GROUP-A

1(a): Greedy strategy does not give the optimal solution for 0/1 knapsack. do you agree with this statement? justify your answer with example.

Answer:

Greedy strategy does not give the optimal solution for 0/1 knapsack . The given statement is correct. The greedy strategy for 0/1 knapsack problem involves selecting items based on their individual values-to weight ratios and adding them to the knapsack until it is full or no more items can be added.

Here is an example to justify the given statement.

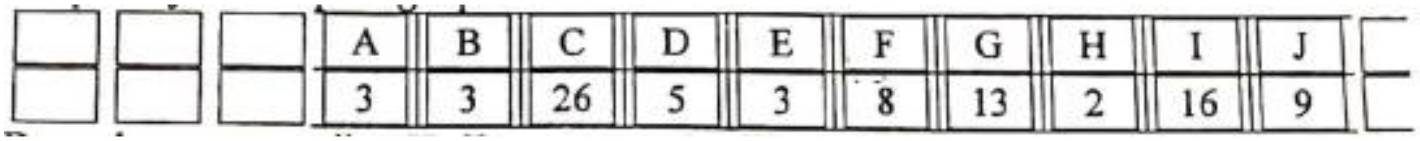
Let us consider that the capacity of the knapsack is W = 8 and the items are as shown in the following table.

Item	**A**	**B**	**C**	**D**
Profit	2	4	7	10
Weight	1	3	5	7

Using the greedy approach of 0/1 knapsack, the weight that's stored in the knapsack would be A+B = 4 with the maximum profit 2 + 4 = 6. But, that solution would not be the optimal solution.

Therefore, dynamic programming must be adopted to solve 0-1 knapsack problems.

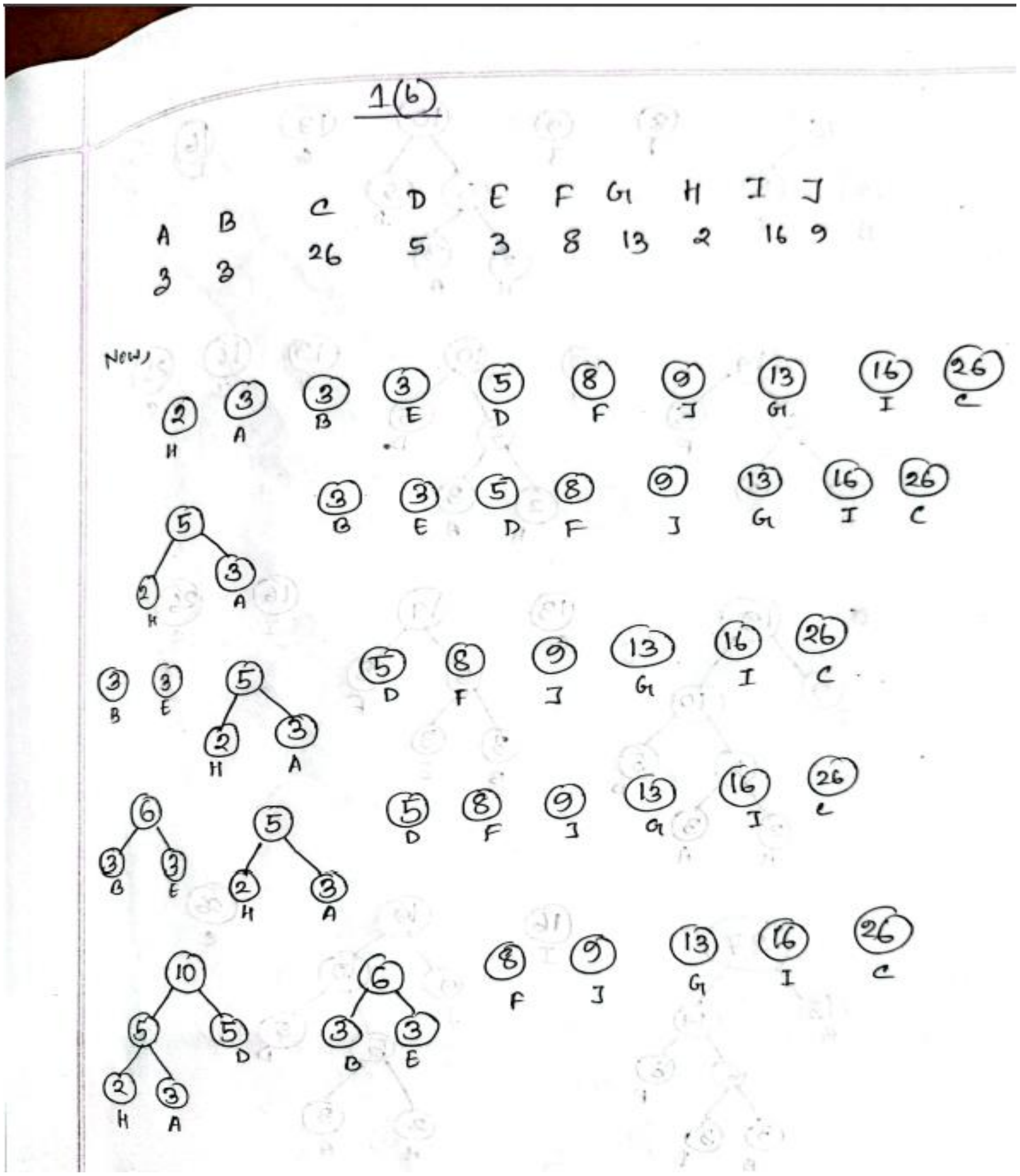
1(b): You have been asked to encode a paragraph with Huffman coding scheme. This paragraph contains the following symbols along with their associate frequency in the paragraph.

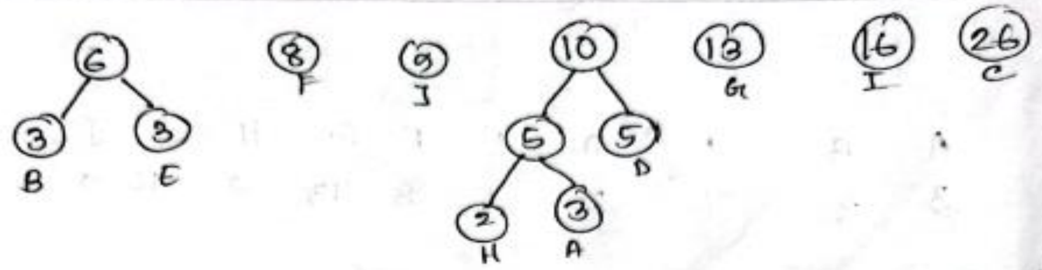


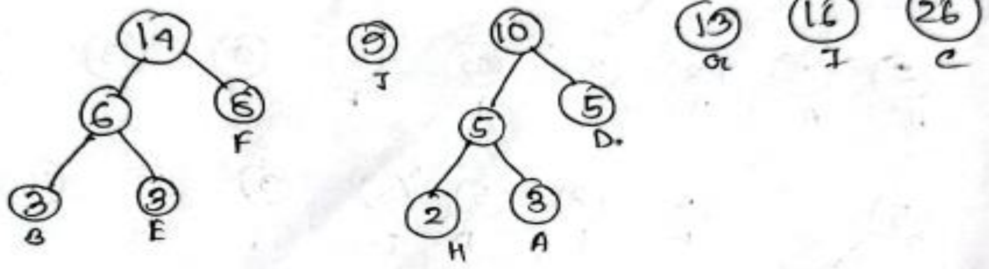
			A	B	C	D	E	F	G	H	I	J	
			3	3	26	5	3	8	13	2	16	9	

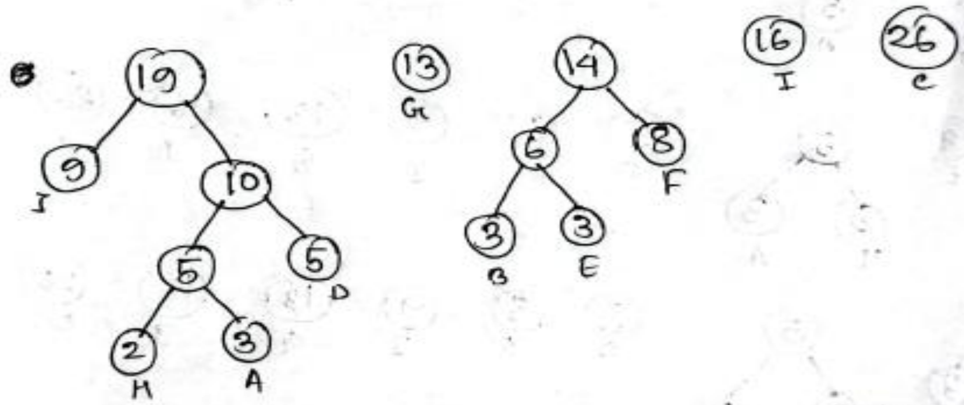
Draw the corresponding Huffman tree and utilizing that tree tell the binary encoding for each letter.

