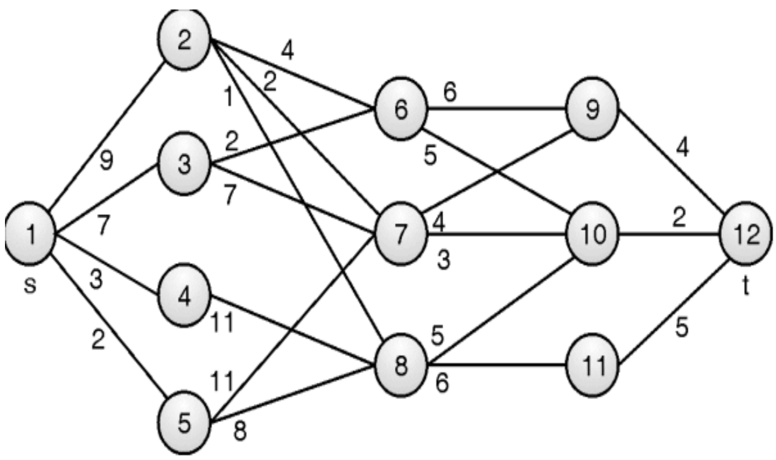
DAA – Question Bank

Unit - 4

Q.No	Questions	Marks

1.	Write an algorithm to find the shortest path in a multi stage graph using dynamic programming using forward approach?
	
	Algorithm FGraph(G, k, n, p)
	// The input is a k-stage graph $G = (V, E)$ with n vertices
	// indexed in order of stages. E is a set of edges and $c[i, j]$
	// is the cost of $\langle i, j\rangle$. $p[1 : k]$ is a minimum-cost path.
	{
	    $cost[n] := 0.0$;
	    for $j := n - 1$ **to** 1 **step** -1 **do**
	    { // Compute $cost[j]$.
	        Let r be a vertex such that $\langle j, r\rangle$ is an edge
	        of G and $c[j, r] + cost[r]$ is minimum;
	        $cost[j] := c[j, r] + cost[r]$;
	        $d[j] := r$;
	    }
	    // Find a minimum-cost path.
	    $p[1] := 1$; $p[k] := n$;
	    for $j := 2$ **to** $k - 1$ **do** $p[j] := d[p[j - 1]]$;
	}
	$cost[n] := 0.0$;
	for $j := n - 1$ **to** 1 **step** -1 **do**
	{ // Compute $cost[j]$.
	    Let r be a vertex such that $\langle j, r\rangle$ is an edge
	    of G and $c[j, r] + cost[r]$ is minimum;
	    $cost[j] := c[j, r] + cost[r]$;
	    $d[j] := r$;
	}
		7M

2.	Find minimum path cost between vertex s and t for following multistage graph using dynamic programming. 	10M

Solution to multistage graph using dynamic programming is constructed as,
Cost[j] = min{c[j, r] + cost[r]}
Here, number of stages k = 5, number of vertices n = 12, source s = 1 and target t = 12
Initialization:

Cost[n] = 0 ⇒ Cost[12] = 0.

p[1] = s ⇒ p[1] = 1

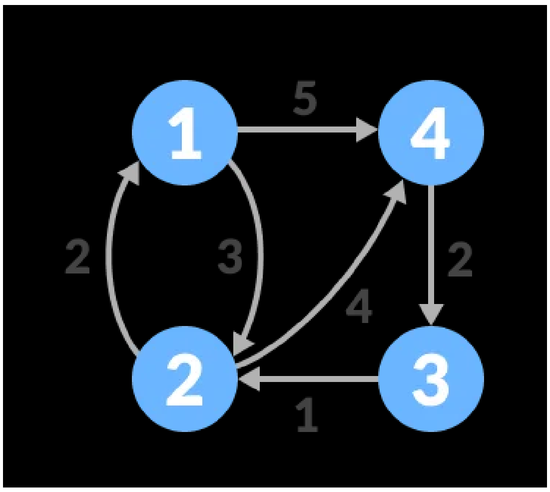
p[k] = t ⇒ p[5] = 12.
r = t = 12.
Stage 4:
cost(5,12)=0
cost(4,9)=c(9,12) = 4 cost(4,10)=c(10,12)=2 cost(4,11)=c(11,12)=5

