 9500, 10000)$
$m[1,4] = 5375$ for $k=1$
$s[1,4] = 1$

Step 5: compute optimal Number of Multiplication.
The optimal number of multiplication $= m[1,n]$

Now $n \leftarrow$ length$(P) - 1$
$n \leftarrow 5 - 1$
$n = 4$

The optimal number of multiplication $= m[1,4]$
$= 5375$.

As we have only 4 matrices A_1, A_2, A_3, A_4 therefore the structure of matrix m and matrix s will be as follows :-

1	2	3	4	
0	750	2500	5375	1
	0	1000	3500	2
		0	5000	3
			0	4

m(i,j)

2	3	4	
1	1	1	1
	2	3	2
		3	3

s(i,j)

Page No. _____

Date :

Step 6 : Compute optimal Parenthization

We have total four matrices

Therefore first parentization will be as follows:

$(A_1 | A_2\ A_3\ A_4)$

Now, check for the breakage from lower limit to upper limit that is from 1 to 4.

Check for $s[1,4]$

$s[1,4] = 1$

Therefore we have

$(A_1 (A_2\ A_3\ A_4))$ -- optimal Parenthization.

check for S[1,4] = 1

Therefore we have
$(A_1(A_2A_3A_4))$ -- optimal Parenthization.

2b)
→ Purpose of Matrix chain Multiplication algorithm.

(i) Matrix chain multiplication is an optimization problem concerning the most efficient way to multiply a given sequence of matrices.

(ii) The matrix chain multiplication problem generalizes to solving a more abstract problem: given a linear sequence of objects, an associative binary operation on those objects, and a way to compute the cost of performing that operation on any two given objects, compute the minimum cost way to group the objects to apply the operation over the sequence.

3.a) $m = 7$, $n = 3$, $(w_1, w_2, w_3) = (2, 3, 3)$

$(P_1, P_2, P_3) = (1, 3, 4)$

P_i	w_i	$w \downarrow$ / $i \rightarrow$	0	1	2	3	4	5	6	7
		0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1
3	3	2	0	0	1	3	3	4	4	4
4	3	3	0	0	1	4	4	5	5	(7)

$V(i, w) = \max \{ V[i-1, w] \; ; V[i-1, w - w[i]] + P[i] \}$

$V(3, 1) = \max \{ V(2, 1) \; ; V[2, 1-3] + 4 \}$

$= \max \{ V(2, 1) \; ; V[2, -2] + 4 \}$

$= 0$

($\because$ 2, -2 doesn't exist so V(2,1) will only be considered)

$V(3,2) = \max\{V(2,2) ; V[2, 2-3] + 4\}$

$= \max\{V(2,2) , V[2,-1] + 4\}$

$= 1$ ($\because$ - - - -)