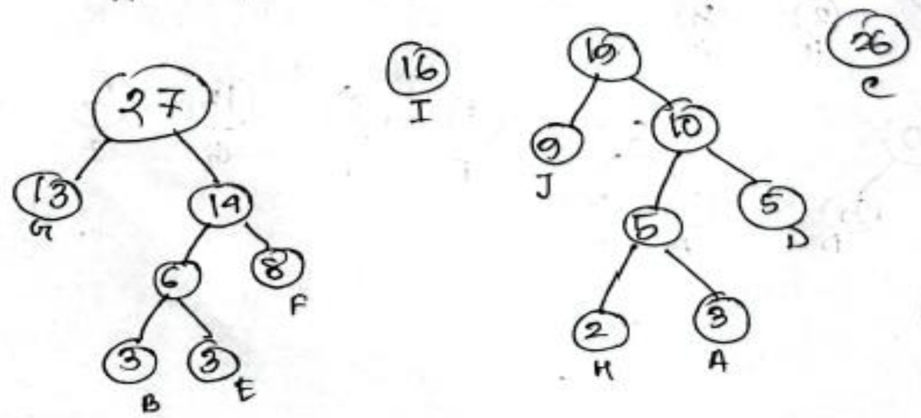
Answer:

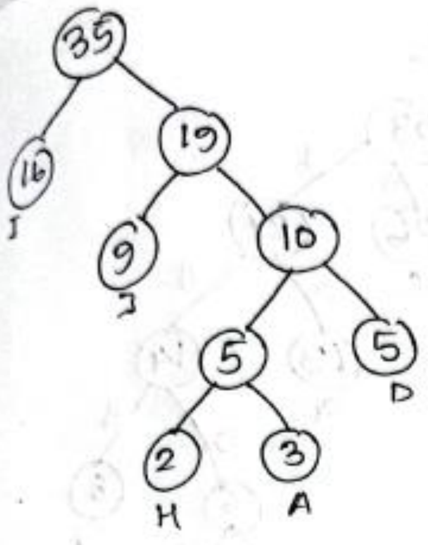


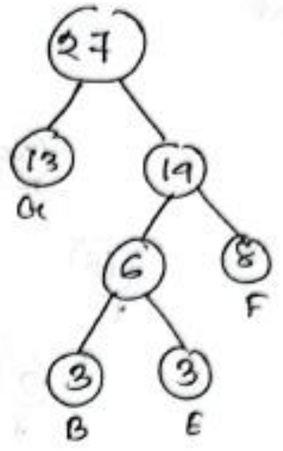

6
3 B
3 E
8 F
9 J
10
5
5 D
2 H
3 A
13 G
16 I
26 C

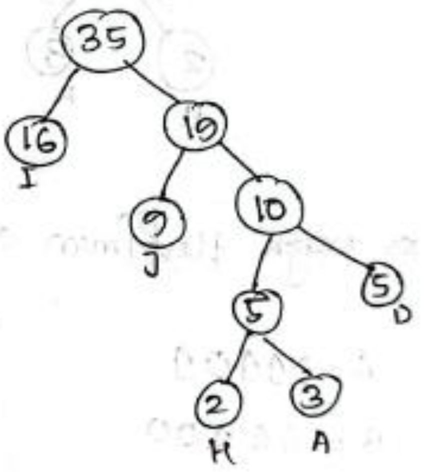

14
6
8 F
3 B
3 E
9 J
10
5
5 D
2 H
3 A
13 G
16 I
26 C

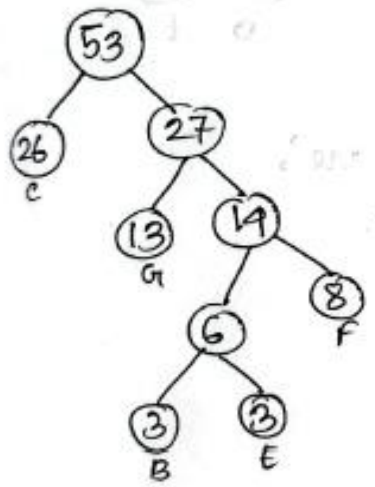

19
9 J
10
5
5 D
2 H
3 A
13 G
14
6
8 F
3 B
3 E
16 I
26 C

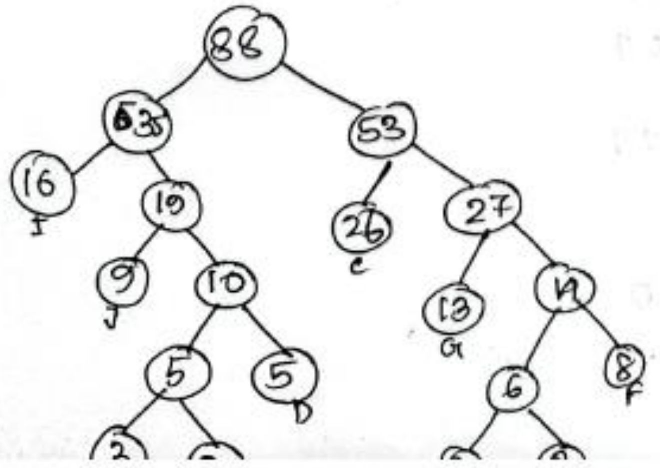

27
13 G
14
6
8 F
3 B
3 E
16 I
19
9 J
10
5
5 D
2 H
3 A
26 C





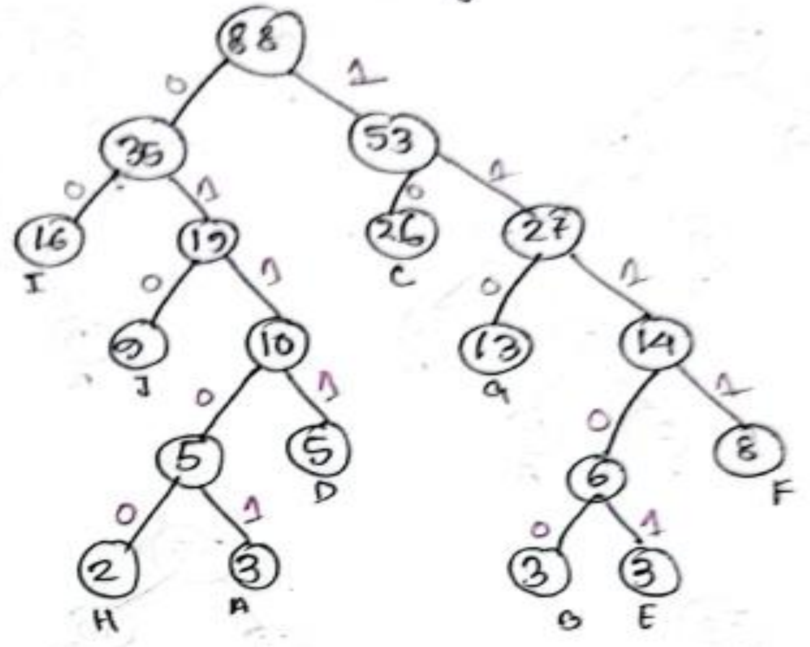






This is our final Huffman Tree.

Let us assign weight "0" to the left edges and "1" to the right.



Let us assign Huffman codes are:

A = 01101

B = 11100

C = 10

D = 0111

E = 11101

F = 1111

G = 110

H = 01100

I = 00

J = 010

OR,

1(b): Suppose we have a knapsack that has a weight limit w. There are items i1,i2, …, in each having w1,w2, …wn and some benefit associated with v1,v2, …., vn. We need to maximize the benefit such that the total weight inside the knapsack is at most w. Demonstrate an algorithm to solve this problem.

<u>Answer:</u>

The problem described is the classic 0/1 Knapsack problem, where we aim to maximize the total benefit while respecting the weight constraint of the knapsack. I'll outline a dynamic programming algorithm to solve this problem:

In this approach we'll define 2 dimensional DP of index for items defined on rows whereas weights from 1 to W on columns and for every weight we can compute the answer for placing items till nth item.

Thus for every, Dp[i][j] we can calculate values for these two cases and store out the maximum of those two ,

Therefore we can get our answer for {i,j} as ,

DP[i][j] = max(v[i] + DP[i-1][j-w[i]],DP[i-1][j])

Pseudocode:

```
 int knapsack01(int W,int N,vector<int> &v,vector<int> &w)
 {
   int DP[N+1][W+1]; // Defining DP /* If there is no space left that is W reaches 0 then DP[i][0]  for every i will be 0.*/
   for(int i=0;i<N+1;i++) DP[i][0] = 0 /* If there are no items left that is N reaches 0 then DP[0][i] for every i will be 0.*/
   for(int i=0;i<W+1;i++) DP[0][i] = 0;
   for(int i=1;i<N+1;i++)
{
      for(int j=1;j<W+1;j++)
     {
         if(w[i-1] <= j) /* Taking max of both the cases i.e to take that item or to ignore it.*/
          {
            DP[i][j] = max(v[i-1]+DP[i-1][j-w[i-1]],DP[i-1][j]);
         }
         else/* Taking max of both the cases i.e to take that item or to ignore it.*/
         {
            DP[i][j] = DP[i-1][j];
```

```
        }
    }
}
   return DP[N][W];  /* returning answer for W space and N items */
}
```

Time complexity of 0 1 Knapsack problem is O(nW) where, n is the number of items and W is the capacity of knapsack.