$$\begin{aligned} cost(3,6) &= \min\ \{6 + cost(4,9), 5 + cost(4,10)\} \\ &= 7 \\ cost(3,7) &= \min\ \{4 + cost(4,9), 3 + cost(4,10)\} \\ &= 5 \end{aligned}$$

$$\begin{aligned} cost(3,8) &= 7 \\ cost(2,2) &= \min\ \{4 + cost(3,6), 2 + cost(3,7), 1 + cost(3,8)\} \\ &= 7 \\ cost(2,3) &= 9 \\ cost(2,4) &= 18 \\ cost(2,5) &= 15 \\ cost(1,1) &= \min\ \{9 + cost(2,2), 7 + cost(2,3), 3 + cost(2,4), \\ &\qquad 2 + cost(2,5)\} \\ &= 16 \end{aligned}$$

Minimum cost path is : 1 – 2 – 7 – 10 – 12
Cost of the path is : 9 + 2 + 3 + 2 = 16

3. Apply Floyd's algorithm for constructing the all pairs shortest path for the following graph. 7M



Answer:

$$A^0 = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

$$A^1 = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & & \\ \infty & & 0 & \\ \infty & & & 0 \end{bmatrix} \end{array} \longrightarrow \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & 9 & 4 \\ \infty & 1 & 0 & 8 \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

$$A^2 = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & & \\ 2 & 0 & 9 & 4 \\ & 1 & 0 & \\ & \infty & & 0 \end{bmatrix} \end{array} \longrightarrow \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 9 & 5 \\ 2 & 0 & 9 & 4 \\ 3 & 1 & 0 & 5 \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

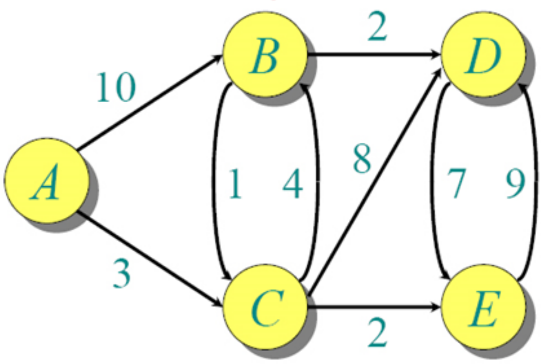
$$A^3 = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & & \infty & \\ & 0 & 9 & \\ \infty & 1 & 0 & 8 \\ & & 2 & 0 \end{bmatrix} \end{array} \longrightarrow \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 9 & 5 \\ 2 & 0 & 9 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{array}$$

<table>
<tr><td></td><td>

$$A^4 = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & & & 5 \\ & 0 & & 4 \\ & & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{array} \rightarrow \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & 7 & 5 \\ 2 & 0 & 6 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{array}$$

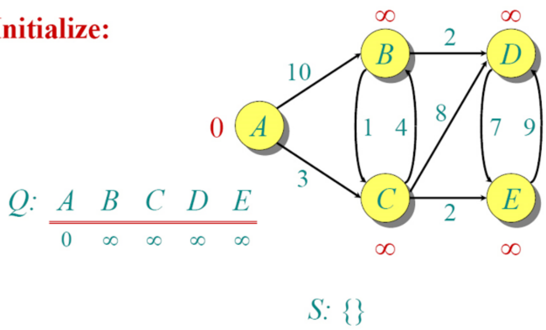
</td><td></td></tr>
<tr><td>4.</td><td>

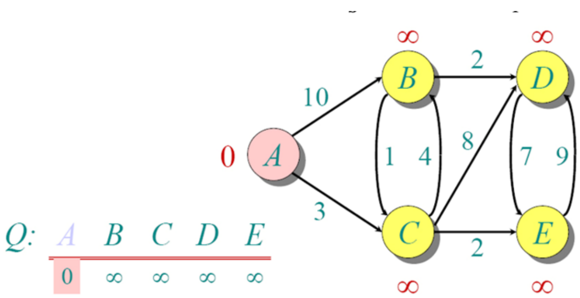
Using Dijkstra's Algorithm, find the shortest distance from source vertex 'A' to remaining vertices in the following graph