$V(3,3) = \max\{V(2,3) ; V[2, 3-3] + 4\}$

$= \max\{V(2,3) ; V[2,0] + 4\}$

$= \max\{V(2,3) , 0 + 4\}$

$= \max\{3, 4\}$

$= 4$

$V(3,4) = \max\{V(2,4) ; V[2, 4-3] + 4\}$

$= \max\{V(2,4) , V[2,1] + 4\}$

$= \max\{3 , 0 + 4\}$

$= \max\{3, 4\}$

$= 4$

$V(3,5) = \max\{V(2,5) ; V[2, 5-3] + 4\}$

$= \max\{4 ; V[2,2] + 4\}$

$= \max\{4 , 1 + 4\}$

$= \max\{4, 5\}$

$= 5$

$V(3,6) = \max\{V(2,6) ; V[2, 6-3] + 4\}$

$= \max\{4 ; V[2,3] + 4\}$

$= \max\{4 , 3 + 4\}$

$= \max\{4, 7\}$

$= 7$

$V(3,7) = \max\{V(2,7) ; V[2, 7-3] + 4\}$

$= \max\{4 ; V[2,4] + 4\}$

$= \max\{4, 7\}$ $\Rightarrow$

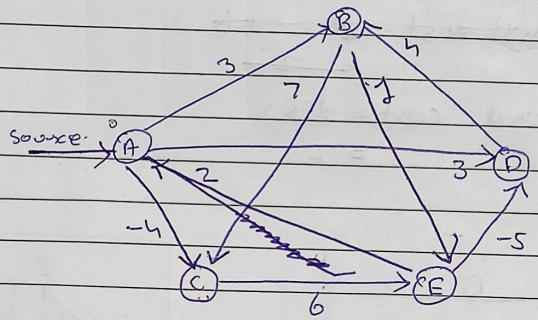
x_1	x_2	x_3
0	1	1

$7 - 4 = 3$

$3 - 3 = 0$

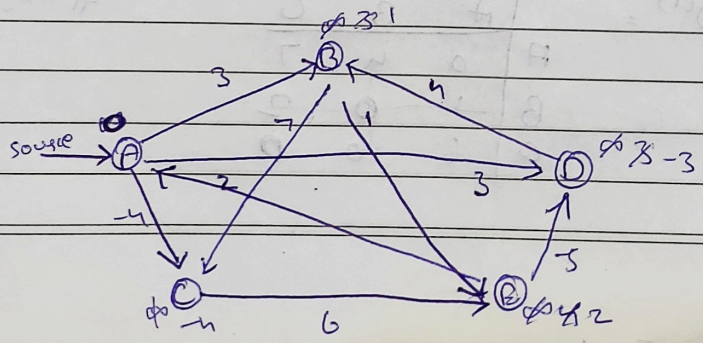
$\therefore$ soln $= \{0, 1, 1\}$

3b



Edges: (A,B) (A,D) (B,C) (B,E) (C,E) (D,B) (E,A) (E,D)

	A	B	C	D	E
0	∞	∞	∞	∞	∞
1	0	3	-4	3	∞
2	0	3	-4	-3	2
3	0	1	-4	-3	2
4	0	1	-4	-3	2



∴ we have distance vectors as.

A	B	C	D	E
0	-1	-4	-3	2

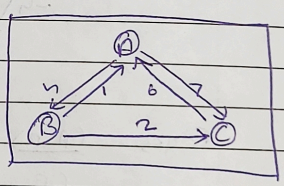
∴ Shortest path from vertex A to E
∴ A → C → E

~~and cost = 0 -4+6~~
~~= 2~~

and cost = d(E)
= 2

4a.

	A	B	C
A	0	4	7
B	1	0	2
C	6	∞	0



$D^{(0)}$ =

	A	B	C
A	0	4	7
B	1	0	2
C	6	∞	0

$\Pi^{(0)}$

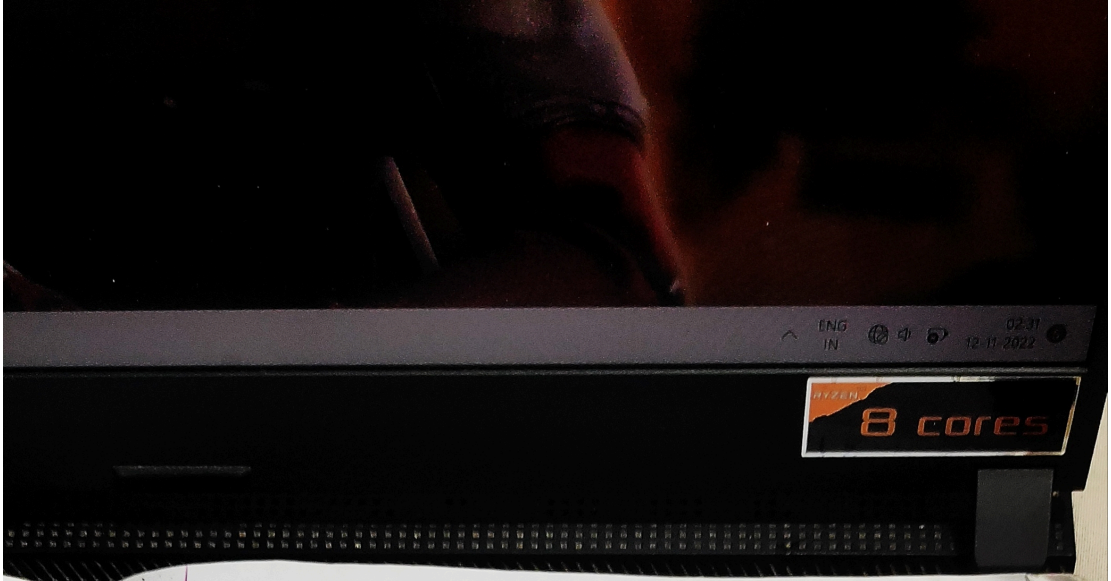
	A	B	C
A	N	A	A
B	B	N	B
C	B	N	N

$D^{(1)}$ =

	A	B	C
A	0	4	7
B	1	0	2
C	6	10	0

$\Pi^{(0)}$ =

	A	B	C
A	N	A	A
B	B	N	B
C	C	A	N



Page No.
Date :