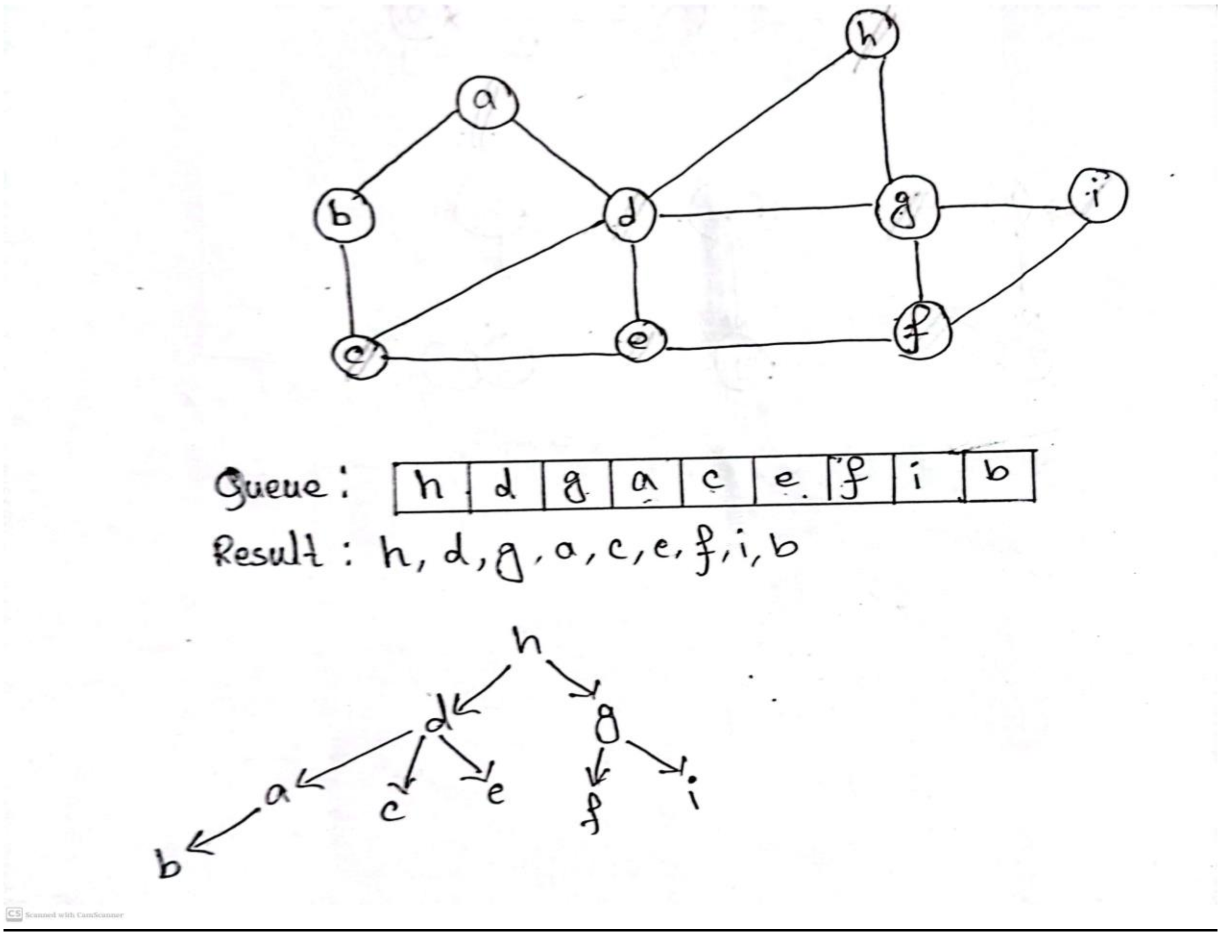
2(a): Draw the following undirected graph

G=(a,b,c,d,e,f,g,h,i}

E={(a,b),(a,d),(b,c),(c,d),(c,e),(d,e),(d,g),(f,e),(g,f),(g,i),(h,d),(h,g),(i,f)}

Traverse the graph using Breadth First Search starting from vertex h. Visit the nodes in lexicographic order (a,b,c ….). Show only the final breadth first tree along with d and π values. You don't need to show the intermediate steps.

Answer:



Queue:

h	d	g	a	c	e	f	i	b

Result : h, d, g, a, c, e, f, i, b

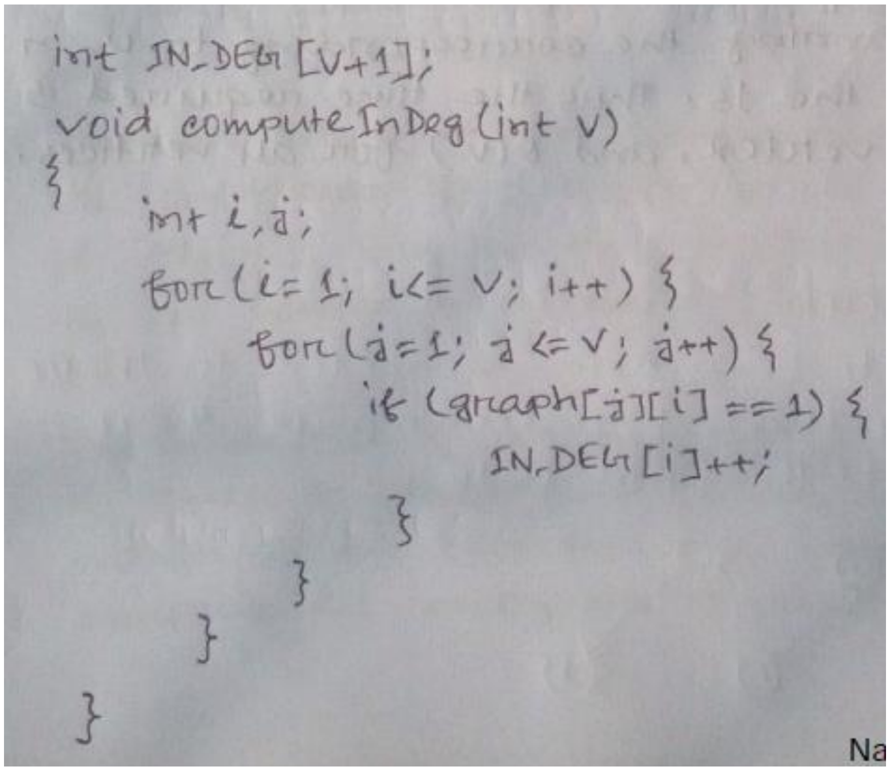
OR

2(a):Traverse the graph using Depth First Search and show the discover time d and finish time f. You don't need to show the intermediate steps.

Answer:

2(b): Write an algorithm for finding the in-degree of each vertex a graph G. G is represented with an adjacency list. What will be the running time of your algorithm?

Answer:

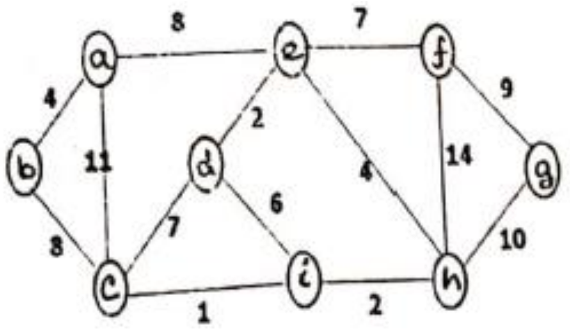


```
int IN_DEG [V+1];
void computeInDeg (int V)
{
    int i,j;
    for (i= 1; i<= V; i++) {
        for (j=1; j <=V; j++) {
            if (graph[j][i] ==1) {
                IN_DEG [i]++;
            }
        }
    }
}
```

Na

The in-degree of a vertex is equal to the number of times it appears in all the lists in Adj. If we search all the lists for each vertex, the time to compute the in-degree of every vertex is Θ(|V||E|).

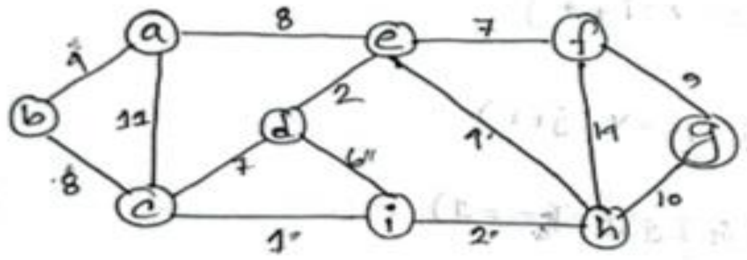
2(c): Write the definition of minimum spanning tree. Consider the following graph. What minimum spanning tree would Prim's algorithm produce? Write the edges in the order that the algorithm would add them to its result.



Answer:

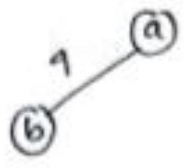
Minimum Spanning Tree :

A minimum Spanning Tree is a subset of edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycle with the minimum possible total edge weight.