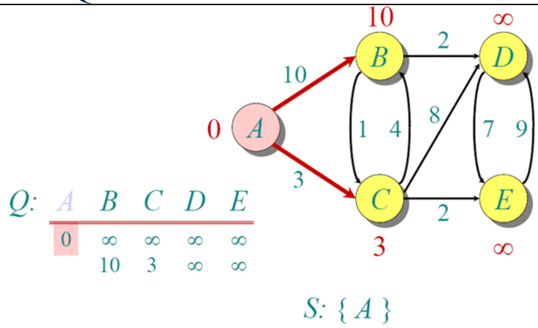
Answer:

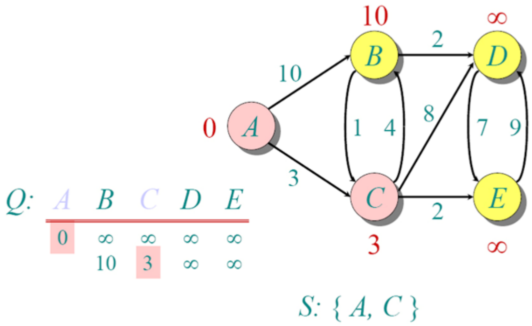
Initialize:

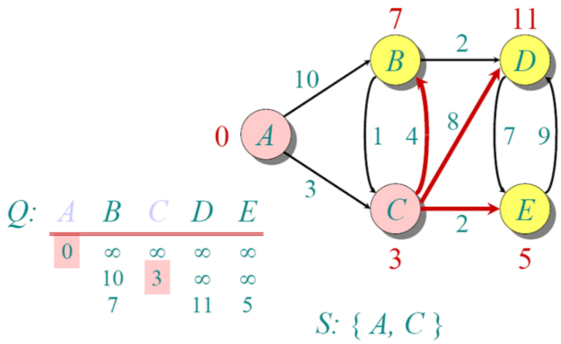


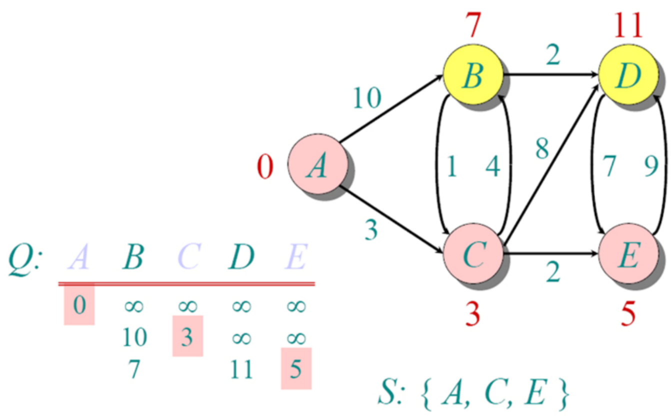


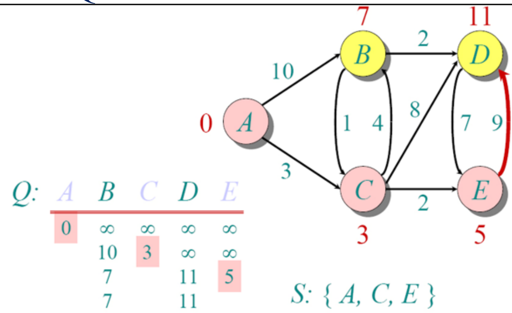
</td><td>10M</td></tr>
</table>

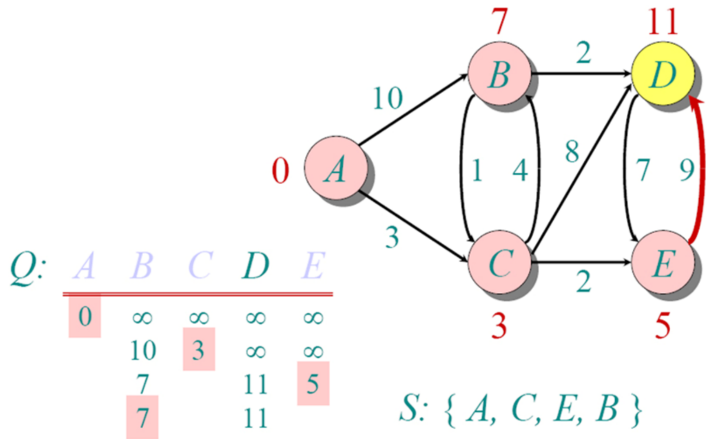

10
∞
2
10
0
A
B
D
1 4
8
7 9
3
C
E
2
3
∞
Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
S: { A }