$D^{(2)}$ =

	A	B	C
A	0	4	6
B	1	0	2
C	6	10	0

$\Pi^{(2)}$ =

	A	B	C
A	N	A	B
B	B	N	B
C	C	A	N

$D^{(3)}$ =

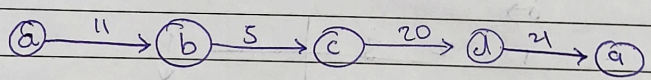
	A	B	C
A	0	4	6
B	1	0	2
C	6	10	0

$\Pi^{(3)}$ =

	A	B	C
A	N	A	B
B	B	N	B
C	C	A	N

4b

	a	b	c	d
a	0	(11)	12	14
b	18	0	(5)	16
c	19	19	0	(20)
d	21	16	12	0



∴ Shortest path = 11 + 5 + 20 + 21

= 57

Page No.
Date :

5a. a=

i	0	1	2	3
p_i	0.3	0.2	0.1	0.1
q_i	0.1	0.1	0.05	0.05

→ j →

0	1	2	3		i ↓
0.1	0.4	0.55	0.7	1	
	0.1	0.25	0.4	2	
		0.05	0.20	3	
w(i,j) ⇒			0.05	4	

j →

0	1	2	3		i ↓
0	0.6	1.05	1.6	1	
	0.1	0.40	0.8	2	
		0.05	0.30	3	
e(i,j)			0.05	4	

j →

1	2	3		i ↓
1	1	1,2	1	
	2	2	2	
		3	3	

root[i,j]

Page No.
Date:

Step 1: filling of $w[i,j]$ matrix

Diagonal 1 =>

$$w(i,j) = \begin{cases} q_{i-1}, & \text{if } j = i-1 \\ w[i,j-1] + p_j + q_j & \text{if } i < j \end{cases}$$

$w(1,0) = j = 1-1 = 0$ i.e. $q_{i-1} = q_{1-1} = q_0 = 0.1$

$w(2,1) = j = 2-1 = 1$ i.e. $q_{2-1} = q_1 = 0.1$

$w(3,2) = j = 3-1 = 2$ i.e. $q_{3-1} = q_2 = 0.05$

$w(4,3) = j = 4-1 = 3$ i.e. $q_{4-1} = q_3 = 0.05$

diagonal 2 =>

$w[1,1] = w[1,0] + p_1 + q_1$

$= 0.1 + 0.2 + 0.1 = 0.4$

$w[2,2] = w[2,1] + p_2 + q_2$

$= 0.1 + 0.1 + 0.05$

$= 0.25$

$w[3,3] = w[3,2] + p_3 + q_3$

$= 0.05 + 0.1 + 0.05$

$= 0.20$

diagonal 3 =>

$w[1,2] = w[1,1] + 0.1 + 0.05$

$= 0.4 + 0.1 + 0.05$

$= 0.55$

$w[2,3] = w[2,2] + 0.1 + 0.05$

$= 0.25 + 0.1 + 0.05$

$= 0.4$

diagonal 4 =>

$w[1,3] = w[1,2] + 0.1 + 0.05$

$= 0.55 + 0.1 + 0.05$

$= 0.7$

Page No.
Date :

step 2 complete e and root matrix

$e[i,j] = q_{i-1}$ if $j = i-1$

$= \min \{ e[i, r-1] + e[r+1, j] + w(i,j) \}$ if $i < j$

diagonal 1:
$e[1,0] = q_{1-1} = q_0 = 0.1$
$e[2,1] = q_{2-1} = q_1 = 0.1$
$e[3,2] = q_{3-1} = q_2 = 0.05$
$e[4,3] = q_{4-1} = q_3 = 0.05$

diagonal 2:
$e[1,1] = \min \{ e(1,1-1) + e(1+1, 1) + w(1,1) \}$ for $r = 1$
$= \min \{ e(1,0) + e(2,1) + w(1,1) \}$
$= 0.1 + 0.1 + 0.4$
$= 0.6$

$e[2,2] = \min \{ e(2,1) + e(3,2) + w(2,2) \}$ for $r = 2$
$= 0.1 + 0.05 + 0.25$
$= 0.40$