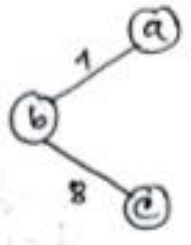


	a	b	c	d	e	f	g	h	i
	0	0	0	0	0	0	0	0	0
T	0	4	8	2	4	7	9	2/	7/6
P	0	a	b	e	h	e	f	i/	c/d

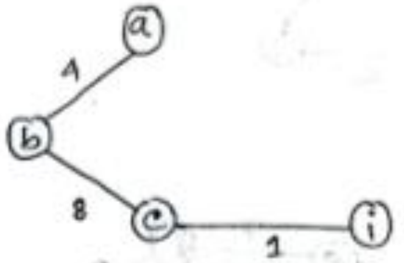
Order of taking edges

1)


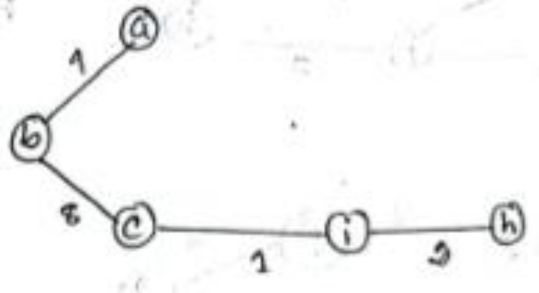
a-b

2)


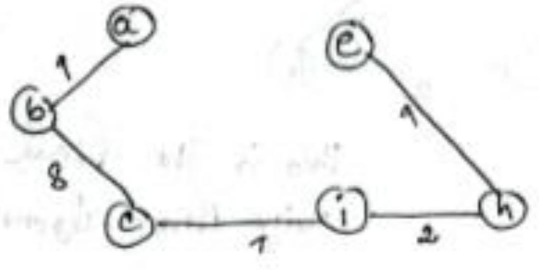
~~ba~~
b—c

3)


c—i

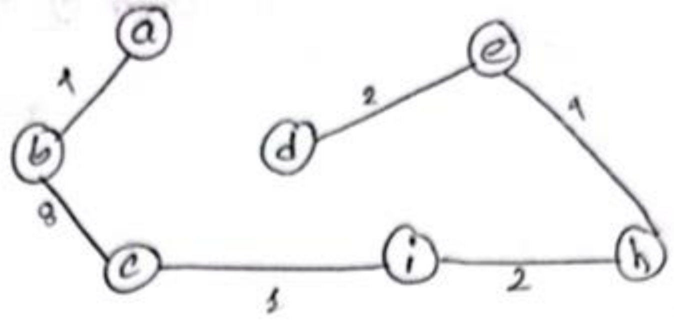
4)


i—h

5)


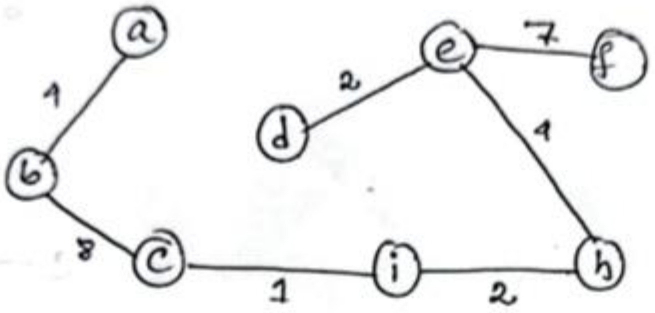
e—h

6)



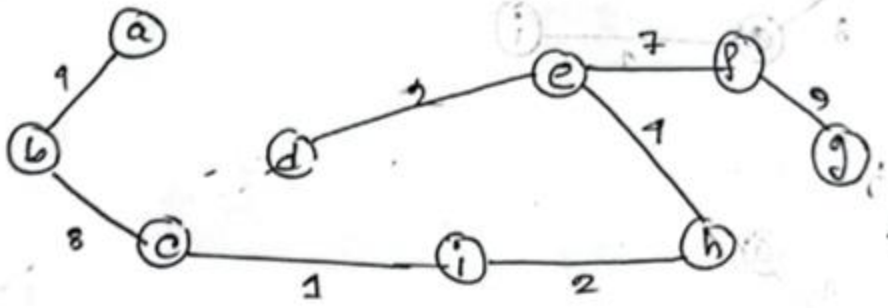
e–d

7)



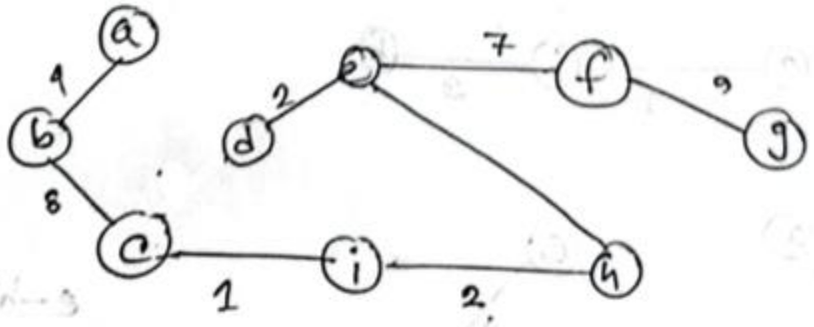
e–f

8)



f–g

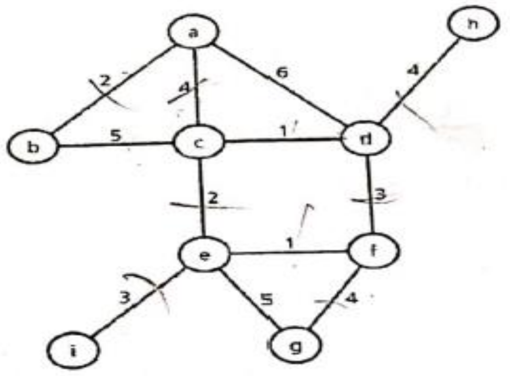
∴ 9)



This is the final MST
using Prims algorithm.

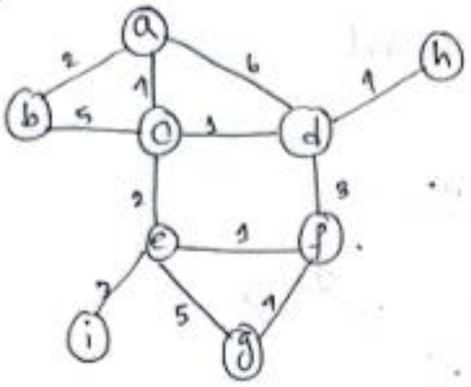
OR

2(c): Write the definition of minimum spanning tree. Consider the following graph. What minimum spanning tree would Krushkal's algorithm produce? Write the edges in the order that the algorithm would add them to its result.



Answer:

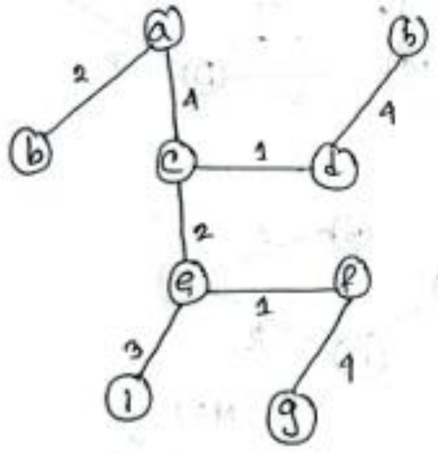
#- Or



Lets sort the edges in non decreasing order;

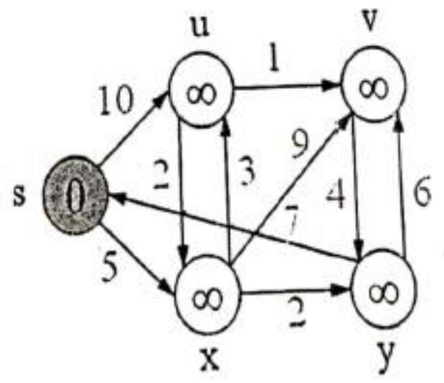
Weight		Terminals
1	—	c-d
1	—	e-f
2	—	a-b
2	—	c-e
3	—	e-i
3	—	d-f×
4	—	a-c
4	—	d-h
4	—	f-g
5	—	b-c×
5	—	e-g.
6	—	a-d.