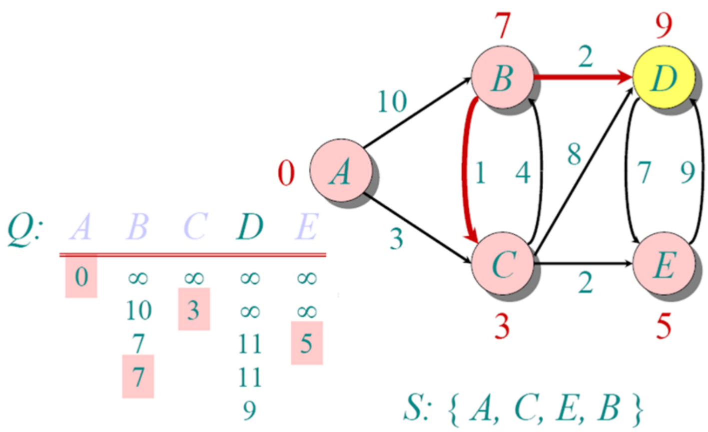

10
∞
2
10
0
A
B
D
1 4
8
7 9
3
C
E
2
3
∞
Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
S: { A, C }

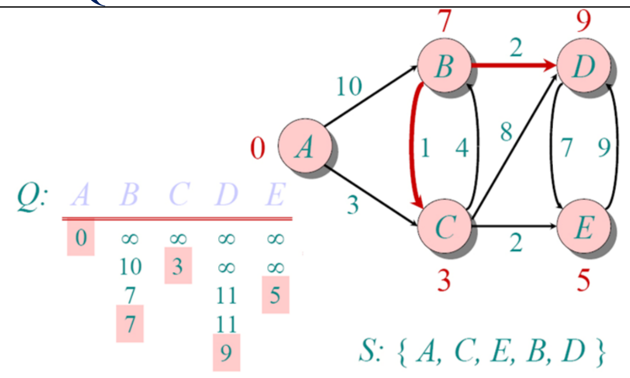

7
11
2
10
0
A
B
D
1 4
8
7 9
3
C
E
2
3
5
Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
7 11 5
S: { A, C }


7
11
2
10
0
A
B
D
1 4
8
7 9
3
C
E
2
3
5
Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
7 11 5
S: { A, C, E }


Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
7 11 5
7 11
S: { A, C, E }


Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
7 11 5
7 11
S: { A, C, E, B }


Q: A B C D E
0 ∞ ∞ ∞ ∞
10 3 ∞ ∞
7 11 5
7 11
9
S: { A, C, E, B }



Shortest Path: A,C,E,B,D

5. Write an algorithm to find the single source shortest paths in a dynamic programming using Dijkstra's Algorithm? — **4M**

```
dist[s] ←0                              (distance to source vertex is zero)
for  all v ∈ V–{s}
        do  dist[v] ←∞                  (set all other distances to infinity)
S←∅                                     (S, the set of visited vertices is initially empty)
Q←V                                     (Q, the queue initially contains all vertices)

while Q ≠∅                              (while the queue is not empty)
do   u ← mindistance(Q,dist)            (select the element of Q with the min.
distance)
      S←S∪{u}                           (add u to list of visited vertices)
       for all v ∈ neighbors[u]
              do  if   dist[v] > dist[u] + w(u, v)        (if new shortest path found)
                          then      d[v] ←d[u] + w(u, v)  (set new value of shortest
path)
                    (if desired, add traceback code)
return dist
```

6. Construct an optimal binary search tree over five key values k1 < k2 < k3 < k4 < k5 with access probability 0.3, 0.2, 0.1, 0.15, and 0.25, respectively. — **14M**

Solution

	j=0	1	2	3	4	5
i=1	0	0.3				
2		0	0.2			
3			0	0.1		
4				0	0.15	
5					0	0.25
6						0

Cost Table:

	j=0	1	2	3	4	5
i=1	0	0.3				
2		0	0.2			
3			0	0.1		
4				0	0.15	
5					0	0.25
6						0

Root:

	j=0	1	2	3	4	5
i=1		1				
2			2			
3				3		
4					4	
5						5
6						

Cost Table:

	j=0	1	2	3	4	5
i=1	0	0.3	0.7			
2		0	0.2	0.4		
3			0	0.1	0.35	
4				0	0.15	0.55
5					0	0.25
6						0

Root:

	j=0	1	2	3	4	5
i=1		1	1			
2			2	2		
3				3	4	
4					4	5
5						5
6						

Cost:

	j=0	1	2	3	4	5
i=1	0	0.3	0.7	1		
2		0	0.2	0.4	0.8	
3			0	0.1	0.35	0.85
4				0	0.15	0.55
5					0	0.25
6						0

Root:

	j=0	1	2	3	4	5
i=1		1	1	2		
2			2	2	3	
3				3	4	5
4					4	5
5						5
6						

Cost:

	j=0	1	2	3	4	5
i=1	0	0.3	0.7	1	1.4	
2		0	0.2	0.4	0.8	1.35
3			0	0.1	0.35	0.85
4				0	0.15	0.55
5					0	0.25
6						0

Root:

	j=0	1	2	3	4	5
i=1		1	1	2	2	
2			2	2	3	4
3				3	4	5
4					4	5
5						5
6						

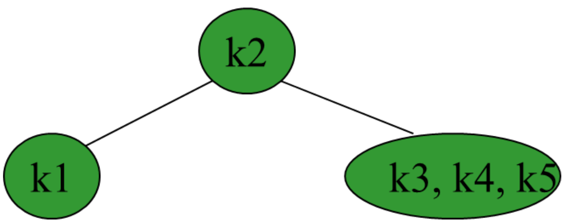
Cost:

	j=0	1	2	3	4	5
i=1	0	0.3	0.7	1	1.4	2.15
2		0	0.2	0.4	0.8	1.35
3			0	0.1	0.35	0.85
4				0	0.15	0.55
5					0	0.25
6						0

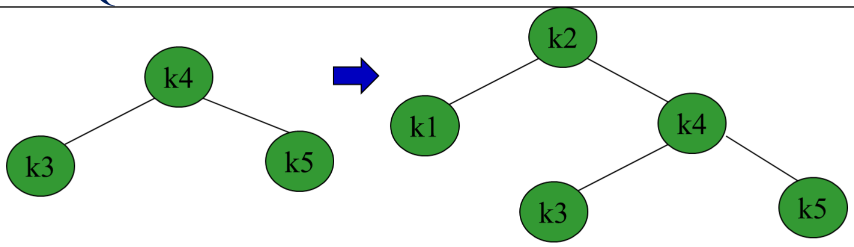
Root:

	j=0	1	2	3	4	5
i=1		1	1	1	2	2
2			2	2	2	4
3				3	4	4
4					4	5
5						5
6						

r[1, 5] = 2 shows that the root of the tree over k1, k2, k3, k4, k5 is k2



r[3, 5] = 4 shows that the root of the subtree over k3, k4, k5 is k4.



7. Find an optimal solution for following 0/1 Knapsack problem using dynamic programming: Number of objects n = 4, Knapsack Capacity M = 5, Weights (W1, W2, W3, W4) = (2, 3, 4, 5) and profits (P1, P2, P3, P4) = (3, 4, 5, 6). **14M**

Solution

Solution of the knapsack problem is defined as

Item	Weight (w_i)	Value (v_i)
I_1	2	3
I_2	3	4
I_3	4	5
I_4	5	6

$$V[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ V[i-1, j] & \text{if } j < wi \\ \max\{V[i-1, j], vi + V[i-1, j-wi]\} & \text{if } j \geq w \end{cases}$$

Filling first column, j = 1

V [1, 1] ⇒ i = 1, j = 1, wi = w1 = 2

As, j < wi, V [i, j] = V [i – 1, j]

V [1, 1] = V [0, 1] = 0

V [2, 1] ⇒ i = 2, j = 1, wi = w2 = 3

As, j < wi, V [i, j] = V [i – 1, j]

V [2, 1] = V [1, 1] = 0

V [3, 1] ⇒ i = 3, j = 1, wi = w3 = 4

As, j < wi, V [i, j] = V [i – 1, j]

V [3, 1] = V [2, 1] = 0

V [4, 1] ⇒ i = 4, j = 1, wi = w4 = 5

As, j < wi, V[i, j] = V[i – 1, j]

	V [4, 1] = V [3, 1] = 0 Filling first column, j = 2 V[1, 2] ⇒ i = 1, j = 2, wi = w1 = 2, vi = 3 As, j ≥ wi, V [i, j]=max {V [i – 1, j], vi + V[i – 1, j – wi] } = max {V [0, 2], 3 + V [0, 0]} V[1, 2] = max (0, 3) = 3 V[2, 2] ⇒ i = 2, j = 2, wi = w2 = 3, vi = 4 As, j < wi, V [i, j] = V[i – 1, j] V[2, 2] = V[1, 2] = 3 V[3, 2] ⇒ i = 3, j = 2, wi = w3 = 4, vi = 5 As, j < wi, V[i, j] = V[i – 1, j] V[3, 2] = V [2, 2] = 3 V[4, 2] ⇒ i = 4, j = 2, wi = w4 = 5, vi = 6 As, j < wi, V[i, j] = V[i – 1, j] V[4, 2] = V[3, 2] = 3 Filling first column, j = 3 V[1, 3] ⇒ i = 1, j = 3, wi = w1 = 2, vi = 3 As, j ≥ wi, V [i, j]=max {V [i – 1, j], vi + V [i – 1, j – wi] } = max {V [0, 3], 3 + V [0, 1]} V[1, 3] = max (0, 3) = 3 V[2, 3] ⇒ i = 2, j = 3, wi = w2 = 3, vi = 4 As, j ≥ wi, V [i, j] = max {V [i – 1, j], vi + V [i – 1, j – wi] } = max {V [1, 3], 4 + V [1, 0]} V[2, 3] = max (3, 4) = 4 V[3, 3] ⇒ i = 3, j = 3, wi = w3 = 4, vi = 5 As, j < wi, V [i, j] = V [i – 1, j] V[3, 3] = V [2, 3] = 4 V[4, 3] ⇒ i = 4, j = 3, wi = w4 = 5, vi = 6 As, j < wi, V[i, j] = V[i – 1, j] V[4, 3] = V [3, 3] = 4 Filling first column, j = 4 V[1, 4] ⇒ i = 1, j = 4, wi = w1 = 2, vi = 3 As, j ≥ wi, V [i, j]=max {V [i – 1, j], vi + V [i – 1, j – wi] } = max {V [0, 4], 3 + V [0, 2]} V[1, 4] = max (0, 3) = 3 V[2, 4] ⇒ i = 2, j = 4, wi = w2 = 3 , vi = 4 As, j ≥ wi, V [i, j] =max {V [i – 1, j], vi + V [i – 1, j – wi] } = max {V [1, 4], 4 + V [1, 1]} V[2, 4] = max (3, 4 + 0) = 4 V[3, 4] ⇒ i = 3, j = 4, wi = w3 = 4, vi = 5 As, j ≥ wi, V [i, j]=max {V [i – 1, j], vi + V [i – 1, j – wi] } = max {V [2, 4], 5 + V [2, 0]} V[3, 4] = max (4, 5 + 0) = 5 V[4, 4] ⇒ i = 4, j = 4, wi = w4 = 5, vi = 6 As, j < wi, V [i, j] = V [i – 1, j] V[4, 4] = V [3, 4] = 5	

, j – wi] }

, j – wi] }

, j – wi] }

, j – wi] }

i = 3	w_3= 4	v_3 =5	0	0	3	4	5	7
i = 4	w_4= 5	v_4 =6	0	0	3	4	5	7

Find selected items for M = 5

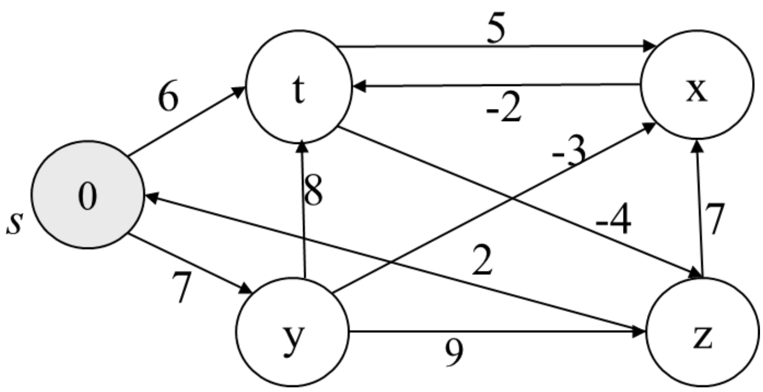
	Step 1 : Initially, i = n = 4, j = M = 5 V[i, j] = V[4, 5] = 7 V[i – 1, j] = V[3, 5] = 7 V[i, j] = V[i – 1, j], so don't select ith item and check for the previous item. so i = i – 1 = 4 – 1 = 3 Solution Set S = { } Step 2 : i = 3, j = 5 V[i, j] = V[3, 5] = 7 V[i – 1, j] = V[2, 5] = 7 V[i, j] = V[i – 1, j], so don't select ith item and check for the previous item. so i = i – 1 = 3 – 1 = 2 Solution Set S = { } Step 3 : i = 2, j = 5 V[i, j] = V[2, 5] = 7 V[i – 1, j] = V[1, 5] = 3 V[i, j] ≠ V[i – 1, j], so add item Ii = I2 in solution set. Reduce problem size j by wi j = j – wi = j – w2 = 5 – 3 = 2 i = i – 1 = 2 – 1 = 1 Solution Set S = {I2} Step 4 : i = 1, j = 2 V[1, j] = V[1, 2] = 3 V[i – 1, j] = V[0, 2] = 0 V[i, j] ≠ V[i – 1, j], so add item Ii = I1 in solution set. Reduce problem size j by wi j = j – wi = j – w1 = 2 – 2 = 0 Solution Set S = {I1, I2} **Problem size has reached to 0, so final solution is** **S = {I1, I2} Earned profit = P1 + P2 = 7**	

8.	Write an algorithm to find the shortest path in a Bellman-Ford Algorithm using dynamic programming.
	Bellman-Ford(G, w, s)
	    Initialize-Single-Source(G, s)
	    for i := 1 to |V| - 1 do
	        for each edge (u, v) ∈ E do
	            Relax(u, v, w)
	    for each vertex v ∈ u.adj do
	        if d[v] > d[u] + w(u, v)
	            then return False // there is a negative cycle
		4M

return True

Relax(u, v, w)
if d[v] > d[u] + w(u, v)
then d[v] := d[u] + w(u, v)
parent[v] := u

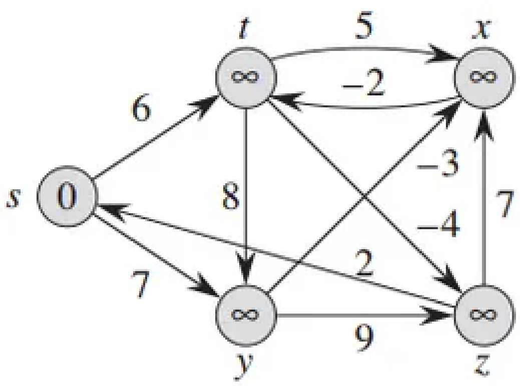
9. Consider the weighted graph below. Find out the minimum shortest path using bellman ford algorithm. Find out whether the given graph is negative cycle or without negative cycle. 10M

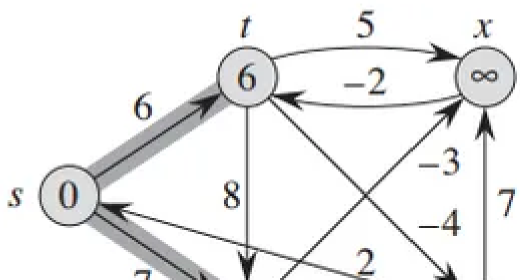


Solution:

Assume that S is our starting vertex. We're now ready to start with the initialization step of the algorithm:
Relaxing all edges n-1 times, (where n is no. of vertex)
Relaxation: if (d[u]+c[u,v]<d[v])
d[v]=d[u]+c[u,v]

Edge List: (s,t) = 6 (y,x) = -3 (s,y)= 7 (y,z) = 9 (t,y) = 8 (x,t) = -2
(t,z) = -4 (z,x) = 7 (t,x) = 5 (z,s) = 2





	Until now 4 iterations completed and shortest path found to every node form source node. Now we have to do one more iteration to find whether there exists negative edge cycle or not. When we do this nth (5th here) relaxation if we found less distance to any vertex from any other path we can say that there is negative edge cycle. Here we can relax any edge to graph which obtained in iteration 4and we can observe that there is no chance to change those values. So we can confirm that there is no negative edge cycle in this graph.	