0.1
0.25
0.40

$e[3,3] = 0.05 + 0.05 + 0.20$ for $r = 3$
$= 0.30$

diagonal 3: $e[1,2]$:

for $r = 1$: $e[1,0] + e[2,2] + w[1,2]$
$= 0.1 + 0.40 + 0.55$
$= 1.05$

for $r = 2$: $e[1,1] + e[3,2] + w[1,2]$
$= 0.6 + 0.05 + 0.55$
$= 1.2$

min = (1.05) for $r = 1$

e[2,3] :

~~for r=1 : e[2,2] + e[2,3] + w[1,3]~~
~~= 1.05 +~~
~~=~~

for r=2 : e[1,1] +

~~for r=1 : e[2,2] + e[2,3] + w[2,3]~~
~~= 0.40 +~~

for r=2 : e[2,1] + e[3,3] + w[2,3]
= 0.1 + 0.30 + 0.4
= 0.8

for r=3 : e[2,2] + e[4,3] + w[2,3]
= 0.40 + 0.05 + 0.4
= 0.85

min (0.8) for r=2

Dragonal

⇒ e[1,3] :

for r=1 : e[1,0] + e[2,3] + w[1,3]
= 0.1 + 0.8 + 0.7
= 1.6

for r=2 : e[1,1] + e[3,3] + w[1,3]
= 0.6 + 0.30 + 0.7
= 1.6

for r=3 : e[1,2] + e[4,3] + w[1,3]
= 1.05 + 0.05 + 0.7
= 1.8

Page No.
Date :

Step 3:

Construct OBST from root matrix.

1	2	3	
1	1	1,2	1
	2	2	2
		3	3

We have ~~Key~~ 1 key (a_1, a_2, a_3, a_4)

$\therefore$ root (a_1, a_2, a_3, a_4) = root $(1,4)$

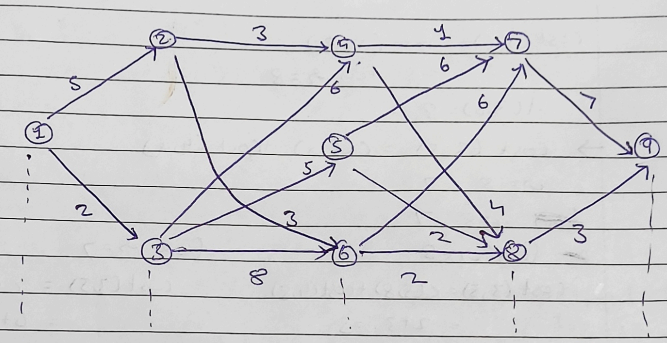
We have 4 key (a_0, a_1, a_2, a_3)

$\therefore$ root (a_0, a_1, a_2, a_3) = root $(0,3)$

= - - -

aage dekh lena thoda sa
abhi step 3 last step hai

6.a.



$cost[i,j] = \min \{ c(i,s) + cost(i+1,s) \mid s \in V_{i+1} \text{ and } (i,s) \in E \}$

→ Computation at level 4 :

$cost(4,8) = c(8,s) + cost(5,s)$

$s = 9$

for $s=9$, $cost(4,8) = c(8,9) + cost(5,9)$

$= 3 + 0$

$= 3$

$d(4,8) = 9$

$cost(4,7) = c(7,s) + cost(5,s)$

$s = 9$

for $s=9$: $cost(4,7) = c(7,9) + cost(5,9)$

$= 7 + 0 = 7$

$d(4,7) = 9$

computation at level 5 :

$cost(5,9) = 0$

computation at level 3 :

→ $cost(3,6) = c(6,s) + cost(4,s)$

$s = 8, 7$

for $s = 8$

$cost(3,6) = c(6,8) + cost(4,8)$

$= 2 + 3 = 5$

for $s = 7$

$cost(3,6) = c(6,7) + cost(4,7)$

$= 6 + 7 = 13$

cost(3,6) = min(5,13)
= 5 for r = 8

d(3,6) = 8

→ cost(3,5) = c(5,r) + cost(4,r)
r = 8, 7

~~for~~ r = 8, 7

~~for~~ for r = 8

cost(3,5) = c(5,8) + cost(4,8)
= 2+3 = 5

cost(3,5) = min(5,13)
= 5 for r = 8

for r = 7
cost(3,5) = c(5,7) + cost(4,7)
= 6+7 = 13

d(3,5) = 8

→ cost(3,4) = c(4,r) + cost(4,r)
r = 8, 7

for r = 8
cost(3,4) = c(4,8) + cost(4,8)
= 4+3 =>

cost(3,4) = min(7,8)
=> for r = 8

d(3,4) = 8

for r = 7
cost(3,4) = c(4,7) + cost(4,7)
= 1+7 = 8

Computation of level 2.