Now,

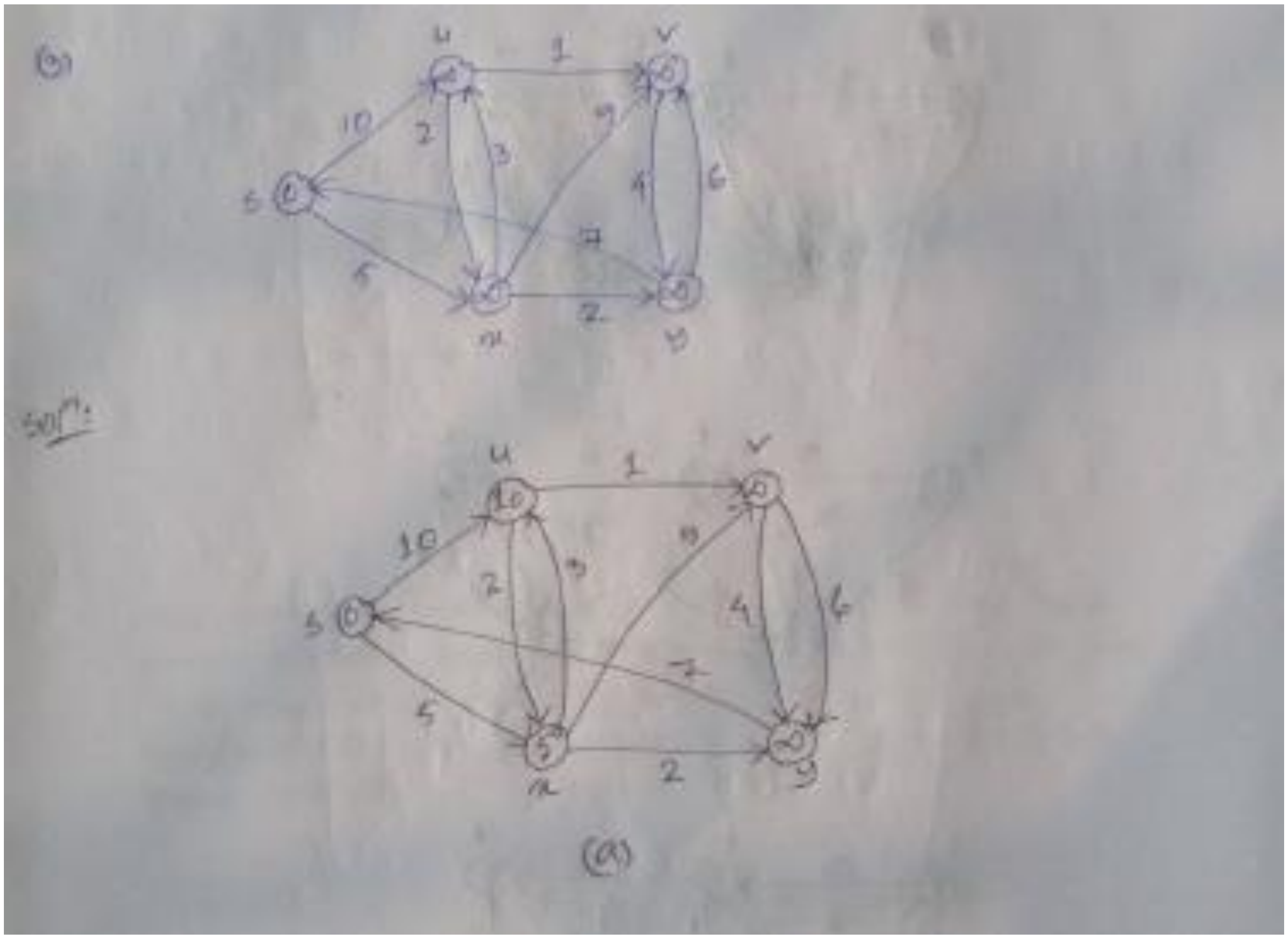


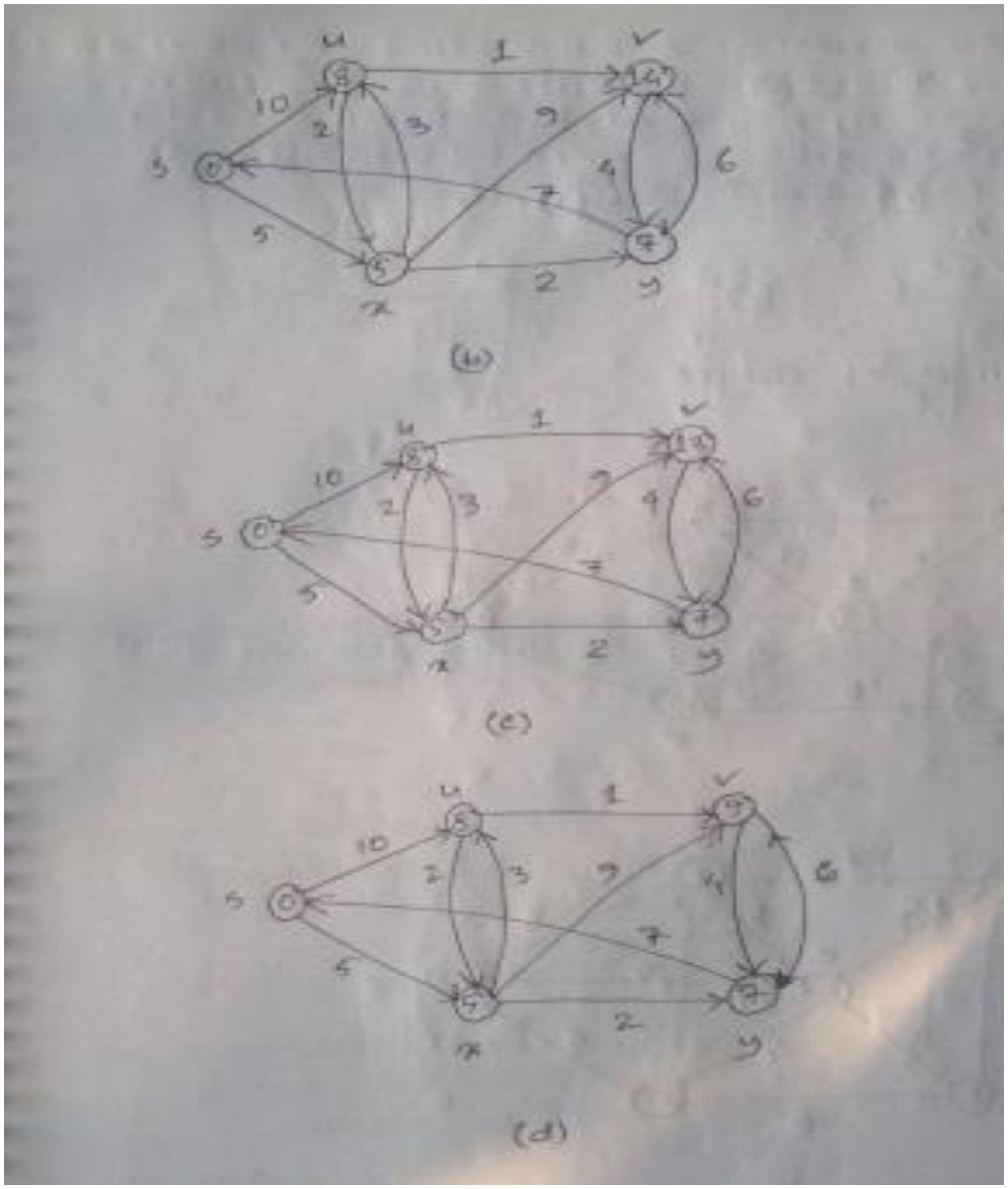
GROUP-B

3(a): Run the Dijkstra's algorithm to find single source shortest path on the weighted directed graph in following figure.



Answer:



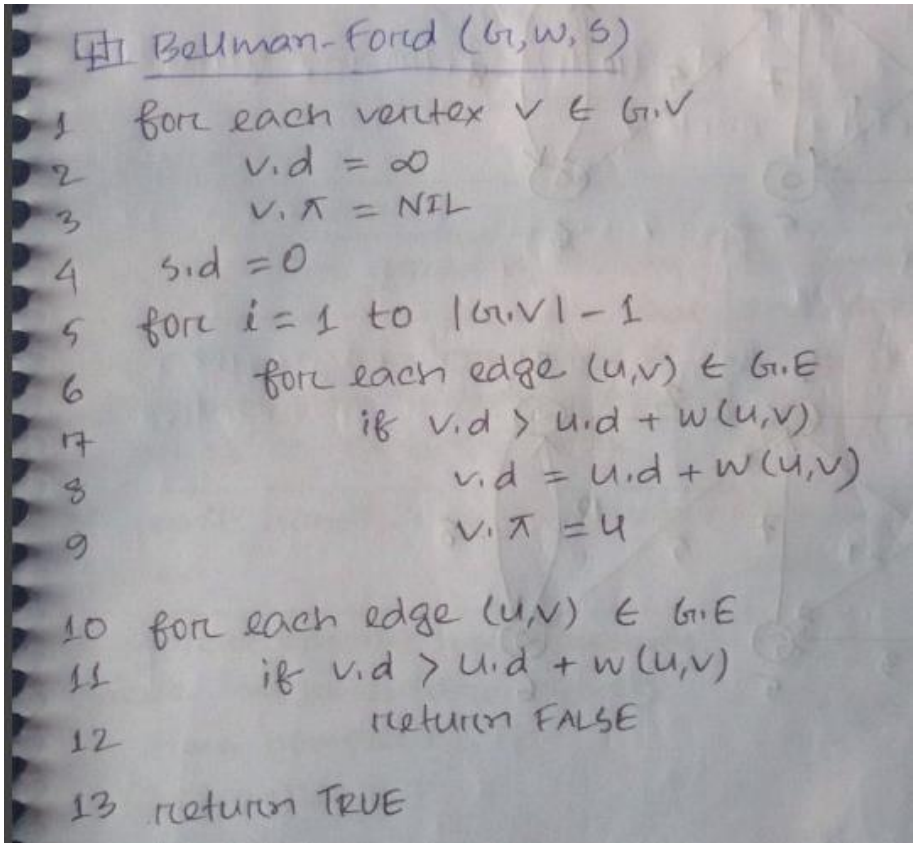


3(b):Describe the Bellman-Ford algorithm with necessary figure.

Answer:

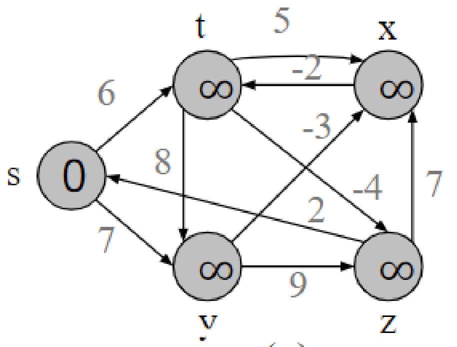
The Bellman-Ford algorithm is a popular algorithm used to find the shortest paths from a source vertex to all other vertices in a weighted directed graph. It can handle negative edge weights, unlike Dijkstra's algorithm. The algorithm iteratively relaxes the edges of the graph, updating the distance estimates until it finds the shortest path.