	Solve the traveling salesman problem by given graph using dynamic programming.	10M

Solution:

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$S = \Phi$

$Cost(2,\Phi,1)=d(2,1)=5Cost(2,\Phi,1)=d(2,1)=5$

$Cost(3,\Phi,1)=d(3,1)=6Cost(3,\Phi,1)=d(3,1)=6$

$Cost(4,\Phi,1)=d(4,1)=8Cost(4,\Phi,1)=d(4,1)=8$

$S = 1$

$Cost(i,s)=min\{Cost(j,s–(j))+d[i,j]\}Cost(i,s)=min\{Cost(j,s)−(j))+d[i,j]\}$

$Cost(2,\{3\},1)=d[2,3]+Cost(3,\Phi,1)=9+6=15cost(2,\{3\},1)=d[2,3]+cost(3,\Phi,1)=9+6=15$

$Cost(2,\{4\},1)=d[2,4]+Cost(4,\Phi,1)=10+8=18cost(2,\{4\},1)=d[2,4]+cost(4,\Phi,1)=10+8=18$

$Cost(3,\{2\},1)=d[3,2]+Cost(2,\Phi,1)=13+5=18cost(3,\{2\},1)=d[3,2]+cost(2,\Phi,1)=13+5=18$

$Cost(3,\{4\},1)=d[3,4]+Cost(4,\Phi,1)=12+8=20cost(3,\{4\},1)=d[3,4]+cost(4,\Phi,1)=12+8=20$

$Cost(4,\{3\},1)=d[4,3]+Cost(3,\Phi,1)=9+6=15cost(4,\{3\},1)=d[4,3]+cost(3,\Phi,1)=9+6=15$

$Cost(4,\{2\},1)=d[4,2]+Cost(2,\Phi,1)=8+5=13cost(4,\{2\},1)=d[4,2]+cost(2,\Phi,1)=8+5=13$

$S = 2$

$Cost(2,\{3,4\},1)= d[2,3]+Cost(3,\{4\},1)=9+20=29$

$d[2,4]+Cost(4,\{3\},1)=10+15=25=25Cost(2,\{3,4\},1) = 25$

$\{d[2,3]+cost(3,\{4\},1)=9+20=29d[2,4]+Cost(4,\{3\},1)=10+15=25=25$

	Cost(3,{2,4},1)= d[3,2]+Cost(2,{4},1)=13+18=31 d[3,4]+Cost(4,{2},1)=12+13=25=25Cost(3,{2,4},1) = 25 {d[3,2]+cost(2,{4},1)=13+18=31d[3,4]+Cost(4,{2},1)=12+13=25=25 Cost(4,{2,3},1)= d[4,2]+Cost(2,{3},1)=8+15=23d[4,3]+Cost(3,{2},1)=9+18=27=23Cost(4,{2,3},1){d[4,2]+cost(2,{3},1)=8+15=23d[4,3]+Cost(3,{2},1)=9+18=27=23 S = 3 Cost(1,{2,3,4},1)= d[1,2]+Cost(2,{3,4},1)=10+25=35 d[1,3]+Cost(3,{2,4},1)=15+25=40 d[1,4]+Cost(4,{2,3},1)=20+23=43=35cost(1,{2,3,4}),1) d[1,2]+cost(2,{3,4},1)=10+25=35 d[1,3]+cost(3,{2,4},1)=15+25=40 d[1,4]+cost(4,{2,3},1)=20+23=43=35	
11.	Write an algorithm to find the shortest path in a Traveling-Salesman-Problem using dynamic programming. Solution: Algorithm: Traveling-Salesman-Problem C ({1}, 1) = 0 for s = 2 to n do for all subsets S Є {1, 2, 3, … , n} of size s and containing 1 C (S, 1) = ∞ for all j Є S and j ≠ 1 C (S, j) = min {C (S – {j}, i) + d(i, j) for i Є S and i ≠ j} Return minj C ({1, 2, 3, …, n}, j) + d(j, i)	4M