~~for~~

→ cost(2,3) = c(3,r) + cost(3,r)
r = 6, 5, 4

for r = 6 :
cost(2,3) = c(3,6) + cost(3,6)
= 8+5 = 13

~~for~~ r = 5 :
cost(2,3) = c(3,5) + cost(3,5)
= 5+5 = 10

for r = 4 :
cost(2,3) = c(3,4) + cost(3,4)
= 6+7 = 13

Page No.
Date :

cost(2,3) = min (13, 10, 13)
= 10 for s = 5
d(2,3) = 5

→ cost (2,2) = c(2,s) + cost(3,s)
s = 6 and 4
for s = 6 :
cost(2,2) = c(2,6) + cost(3,6)
= 3+5 = 8
for s = 4 :
cost (2,2) = c(2,4) + cost(3,4)
= 3+7 = 10
cost(2,2) = min(8,10)
= 8 for s = 6
d(2,2) = 6

Computation at level 1 :
→ cost(1,1) = c(1,s) + cost(2,s)
s = 3 and 2
for s = 3
cost(1,1) = c(1,3) + cost(2,3)
= 2+10 = 12
for s = 2
cost(1,1) = c(1,2) + cost(2,2)
= 5+8 = 13
cost(1,1) = min(12,13)
= 12 for s = 3
d(1,1) = 3

Cost(i,j)	d(i,j)
cost(5,9) = 0	
cost(4,8) = 3	d(4,8) = 9
cost(4,7) = 7	d(4,7) = 9
cost(3,6) = 5	d(3,6) = 8
cost(3,5) = 5	d(3,5) = 8
cost(3,4) = 7	d(3,4) = 8
cost(2,3) = 10	d(2,3) = 5
cost(2,2) = 8	d(2,2) = 6
cost(1,1) = 12	d(1,1) = 3

algo followed below: p(1) ← 1
p(k) ← n
for j=2 to k-1
do p(j) ← d(p(j-1))

j here is kernel

for level 1:

①

for level 2:

p(2) ← d(p(2-1))
p(2) ← d(p(1))
p(2) ← d(1)
p(2) ← d(1,1)
d(2) = 3 ③

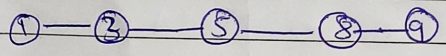
for level 3

p(3) ← d(p(3-1))
p(3) ← d(p(2))
p(3) ← d(3)
p(3) ← d(2,3)
p(3) = 5 ⑤

for level 4

p(4) ← d(p(4-1))
p(4) ← d(p(3))
p(4) ← d(5)
p(4) ← d(3,5)
p(4) = 8 ⑧

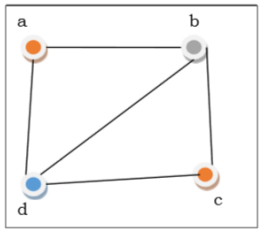
∴ Shortest path:



path length = 12

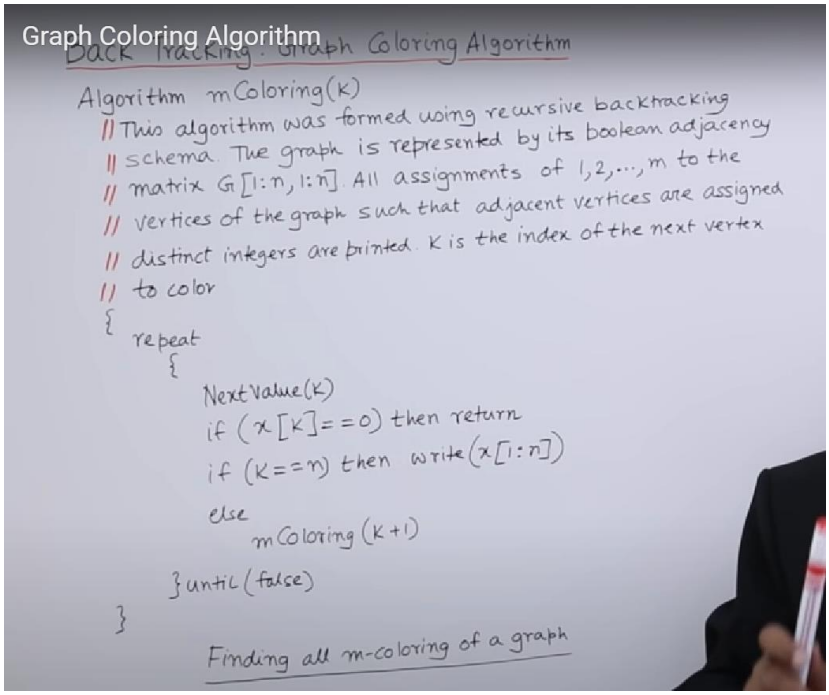
7. a) Explain Graph coloring method with example. Give algorithm for it.