The algorithm works as-

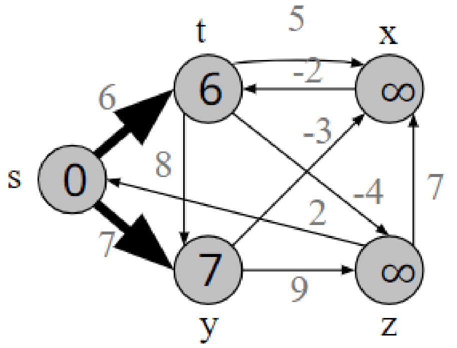


```
[4] Bellman-Ford (G, w, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
for i = 1 to |G.V| - 1
    for each edge (u,v) ∈ G.E
        if v.d > u.d + w(u,v)
            v.d = u.d + w(u,v)
            v.π = u

for each edge (u,v) ∈ G.E
    if v.d > u.d + w(u,v)
        return FALSE

return TRUE
```

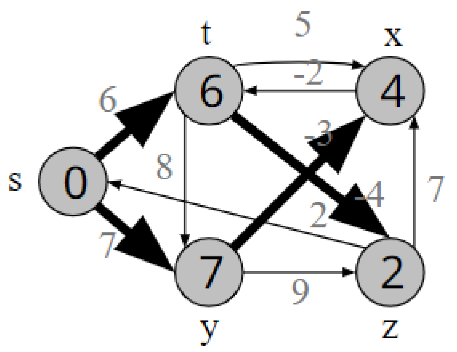
Here is an illustration of the algorithm-



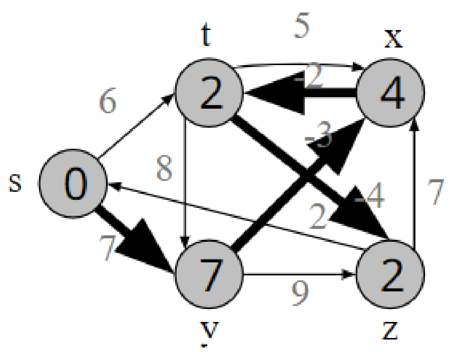
(a)



(b)

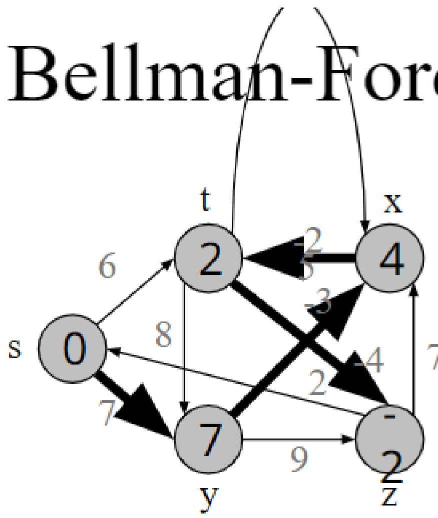


(c)



(d)

Bellman-Ford Algorithm (cont.)



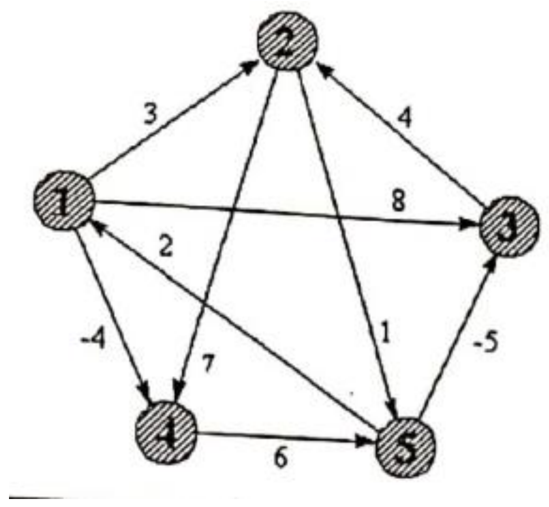
(e)

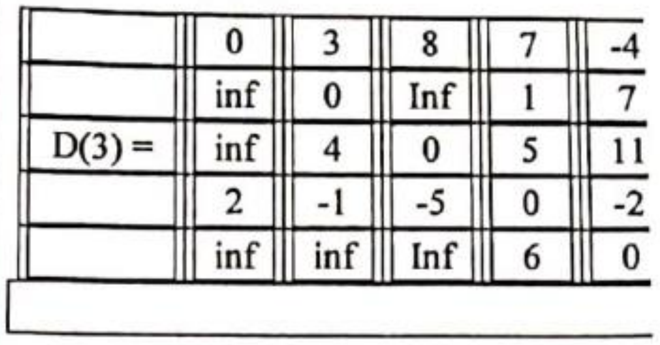
d and π values in (e) are the final values

- Bellman-Ford running time:
  - (|V|-1)|E| + |E| = Θ(VE)

OR

3(b): Consider the following graph for finding all pair shortest path using Floyd-Warshall algorithm.





	0	3	8	7	-4
	inf	0	Inf	1	7
D(3) =	inf	4	0	5	11
	2	-1	-5	0	-2
	inf	inf	Inf	6	0

What is the value of matrix D(4) calculated from matrix D(3) given above.

<u>Answer:</u>

The given matrix of $D(3)$

$$D(3) = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & 7 & -4 \\ 2 & \text{inf} & 0 & \text{inf} & 1 & 7 \\ 3 & \text{inf} & 4 & 0 & 5 & 11 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & \text{inf} & \text{inf} & \text{inf} & 6 & 0 \end{array}$$

here,

$D^3[1,2] < D^3[1,4] + D^3[4,2]$

$3 < 7 + (-1)$

$3 < 6$

$\therefore D^4[1,2] = 3.$

$D^3[1,3] \quad D^3[1,4] + D^3[4,3]$

$8 > 7 + (-5)$

$8 > 2$

$\therefore D^4[1,3] = 2.$

$D^3[1,5] < D^3[1,4] + D^3[4,5]$

$-4 < 7 + (-2)$

$-4 < 5$

$\therefore D^4[1,5] = -4.$

$D^3[2,1] > D^3[2,4] + D^3[4,1]$

$\inf > 1 + 2$

$\inf > 3$

$\therefore D^4[2,1] = 3.$

$D^3[2,3] > D^3[2,4] + D^3[4,3]$

$\inf > 1 + (-5)$

$\inf > -4.$

$\therefore D^4[2,3] = -4.$

$D^3[2,5] > D^3[2,4] + D^3[4,5]$

$7 > 1 + (-2)$

$7 > -1$

$\therefore D^4[2,5] = -1$

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

$\inf > 5 + 2$

$\inf > 7$

$D^4[3,1] = 7$

$D^3[3,2] = D^3[3,4] + D^3[4,2]$

$4 = 5 + (-1)$

4

$\therefore D^4[3,2] = 4$

$D^3[3,5] > D^3[3,4] + D^3[4,5]$

$11 > 5 + (-2)$

$11 > 3.$

$\therefore D^4[3,5] = 3.$

$D^3[5,1] > D^3[5,4] + D^3[4,1]$

$\inf > 6 + 2$

$\inf > 8$

$\therefore D^4[5,1] = 8.$

$D^3[5,2] > D^3[5,4] + D^3[4,2]$

$\inf > 6 + (-1)$

$\inf > 5$

$\therefore D^4[5,2] = 5$

$D^3[5,3] > D^3[5,4] + D^3[4,3]$

$\inf > 6 + (-5)$

$\inf > 1$

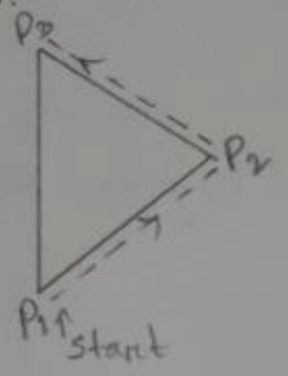
$\therefore D^4[5,3] = 1$

$$\therefore D_4 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 2 & 7 & -4 \\ 2 & 3 & 0 & -4 & 1 & -1 \\ 3 & 7 & 4 & 0 & 5 & 3 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & 8 & 5 & 1 & 6 & 0 \end{array}$$

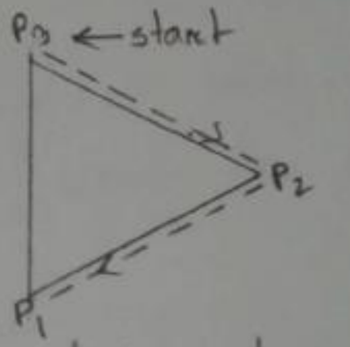
4(a): Let P$_1$(x$_1$,y$_2$), P$_2$(x$_1$,y$_2$) and P$_3$(x$_1$,y$_2$) are three points. Show using the concept of that if going through P$_1$,P$_2$,P$_3$ has a left turn at P$_2$ then going through P$_3$,P$_2$,P$_1$ has a right turn at P$_2$. You have to show it using the concept of cross product.

Answer:

⇒ let $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, and $P_3(x_3,y_3)$ are three points.



Left turn at P_2



Right turn at P_2

$P_2' = P_2 - P_1 = (x_2 - x_1, y_2 - y_1)$

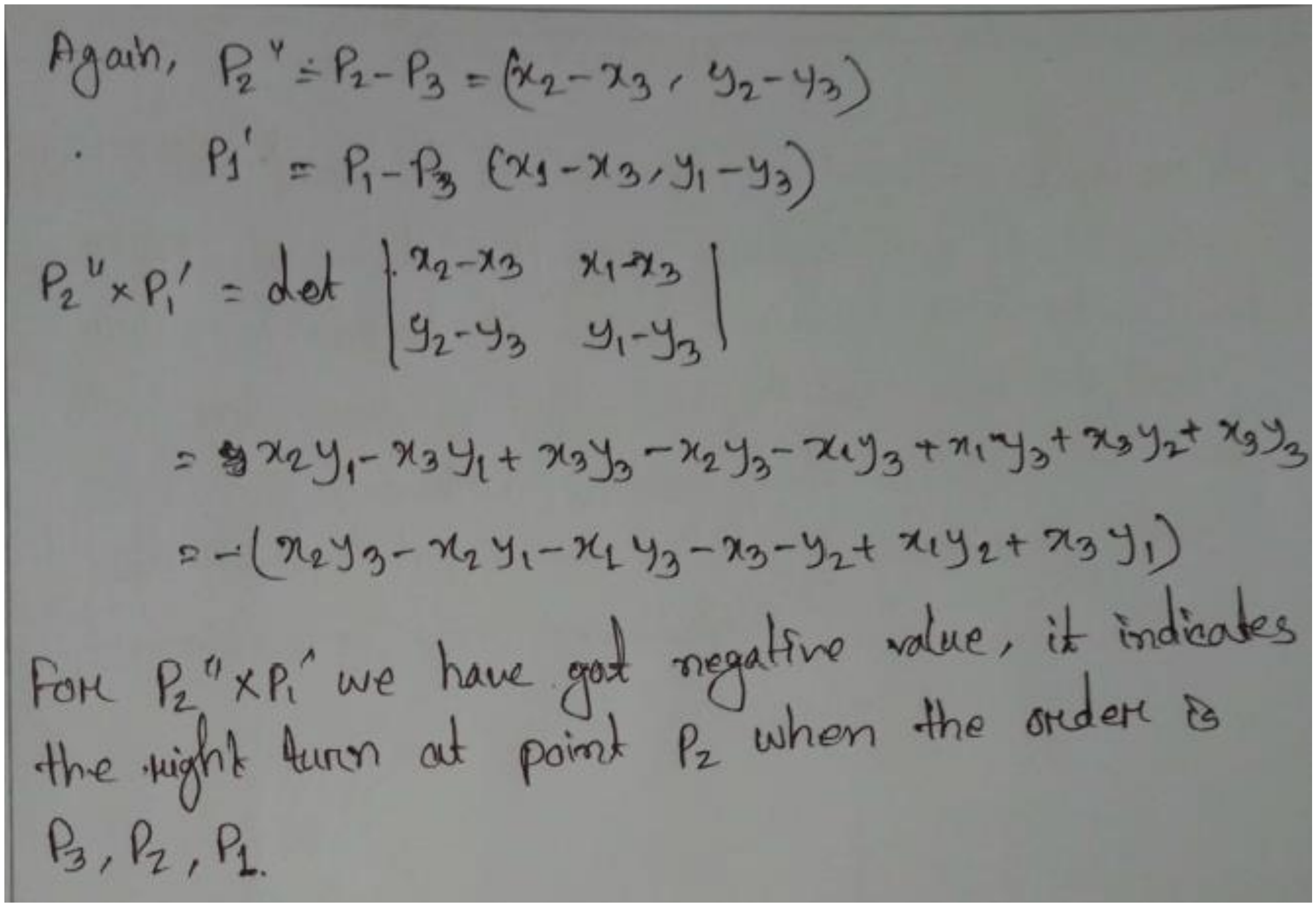
$P_3' = P_3 - P_1 = (x_3 - x_1, y_3 - y_1)$

$$P_2' \times P_3' = \det \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

$$= x_2y_3 - x_2y_1 - x_1x_3 + x_1y_1 - x_3y_2 + x_1y_2 + x_3y_1 - x_1y$$

$$= (x_2y_3 - x_2y_1 - x_1y_3 - \cancel{y} x_3y_2 + x_1y_2 + x_3y_1)$$

For $P_2' \times P_3'$ we have got positive value which indicates that at point P_2 it turns left when the order is P_1, P_2, P_3



Again, $P_2'' = P_2 - P_3 = (x_2 - x_3, y_2 - y_3)$

$P_1' = P_1 - P_3 \, (x_1 - x_3, y_1 - y_3)$

$$P_2'' \times P_1' = \det \begin{vmatrix} x_2 - x_3 & x_1 - x_3 \\ y_2 - y_3 & y_1 - y_3 \end{vmatrix}$$

$$= x_2 y_1 - x_3 y_1 + x_3 y_3 - x_2 y_3 - x_1 y_3 + x_1 y_3 + x_3 y_2 + x_3 y_3$$

$$= -(x_2 y_3 - x_2 y_1 - x_1 y_3 - x_3 - y_2 + x_1 y_2 + x_3 y_1)$$

For $P_2'' \times P_1'$ we have got negative value, it indicates the right turn at point P_2 when the order is P_3, P_2, P_1.

OR

4(a):Suppose you are given two points $P_1(3,7)$ and $P_2(9,2)$. Determine which one has greater polar angle with respect to

i) **Origin(0,0)**
ii) **(20,0)**

__Answer:__

i. __With respect to the origin (0,0):__
For point P1(3,7):
Polar angle of P1 = $\tan^{-1}$ (7/3) ≈ 66.801 degree.

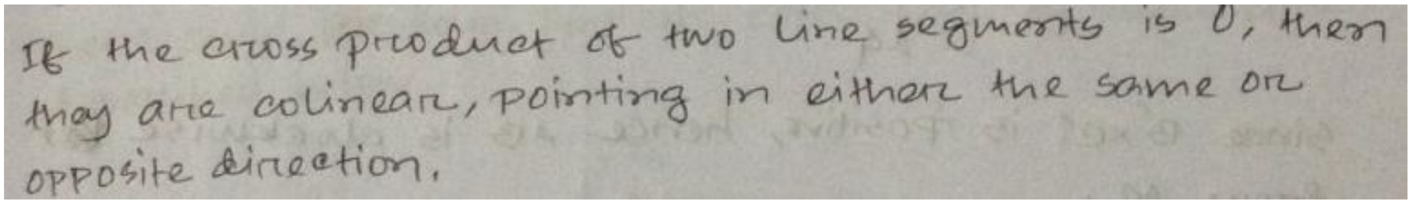
For point P2(9,2)
Polar angle of P2 = $\tan^{-1}$ (2/9) ≈ 12.528 degree.

ii. __With respect to the origin (0,0):__
For point P1(3,7):
Shifted coordinates: P1' = (3-20, 7-0) = (-17, 7)
ϴ1= (7/-17) = -22.380 degree.

For point P2(9,2):
Shifted coordinates: P2' = (9-20, 2-0) = (-11, 2)
ϴ2= (2/-11) = -10.304 degree.

4(b): How can you determine whether two line segments are collinear or not? Describe the basic idea using figure.

Answer:



If the cross product of two line segments is 0, then they are colinear, pointing in either the same or opposite direction.

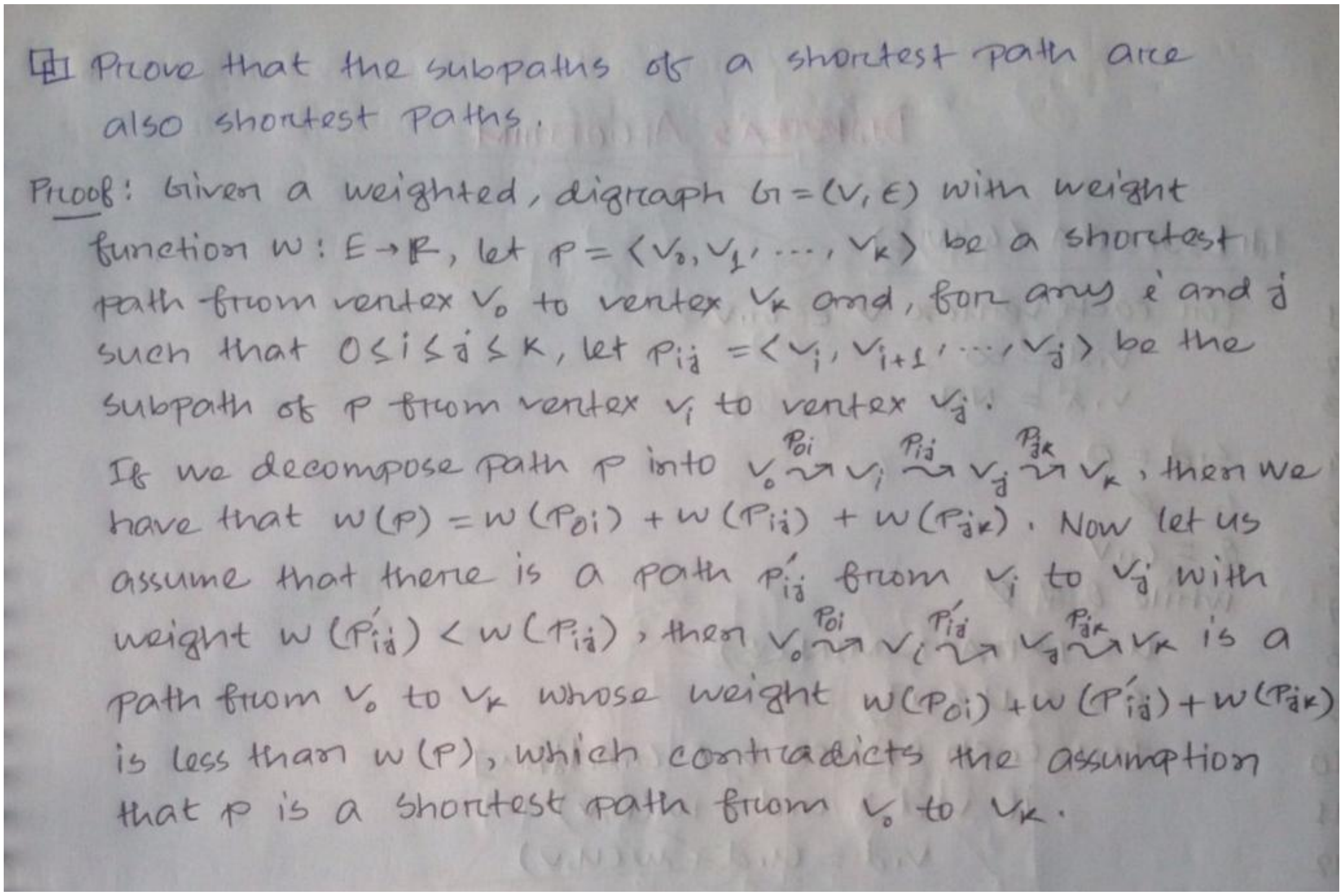
OR

4(b): Given three points on a 2D plane, How can you determine whether they form a triangle or not using concept of cross product?

Answer:

4(c): Prove that the subpaths of shortest paths are also shortest paths.

Answer:



⊞ Prove that the subpaths of a shortest path are also shortest paths.

Proof: Given a weighted, digraph $G=(V,E)$ with weight function $w: E \to \mathbb{R}$, let $p=\langle v_0, v_1, \cdots, v_k \rangle$ be a shortest path from vertex v_0 to vertex v_k and, for any i and j such that $0 \le i \le j \le k$, let $p_{ij} = \langle v_i, v_{i+1}, \cdots, v_j \rangle$ be the subpath of p from vertex v_i to vertex v_j.

If we decompose path p into $v_0 \overset{p_{0i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$, then we have that $w(p) = w(p_{0i}) + w(p_{ij}) + w(p_{jk})$. Now let us assume that there is a path p'_{ij} from v_i to v_j with weight $w(p'_{ij}) < w(p_{ij})$, then $v_0 \overset{p_{0i}}{\rightsquigarrow} v_i \overset{p'_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$ is a path from v_0 to v_k whose weight $w(p_{0i}) + w(p'_{ij}) + w(p_{jk})$ is less than $w(p)$, which contradicts the assumption that p is a shortest path from v_0 to v_k.

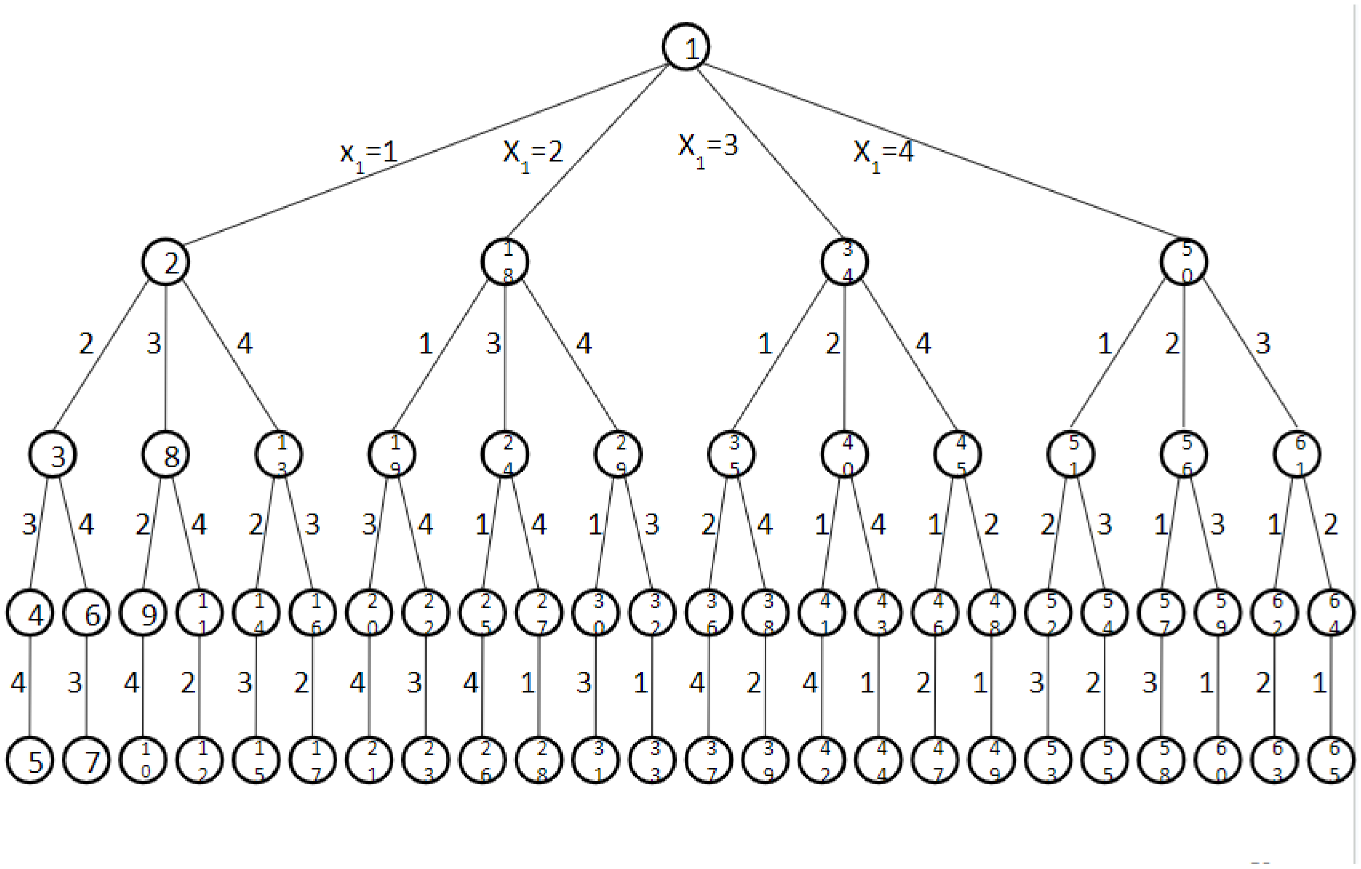
5(a): Consider the 4-queen problem. Your task is to place the 4 queen in 4x4 chessboard so that no two queens attack each other.

i. **What is dead node in N-queen problem?**
ii. **Design a state space tree which represents all possible arrangements for 4 non-attacking queens.**

Answer:

i. Dead node is a generated node that is not to be expended any further. In N-queen problem when any queen have the most possibility to be acttacked by the positioning of others queens Dead node arise. By backtracking the problem can be solved.

ii.



All possible arrangements for 4 non-attacking queens: 2,4,1,3.

5(b): Define the following classes: P, NP, NP-complete.

Answer:

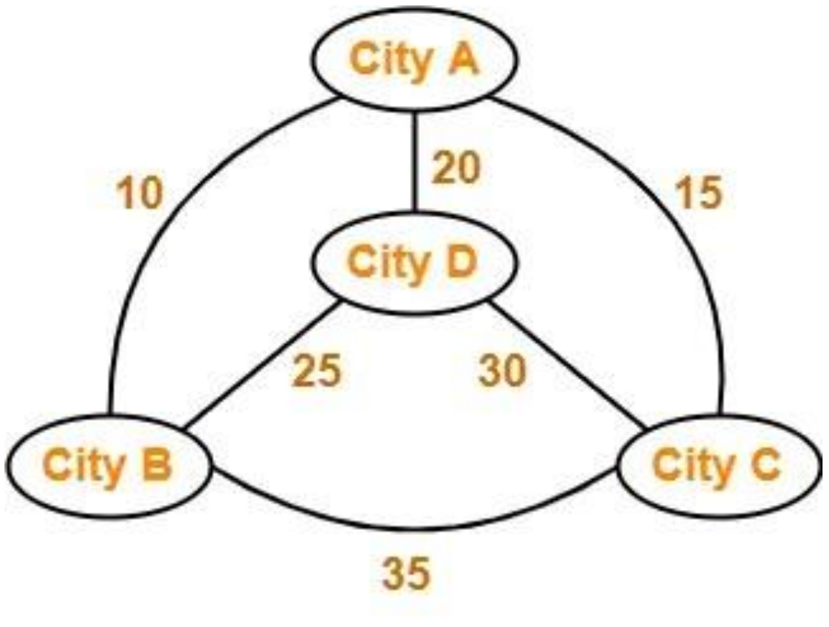
Find the solution in Spring-18 5(c).

5(c): What is the basic principle of a Branch-and -Bound algorithm? Explain how you will apply it in solving travelling salesman problem.

Answer:

Branch-and-Bound

- Effective for optimization problems
- Extended Backtracking Algorithm
- Instead of stopping once a single solution is found, continue searching until the best solution is found
- Has a scoring mechanism to choose most promising configuration in each iteration



Travelling Salesman Problem

If salesman starting city is A, then a TSP tour in the graph is-

A → B → D → C → A

Cost of the tour

= 10 + 25 + 30 + 15

= 80 units

To reduce a matrix, perform the row reduction and column reduction of the matrix separately.

A row or a column is said to be reduced if it contains at least one entry ‘0’ in it.