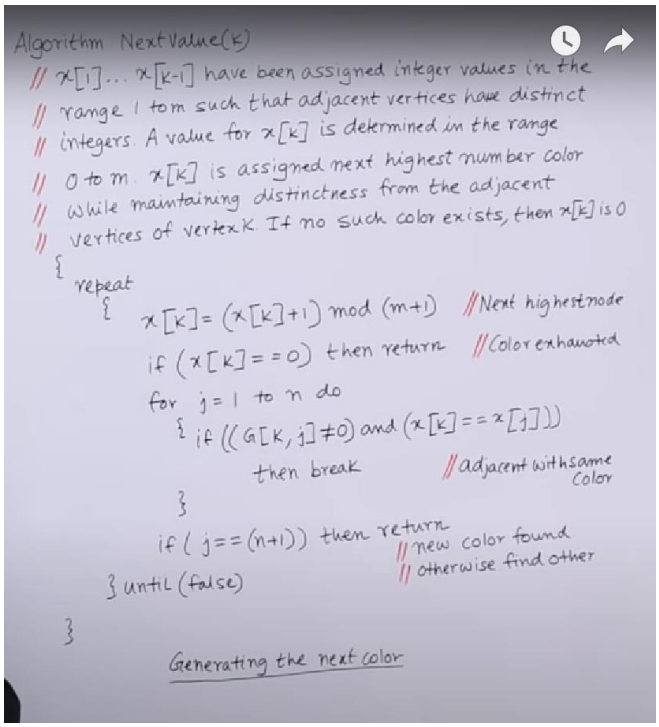
ANS: Graph coloring is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. The objective is to minimize the number of colors while coloring a graph. The smallest number of colors required to color a graph G is called its chromatic number of that graph. Graph coloring problem is a NP Complete problem. Example:



In the above figure, at first vertex a is colored red. As the adjacent vertices of vertex a are again adjacent, vertex b and vertex d are colored with different color, green and blue respectively. Then vertex c is colored as red as no adjacent vertex of c is colored red. Hence, we could color the graph by 3 colors. Hence, the chromatic number of the graph is 3.

Algorrithm:

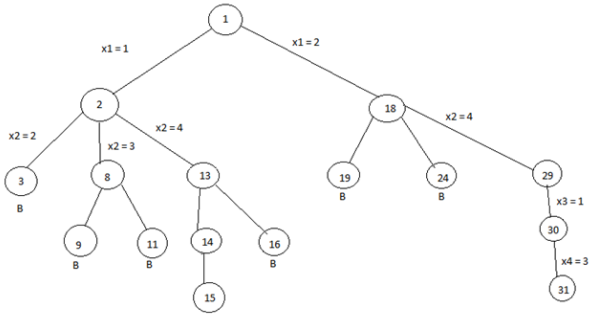




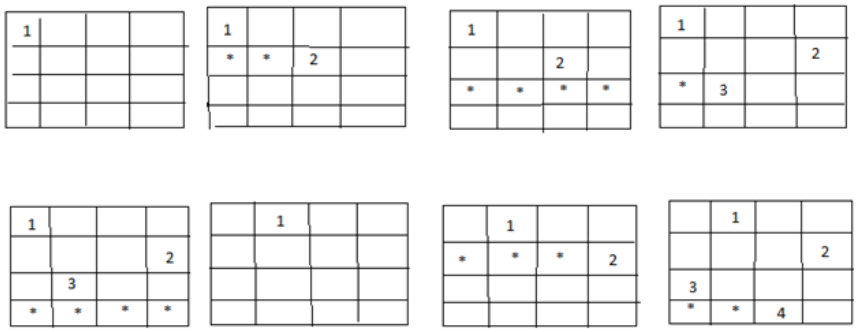
7. b) Discuss 4-Queen's problem and give its algorithm using Backtracking approach

ANS: OVERVIEW: In 4- queens problem, we have 4 queens to be placed on a 4*4 chessboard, satisfying the constraint that no two queens should be in the same row, same column, or in same diagonal.
The solution space according to the external constraints consists of 4 to the power 4, 4-tuples i.e., Si = {1, 2, 3, 4} and 1<= I <=4, whereas according to the internal constraints they consist of 4! solutions i.e., permutation of 4.

Extra: We can solve 4-queens problem through backtracking by taking it as a bounding function .in use the criterion that if (x1, x2, ……., xi) is a path to a current E-node, then all the children nodes with parent-child labelings x (i+1) are such that (x1, x2, x3, ….., x(i+1)) represents a chessboard configuration in which no queens are attacking.
So we start with the root node as the only live node. This time this node becomes the E-node and the path is (). We generate the next child. Suppose we are generating the child in ascending order. Thus the node number 2 is generated and path is now 1 i.e., the queen 1 is placed in the first row and in the first column.
Now, node 2 becomes the next E-node or line node. Further, try the next node in the ascending nodes i.e., the node 3 which is having x2 = 2 means queen 2 is placed in the second column but by this the queen 1 and 2 are on the same diagonal, so node 3 becomes dead here so we backtrack it and try the next node which is possible.
Here, the x2 = 3 means the queen 2 is placed in the 3rd column. As it satisfies all the constraints so it becomes the next live node.
After this try for next node 9 having x3 = 2 which means the queen 3 placed in the 2nd column, but by this the 2 and 3 queen are on the same diagonal so it becomes dead. Now we try for next node 11 with x3 = 4, but again the queens 2 and 3 are on the same diagonal so it is also a dead node.



***** The B denotes the dead node.
We try for all the possible positions for the queen 3 and if not any position satisfy all the constraints then backtrack to the previous live node.



Now, the node13 become the new live node with x2 = 4, means queen 2 is placed in the 4th column. Move to the next node 14. It becomes the next live node with x3 = 2 means the queen 3 is placed in the 2nd column. Further, we move to the next node 15 with x4 = 3 as the live node. But this makes the queen 3 and 4 on the same diagonal resulting this node 15 is the dead node so we have to backtrack to the node 14 and then backtrack to the node 13 and try the other possible node 16 with x3 = 3 by this also we get the queens 2 and 3 on the same diagonal so the node is the dead node.
So we further backtrack to the node 2 but no other node is left to try so the node 2 is killed so we backtrack to the node 1 and try another sub-tree having x1 = 2 which means queen 1 is placed in the 2nd column.
Now again with the similar reason, nodes 19 and 24 are killed and so we try for the node 29 with x2 = 4 means the queen 2 is placed in the 4th column then we try for the node 30 with x3 = 1 as a live node and finally we proceed to next node 31 with x4 = 3 means the queen 4 is placed in 3rd column.
Here, all the constraints are satisfied, so the desired result for 4 queens is {2, 4, 1, 3}.

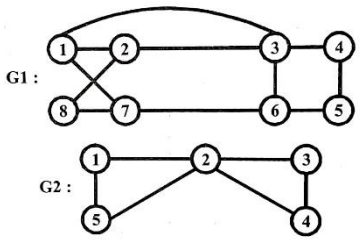
ALGORITHM:

```
private static boolean isSafePlace(int column, int Qi, int[] board) {
//check for all previously placed queens
for (int i = 0; i < Qi; i++) {
if (board[i] == column) { // the ith Queen(previous) is in same column
return false;}
//the ith Queen is in diagonal
//(r1, c1) - (r2, c1). if |r1-r2| == |c1-c2| then they are in diagonal
if (Math.abs(board[i] - column) == Math.abs(i - Qi)) {
return false;}}
return true;}
```

8. a) Explain backtracking algorithm for sum of subsets problem. State its implicit and explicit constraints.

8. b) Discuss Hamiltonian cycle. Also write an algorithm for finding Hamiltonian cycle of a graph.

ANS: Let G=(V, E) be a connected graph with n vertices. A Hamiltonian cycle is a round-trip path along n edges of G that visits every vertex once and returns to its initial or starting position. In other words if a Hamiltonian cycle begins at some vertex



Vi Î G and the vertices of G are visited in the order V1, V2, ..., Vn+1, then the edges (Vi, Vi+1) are in E, 1<=i<n, and the Vi are different except for Vi and Vn+1, which are equal. Hamiltonian cycle was suggested by Sir William Hamilton.

Fig. shows a graph G1 which contains the Hamiltonian cycle 1, 2, 8, 7, 6, 5, 4, 3, 1. The graph G2 does not contain any Hamiltonian cycle.

There is no easy way to find whether a given graph contains a Hamiltonian cycle. We have backtracking algorithm that finds all the Hamiltonian cycles in a graph. The graph may be directed or undirected. Only distinct cycles are output.

Algorithm – The backtracking solution vector (X1, ...,Xn) is defined so that xi represents the ith visited vertex of the proposed cycle. Now we have to find how to compute the set of possible vertices for Xk if X1, ..., Xk-1, have already been chosen. If k = 1 the x, can be any of the n vertices. To avoid printing the same cycle n times, we require that xy = 1. If 1 <k<n, then xk can be any vertex v that is distinct from X1, X2, ...,Xk-1 and v is connected by an edge to Xk-1. Only the vertex Xn be the one remaining vertex and it must be connected to both Xn-1 and X1. Algorithm 1 NextValue (k) determines a possible next vertex of the proposed cycle.

START HERE:

```
Algorithm Hamiltonian (k)
/* This algorithm uses the recursive formulation of backtracking to find all the Hamiltonian cycles of a graph. The graph is stored as an adjacency matrix G[1 : n, 1:n). All cycles begin at node 1.*/
{
repeat
{ // Generate values for x [k].
NextValue (k); // Assign a legal next value to x[k].
if (x [k] = 0) then return;
if (k = n) then write (x [1 : n]);
else Hamiltonian (k+ 1);
} until (False);
.}
```

9. a) Comment on P = NP

ANS: Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.All problems in P can be solved with polynomial time algorithms, whereas all problems in NP - P are intractable.It is not known whether P = NP. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.

If P ≠ NP, there are problems in NP that are neither in P nor in NP-Complete.

The problem belongs to class P if it's easy to find a solution for the problem. The problem belongs to NP, if it's easy to check a solution that may have been very tedious to find.

b) Explain Polynomial Reduction

ANS:

c) Give the definitions of NP hard and NP-complete class of problems

ANS: ***NP-Hard Program***

Any given problem X acts as NP-Hard only if there exists a problem Y that is NP-Complete. Here, problem Y becomes reducible to problem X in a polynomial time. The hardness of an NP-Hard problem is equivalent to that of the NP-Complete Problem. But here, the NP-Hard Problems don't need to be in the NP Class.

NP-Complete Problem=

Any given problem X acts as NP-Complete when there exists an NP problem Y- so that the problem Y gets reducible to the problem X in a polynomial line. This means that a given problem can only become NP-Complete if it is a part of NP-Hard as well as NP Problems. A Turing machine of non-deterministic nature can easily solve this type of problem in a given polynomial time.

10. a) Explain Non deterministic algorithm. Give non deterministic algorithm for searching and sorting problem

ANS: A non-deterministic algorithm can provide different outputs for the same input on different executions. Unlike a deterministic algorithm which produces only a single output for the same input even on different runs, a non-deterministic algorithm travels in various routes to arrive at the different outcomes.

Non-deterministic algorithms are useful for finding approximate solutions, when an exact solution is difficult or expensive to derive using a deterministic algorithm.

ALGORITM:

```
Algorithm nd_search(a,n,x)
{// “a” = array of size “n” and “x” is element to be searched
for i = 1 to n do
{ j = select(a,n) //select a location “j” from given array
if (a[j] = x) □ Verification process
success(); }
failure(); }
//Try above algorithm using repeat until and comment upon the time complexity//
Algorithm nd_sort(a,b,n)
{ //”a” is array to be sorted and “b” is array for auxiliary storage
for i = 1 to n do
{ j = select(a,n)
b[i] = a[j] //create the array “b” by selecting “n” elements }
for j = 1 to n do
{ if(b[i] > b[i+1])
failure(); }
success(); }
```

10. b) Explain the concept of Polynomial Reduction and how it can be used for showing NP completeness of problem

ANS: