FOURTH SEMESTER B.E DEGREE EXAMINATION

DEC 2019 / JAN 2020

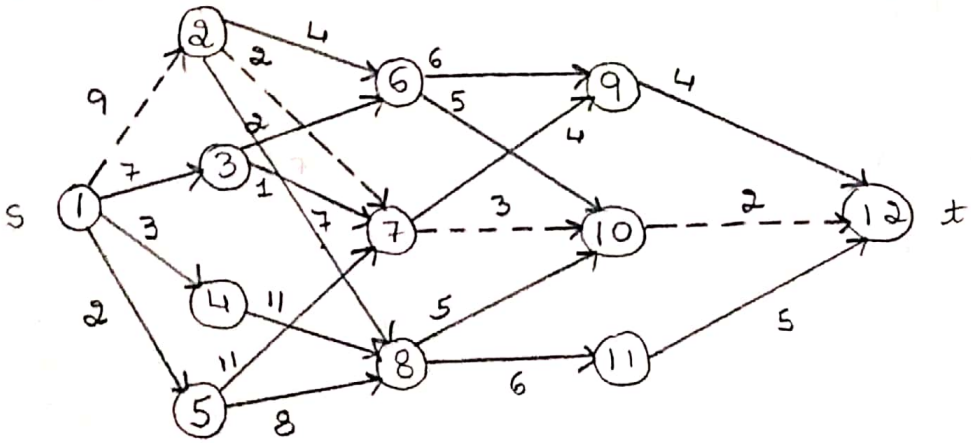
Design and Analysis of Algorithms

Module-04

7 a) Explain the general procedure to solve a multistage graph problem using backward approach with an example. (10 marks)

→ A multistage graph $G=(V, E)$ is a directed graph in which the vertices are partitioned into $k \geq 2$ disjoint sets, V_i, $1 \leq i \leq k$.

Ex: V_1 V_2 V_3 V_4 V_5



- In the above diagram s -> source, t -> destination
- $c(i, j)$ -> cost of edges (i, j)
  V_i -> stages in graph
- Minimum cost from s to t = sum of the cost of the edges on the path

cost (ith stage, jth vertex)

$$bcost(i,j) = \min_{\substack{l \in V_{i-1} \\ (l,j) \in E}} \{ bcost(i-1,l) + c(l,j) \}$$

Stage 1 :

$bcost[1] = 0$

Stage 2 :

$bcost(2,2) = \min\{ bcost(1,1) + c(1,2) \} = 9$

$bcost(2,3) = \{ bcost(1,1) + c(1,3) \} = 7$

$bcost(2,4) = \{ bcost(1,1) + c(1,4) \} = 3$

$bcost(2,5) = \{ bcost(1,1) + c(1,5) \} = 2$

Stage 3 :

$bcost(3,6) = \min\{ bcost(2,2) + cost(2,6),$
$bcost(2,3) + cost(3,6) \}$

$= \min\{13, 9\} = 9$

$bcost(3,7) = \min\{ bcost(2,2) + cost(2,7),$
$bcost(2,3) + cost(3,7),$
$bcost(2,5) + cost(5,7) \}$

$= \min\{11, 14, 13\} = 11$

$bcost(3,8) = \min\{ bcost(2,2) + cost(2,8),$
$bcost(2,4) + cost(4,8),$
$bcost(2,5) + cost(5,8) \}$

$= \min\{10, 14, 10\} = 10$

Stage 4:

bcost (4, 9) = Min { bcost (3, 6) + cost (6, 9),
bcost (3, 7) + cost (7, 9) }

= Min { 15, 15 } = 15

bcost (4, 10) = min { bcost (3, 6) + cost (6, 10),
bcost (3, 7) + cost (7, 10),
bcost (3, 8) + cost (8, 10) }

= Min { 14, 14, 15 } = 14

bcost (4, 11) = { bcost (3, 8) + c (8, 11) } = 16

~~bcost (5,~~

Stage 5:

bcost (5, 12) = Min { bcost (4, 9) + c (9, 12),
bcost (4, 10) + c (10, 12),
bcost (4, 11) + c (11, 12) }

= Min { 17, 16, 21 } = 16

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	0	9	7	3	2	9	11	10	13	14	16	16
d	1	1	1	1	1	3	2	2/5	6	6/7	8	10

Shortest path is

p[1] = 1 ; p[5] = 12

p[j] = d[p[j+1]];

p[4] = d[p[5]] = d[12] = 10

P[3] = d[p[4]] = d[10] = 6/7

P[2] = d[p[3]] = d[6] = 3; p[2] = d[p[3]] = d[7] = 2

P[1] = d[p[2]] = d[3] = 1; p[1] = d[p[2]] = d[2] = 1

p	[1]	[2]	[3]	[4]	[5]
	1	3/2	6/7	10	12

-> Path Table

Algorithm for Backward approach :

```
Algorithm BGraph (G, k, n, p)
// Same function as FGraph
{
  bcost[1] := 0.0;
  for j := 2 to n do
  { // Compute bcost[j]
      Let r be such that {r, j} is an edge of
      G and bcost[r] + c[r, j] is minimum;
      bcost[j] := bcost[r] + c[r, j];
      d[j] := r;
  }
  // Find a minimum-cost path
  p[1] := 1; p[k] := n;
  for j := k-1 to 2 do p[j] := d[p[j+1]];
}
```

b. Construct an optimal binary search tree for the following data. (10 marks)

Items	A (1)	B (2)	C (3)	D (4)
Probabilities	0.1	0.2	0.4	0.3

-> At initial

Main Table

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

Root Table

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

We compute each data with the formula

$$C(i,j) = \min_{i \le k \le j} \{C(i,k-1) + C(k+1,j)\} + \sum_{s=i}^{j} P_s \quad \text{for } 1 \le i \le j \le n$$

when $i = 1;\ j = 2;\ k = 1, 2$

$$C(1,2) = \min \begin{cases} k=1;\ C(1,0) + C(2,2) + \sum_{s=1}^{2} P_s = 0 + 0.2 + 0.3 = 0.5 \\ k=2;\ C(1,1) + C(3,2) + \sum_{s=1}^{2} P_s = 0.1 + 0 + 0.3 = 0.4 \end{cases}$$

$= 0.4$

When $i = 2;\ j = 3;\ k = 2, 3$

$$C(2,3) = \min \begin{cases} k=2;\ c(2,1) + c(3,3) + \sum_{s=2}^{3} P_s = 0 + 0.4 + 0.6 = 1.0 \\ k=3;\ C(2,2) + c(4,3) + \sum_{s=2}^{3} P_s = 0.2 + 0 + 0.6 = 0.8 \end{cases}$$

$= 0.8$

When $i = 3$; $j = 4$; $k = 3, 4$

$$c(3,4) = \min \begin{cases} k=3: c(3,2) + c(4,4) + \sum_{s=3}^{4} P_s = 0 + 0.3 + 0.7 = 1.0 \\ k=4: c(3,3) + c(5,4) + \sum_{s=3}^{4} P_s = 0.4 + 0 + 0.7 = 1.1 \end{cases}$$

$$= 1.0$$

When $i = 1$; $j = 3$; $k = 1, 2, 3$

$$c(1,3) = \min \begin{cases} k=1: c(1,0) + c(2,3) + \sum_{s=1}^{3} P_s = 0 + 0.8 + 0.7 = 1.5 \\ k=2: c(1,1) + c(3,3) + \sum_{s=1}^{3} P_s = 0.1 + 0.4 + 0.7 = 1.2 \\ k=3: c(1,2) + c(4,3) + \sum_{s=1}^{3} P_s = 0.4 + 0 + 0.7 = 1.1 \end{cases}$$

$$= 1.1$$

When $i = 2$; $j = 4$; $k = 2, 3, 4$

$$c(2,4) = \min \begin{cases} k=2: c(2,1) + c(3,4) + \sum_{s=2}^{4} P_s = 0 + 1.0 + 0.9 = 1.9 \\ k=3: c(2,2) + c(4,4) + \sum_{s=2}^{4} P_s = 0.2 + 0.3 + 0.9 = 1.4 \\ k=4: c(2,3) + c(5,4) + \sum_{s=2}^{4} P_s = 0.8 + 0 + 0.9 = 1.7 \end{cases}$$

$$= 1.4$$

When $i = 1$; $j = 4$; $k = 1, 2, 3, 4$

$$c(1,4) = \min \begin{cases} k=1: c(1,0) + c(2,4) + \sum_{s=1}^{4} P_s = 0 + 1.0 + 1.0 = 2.0 \\ k=2: c(1,1) + c(3,4) + \sum_{s=1}^{4} P_s = 0.1 + 1.0 + 1.0 = 2.1 \\ k=3: c(1,2) + c(4,4) + \sum_{s=1}^{4} P_s = 0.4 + 0.3 + 1.0 = 1.7 \\ k=4: c(1,3) + c(5,4) + \sum_{s=1}^{4} P_s = 1.1 + 0 + 1.0 = 2.0 \end{cases}$$

$$= 1.7$$

Main Table

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

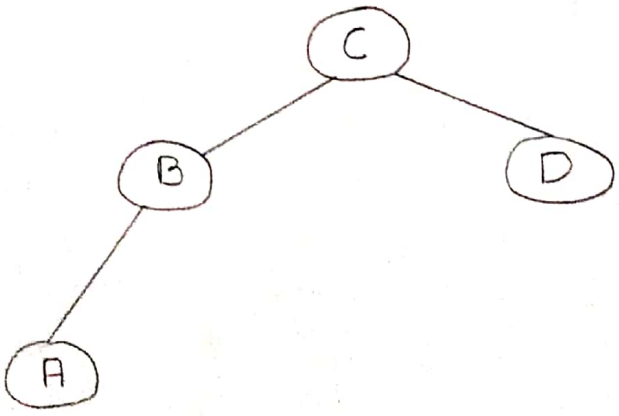
Root Table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

Thus the average number of key comparision in the optimal tree is equal to 1.7.

With the help of root table we can construct the key binary search tree such that

- Root is key no. 3 [R(1,4)=3] i.e., C where as key ~~no~~ 1, is the part of left subtree and key-4 is the part of right subtree.
- Construct subtree between 2 and 1, make 2 as the root and 1 as the left node
- R(1,2) = 2(B) R(1,1) = 1(A) R(4,4) = 4(D)



8 a) Design Floyd's algorithm to find shortest distance from all nodes to all other nodes. (10 marks)

→ ALGORITHM Floyd[W[1..n, 1..n])

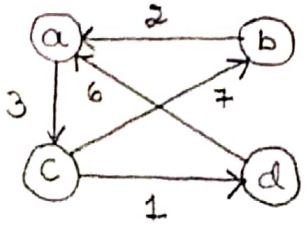
// Implements Floyd's algorithm for all-pairs shortest-
// path problem
// Input: The weight matrix W of a graph with no
// negative-length cycle.
// Output: The distance matrix of the shortest
// paths' lengths.

D ← W // is not necessary if W can be overwritten

for k ← 1 to n do
  for i ← 1 to n do
    for j ← 1 to n do
      D[i, j] ← min{D[i, j], D[i, k] + D[k, j]}

return D

Ex:



The cost adjacency matrix for the above graph is as follows

$D^{(0)}$ =

	a	b	c	d
a	0	∞	3	∞
b	2	0	∞	∞
c	∞	7	0	1
d	6	∞	∞	0

Step1: Consider the shortest path distance through the vertex 'a'.

$D_{ij}(K) = \min \{ D_{ij}^{(K-1)}, D_{iK}^{(K-1)} + D_{Kj}^{(K-1)} \}$

$D_1(b,c) = \min \{ D_0(b,c), D_0(b,a) + D_0(a,c) \}$

$= 5$

$D_1(d,c) = \min \{ D_0(d,c), D_0(d,a) + D_0(a,c) \}$

$= 9$

$$D^{(1)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

~~$D_2(c,a)$~~ Step2: Consider the shortest path distance through the vertex 'b'.

$D_2(c,a) = \min \{ D_1(c,a), D_1(c,b) + D_1(b,a) \}$

$= 9$

$$D^{(2)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ 9 & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

Step 3: Consider the shortest path distance through the vertex 'c'.

$D_3(a,b) = \min \{ D_2(a,b), D_2(a,c) + D_2(c,b) \}$

$= 10$

$D_3(a,d) = \min \{ D_2(a,d), D_2(a,c) + D_2(c,d) \}$

$= 4$

$D_3(d,b) = \min\{D_2(d,b), D_2(d,c) + D_2(c,b)\}$

$= 16$

$D_3(b,d) = \min\{D_2(b,d), D_2(b,c) + D_2(c,d)\}$

$= 6$

$$D^{(3)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

Step 4: Consider the shortest path distance through the vertex 'd'.

$D_4(c,a) = \min\{D_3(c,a), D_2(c,d) + D_2(d,a)\}$

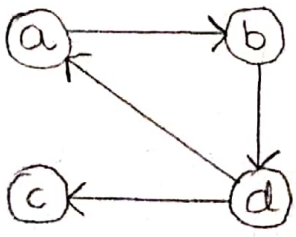
$= 7$

Hence the resultant matrix is

$$D^{(4)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

b) Apply Warshall's algorithm to compute transitive closure for the graph below.

(10 marks)



→ The adjacency matrix for given graph is

$$R^{(0)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Step1 : Consider the path through the vertex 'a'.

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} \;||\; (R_{ik}^{(k-1)} \;\&\&\; R_{kj}^{(k-1)})$$

By comparing all the paths, the changing path is

$$R_1(d,b) = R_0(d,b) \;||\; \{R_0(d,a) \;\&\&\; R_0(a,b)\}$$
$$= 0 \;||\; \{1 \;\&\&\; 1\}$$
$$= 1$$

$$R^{(1)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Step 2: Consider the path through the vertex 'b'.

$$R_2(a,d) = R_1(a,d) \;||\; \{R_1(a,b) \;\&\&\; R_1(b,d)\}$$
$$= 1$$

$$R_2(d,d) = R_1(d,d) \;||\; \{R_1(d,b) \;\&\&\; R_1(b,d)\}$$
$$= 1$$

$$R^{(2)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Step 3: Consider the path through the vertex 'c'.

By comparing all the paths, we can observe that their is no changing path exists.

$$R^{(3)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Step 4: Consider the path through the vertex 'd'.

$R_4(a,a) = R_3(a,a) \;||\; \{R_3(a,d) \;\&\&\; R_3(d,a)\}$

$= 1$

$R_4(a,c) = R_3(a,c) \;||\; \{R_3(a,d) \;\&\&\; R_3(d,c)\}$

$= 1$

$R_4(b,a) = R_3(b,a) \;||\; \{R_3(b,d) \;\&\&\; R_3(d,a)\}$

$= 1$

$R_4(b,b) = R_3(b,b) \;||\; \{R_3(b,d) \;\&\&\; R_3(d,b)\}$

$= 1$

$R_4(b,c) = R_3(b,c) \;||\; \{R_3(b,d) \;\&\&\; R_3(d,c)\}$

$= 1$

Hence the resultant matrix is

$$R^{(4)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

FOURTH SEMESTER B.E. DEGREE EXAMINATION

JUNE / JULY 2019

Design And Analysis of Algorithms

Module - 04

7 a) Define transitive closure of a directed graph. Find the transitive closure matrix for the graph whose adjacency matrix is given (10 marks)

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

→ The transitive closure of a directed graph with n vertices can be defined as the $n \times n$ boolean matrix $T = \{t_{ij}\}$, in which the element in the ith row $(1 \le i \le n)$ and the jth column $(1 \le j \le n)$ is 1 if there exists a nontrivial directed path (i.e., a directed path of a positive length) from the ith vertex to the jth vertex; otherwise, t_{ij} is 0.

For the given adjacency matrix

Step 1: Consider the path through the vertex 'a'.

$R^{(0)} =$

	a	b	c	d	e
a	1	0	0	1	0
b	0	1	0	0	0
c	0	0	0	1	1
d	1	0	0	0	0
e	0	1	0	0	1

By comparing all the paths, the changing path is

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} \;||\; (R_{ik}^{(k-1)} \;\&\&\; R_{kj}^{(k-1)})$$

$$R_1(d,d) = R_0(d,d) \;||\; \{R_0(d,a) \;\&\&\; R_0(a,d)\}$$

$$= 0 \;||\; \{1 \;\&\&\; 1\}$$

$$= 1$$

$$R^{(1)} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 2: Consider the path through the vertex 'b'.

By comparing all the paths, we can observe that their is no changing path exists.

$$R^{(2)} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 3: Consider the path through the vertex 'c'.

By comparing all the paths, we can observe that their is no changing path exists

$$R^{(3)} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Step 4: Consider the path through the vertex 'd'.

$R_4(c,a) = R_3(c,a) \,||\, \{R_3(c,d) \,\&\&\, R_3(d,a)\}$

$= 0 \,||\, \{1 \,\&\&\, 1\}$

$= 1$

$R(4) =$

	a	b	c	d	e
a	1	0	0	1	0
b	0	1	0	0	0
c	1	0	0	1	1
d	1	0	0	1	0
e	0	1	0	0	1

Step 5: Consider the path through the vertex 'e'.

$R_5(c,b) = R_4(c,b) \,||\, \{R_4(c,e) \,\&\&\, R_4(e,b)\}$

$= 0 \,||\, \{1 \,\&\&\, 1\}$

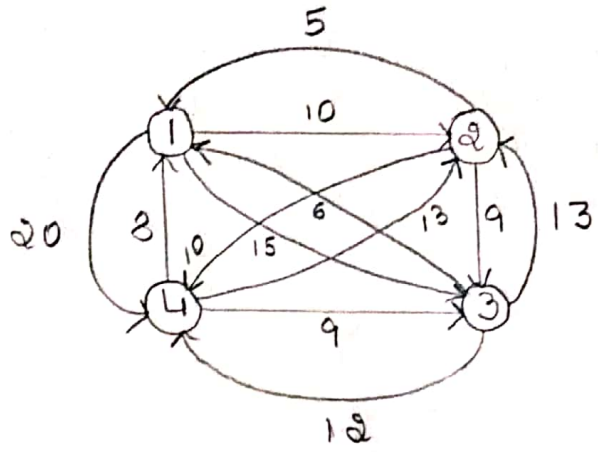
$= 1$

Hence the transitive closure matrix is

$R(5) =$

	a	b	c	d	e
a	1	0	0	1	0
b	0	1	0	0	0
c	1	1	0	1	1
d	1	0	0	1	0
e	0	1	0	0	1

b) Find the optimal tour for salesperson using dynamic program technique. The directed graph is shown in Fig. (10 marks)



→ The cost adjacency matrix is

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

The general formula is

$$g(1, V-\{1\}) = \min_{2 \le k \le n} \{ c_{1k} + g(k, V-\{1,k\}) \}$$

$$g(1,\{2,3,4\}) = \min\{ c_{12} + g(2,\{3,4\}),\ c_{13} + g(3,\{2,4\}),\ c_{14} + g(4,\{2,3\}) \} \quad \text{---- (1)}$$

We have the formula

$$g(i,s) = \min_{j \in s} \{ c_{ij} + g(j, s-\{j\}) \}$$

$$g(2,\{3,4\}) = \min\{ c_{23} + g(3,\{4\}),\ c_{24} + g(4,\{3\}) \} \quad \text{---- (2)}$$

$$g(3,\{2,4\}) = \min\{ c_{32} + g(2,\{4\}),\ c_{34} + g(4,\{2\}) \} \quad \text{---- (3)}$$

$$g(4,\{2,3\}) = \min\{c_{42} + g(2,\{3\}),\ c_{43} + g(3,\{2\})\} \quad \text{---} \quad (4)$$

Since

$$g(2,\phi) = c_{21} = 5$$

$$g(3,\phi) = c_{31} = 6$$

$$g(4,\phi) = c_{41} = 8$$

$$g(3,\{4\}) = \min\{c_{34} + g(4,\phi)\} = 20$$

$$g(4,\{3\}) = \min\{c_{43} + g(3,\phi)\} = 15$$

$$g(2,\{4\}) = \min\{c_{24} + g(4,\phi)\} = 18$$

$$g(4,\{2\}) = \min\{c_{42} + g(2,\phi)\} = 18$$

$$g(2,\{3\}) = \min\{c_{23} + g(3,\phi)\} = 15$$

$$g(3,\{2\}) = \min\{c_{32} + g(2,\phi)\} = 18$$

(2) becomes $\rightarrow \min(29, 25) = 25$

(3) becomes $\rightarrow \min(31, 30) = 30$

(4) becomes $\rightarrow \min(28, 27) = 27$

Hence eqn (1) becomes

$$g(1,\{2,3,4\}) = \min(35, 45, 47)$$

$$= 35$$

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 = 35$

is the optimal tour

8 a) Write an algorithm to construct optimal binary search tree for following data.

Key	A	B	C	D
Probability	0.1	0.2	0.4	0.3

-> ALGORITHM OptimalBST(P[1..n])

// Finds an optimal binary search tree by dynamic programming

// Input: An array P[1..n] of search probabilities
// for a sorted list of n keys.

// Output: Average number of comparisions in
// successful searches in the optimal BST and
// table R of subtrees' roots in the optimal BST

```
for i ← 1 to n do
    C[i, i-1] ← 0
    C[i, i] ← P[i]
    R[i, i] ← i
C[n+1, n] ← 0
for d ← 1 to n-1 do        // diagonal count
  for i ← 1 to n-d do
      j ← i + d
      minval ← ∞
      for k ← i to j do
          if C[i, k-1] + C[k+1, j] < minval
              minval ← C[i, k-1] + C[k+1, j]; kmin ← k
      R[i, j] ← kmin
      sum ← P[i]; for s ← i+1 to j do sum ← sum + P[s]
      C[i, j] ← minval + sum
return C[1, n], R
```

Main Table

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

Root Table

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

We have

$$C(i,j) = \min_{i \le k \le j} \{ C(i,k-1) + C(k+1,j) \} + \sum_{s=1}^{j} P_s \quad \text{for } 1 \le i \le j \le n$$

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

$$C(1,2) = \min \begin{cases} k=1;\ C(1,0) + C(2,2) + \sum_{s=1}^{2} P_s = 0 + 0.2 + 0.3 = 0.5 \\ k=2;\ C(1,1) + C(3,2) + \sum_{s=1}^{2} P_s = 0.1 + 0 + 0.3 = 0.4 \end{cases}$$

$$= 0.4$$

When $i=2$; $j=3$; $k=2,3$

$$C(2,3) = \min \begin{cases} k=2;\ C(2,1) + C(3,3) + \sum_{s=2}^{3} P_s = 0 + 0.4 + 0.6 = 1.0 \\ k=3;\ C(2,2) + C(4,3) + \sum_{s=2}^{3} P_s = 0.2 + 0 + 0.6 = 0.8 \end{cases}$$

$$= 0.8$$

When $i=3$; $j=4$; $k=3,4$

$$C(3,4) = \min \begin{cases} k=3;\ C(3,2) + C(4,4) + \sum_{s=3}^{4} P_s = 0 + 0.3 + 0.7 = 1.0 \\ k=4;\ C(3,3) + C(5,4) + \sum_{s=3}^{4} P_s = 0.4 + 0 + 0.7 = 1.1 \end{cases}$$

$$= 1.0$$

When $i=1$; $j=3$; $k=1,2,3$

$$C(1,3) = \min \begin{cases} k=1;\ C(1,0) + C(2,3) + \sum_{s=1}^{3} P_s = 0 + 0.8 + 0.7 = 1.5 \\ k=2;\ C(1,1) + C(3,3) + \sum_{s=1}^{3} P_s = 0.1 + 0.4 + 0.7 = 1.2 \\ k=3;\ C(1,2) + C(4,3) + \sum_{s=1}^{3} P_s = 0.4 + 0 + 0.7 = 1.1 \end{cases}$$

$$= 1.1$$

When $i = 2$; $j = 4$; $k = 2, 3, 4$

$$c(2,4) = \min \begin{cases} k=2 : c(2,1) + c(3,4) + \sum_{s=2}^{4} P_s = 0 + 1.0 + 0.9 = 1.9 \\ k=3 : c(2,2) + c(4,4) + \sum_{s=2}^{4} P_s = 0.2 + 0.3 + 0.9 = 1.4 \\ k=4 : c(2,3) + c(5,4) + \sum_{s=2}^{4} P_s = 0.8 + 0 + 0.9 = 1.7 \end{cases}$$

$$= 1.4$$

When $i = 1$; $j = 4$; $k = 1, 2, 3, 4$

$$c(1,4) = \min \begin{cases} k=1 : c(1,0) + c(2,4) + \sum_{s=1}^{4} P_s = 0 + 1.0 + 1.0 = 2.0 \\ k=2 : c(1,1) + c(3,4) + \sum_{s=1}^{4} P_s = 0.1 + 1.0 + 1.0 = 2.1 \\ k=3 : c(1,2) + c(4,4) + \sum_{s=1}^{4} P_s = 0.4 + 0.3 + 1.0 = 1.7 \\ k=4 : c(1,3) + c(5,4) + \sum_{s=1}^{4} P_s = 1.1 + 0 + 1.0 = 2.0 \end{cases}$$

$$= 1.7$$

Main Table

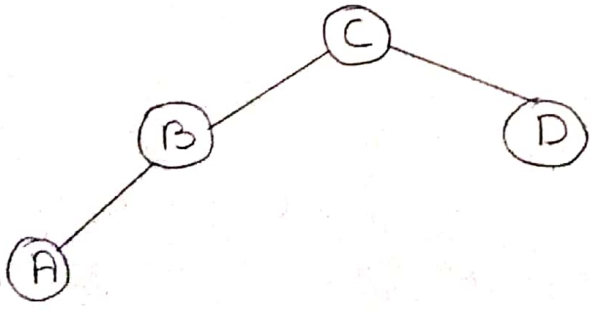
	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

Root Table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

$R(1,4) = 3(C)$; $R(1,2) = 2(B)$;

$R(1,1) = 1(A)$; $R(4,4) = 4(D)$;



b) Apply the bottom-up dynamic programming algorithm to following instance of the knapsack problem. Knapsack capacity $W = 10$.

Item	Weight	Value
1	7	42
2	3	12
3	4	40
4	5	25

→ Step 1: Construct the profit table of N+1 rows and W+1 columns, initialize $V[i, j] = 0$ when $i = 0$ (or) $j = 0$

N = No. of items = 4

W = Knapsack capacity = 10

			capacity j										
		i	0	1	2	3	4	5	6	7	8	9	10
w_i	v_i	0	0	0	0	0	0	0	0	0	0	0	0
7	42	1	0										
3	12	2	0										
4	40	3	0										
5	25	4	0										

Step 2: Using the below relation compute the entries.

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

when $i = 1$, $w = 7$, $v = 42$

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

$V[1,3] = V[0,3] = 0$; $V[1,4] = V[0,4] = 0$;

$V[1,5] = V[0,5] = 0$; $V[1,6] = V[0,6] = 0$;

$V[1,7] = \max\{V[0,7], 42 + V[0,7-7]\} = \max(0,42) = 42$

$V[1,8] = \max(0,42) = 42$; $V[1,9] = 42$

$V[1,10] = 42$;

When $i = 2$, $w = 3$, $v = 12$

$V[2,1] = V[(2-1), 1] = 0$ $\because j - w \Rightarrow 1 - 3 = -2 < 0$

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

$V[2,3] = \max\{V[1,3], 12 + V[1,(3-3)]\} = 12$

$V[2,4] = \max\{0, 12\} = 12$; $V[2,5] = \max\{0,12\} = 12$

$V[2,6] = \max\{0,12\} = 12$; $V[2,7] = \max\{42,12\} = 42$

$V[2,8] = \max\{42,12\} = 42$; $V[2,9] = \max\{42,12\} = 42$

$V[2,10] = \max\{42,54\} = 54$;

When $i = 3$, $w = 4$, $v = 40$

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

$V[3,3] = V[2,3] = 12$; $V[3,4] = \max\{12,40\} = 40$

$V[3,5] = \max\{12,40\} = 40$; $V[3,6] = \max\{12,40\} = 40$

$V[3,7] = \max\{42,52\} = 52$; $V[3,8] = \max\{42,52\} = 52$

$V[3,9] = \max\{42,52\} = 52$; $V[3,10] = \max\{54,52\} = 54$

When $i = 4$, $w = 5$, $v = 25$

$V[4,1] = [3,1] = 0$; $V[4,2] = V[3,2] = 0$

$V[4,3] = \max\{12,$ $V[4,4] = V[3,4] = 40$

$V[3,3] = 12$;

$V[4,5] = \max\{40, 25\} = 40$; $V[4,6] = \max\{40, 25\} = 40$

$V[4,7] = \max\{52, 0\} = 52$; $V[4,8] = \max\{52, 37\} = 52$

$V[4,9] = \max\{52, 65\} = 65$; $V[4,10] = \max\{54, 65\} = 65$

Hence update the above entries in the profit table.

		i \ capacity j	0	1	2	3	4	5	6	7	8	9	10
w_i	v_i	0	0	0	0	0	0	0	0	0	0	0	0
7	42	1	0	0	0	0	0	0	0	42	42	42	42
3	12	2	0	0	0	12	12	12	12	42	42	42	54
4	40	3	0	0	0	12	40	40	40	52	52	52	54
5	25	4	0	0	0	12	40	40	40	52	52	65	65

The objects that are selected are

w_i	v_i	Selected	Profit
7	42	No	
3	12	No	
4	40	Yes	40-40 = 0
5	25	Yes	65-25 = 40

weight = 5+4
= 9

Thus the maximal value is $V(4,10) = 65$

The final part of optimal solution is {item 3, item 4}

FOURTH SEMESTER B.E DEGREE EXAMINATION

JUNE / JULY 2017

Design And Analysis of Algorithms

Module - 04

7 a) Explain the concept of dynamic programming, with example. (8 marks)

→ Dynamic programming is an algorithm design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.

It involves :

- Dividing complex problem into simpler subproblems.
- Finding optimal solution of these subproblems
- Store the results of subproblems.
- Reusing the subproblems
- Finally obtaining optimal solution of complex problem.

Dynamic programming can be applied when :

A bigger problem is composed of many sub-problems and

a) Solutions of same subproblems are needed again and again.

b) Optimal solution of the given problem can be obtained by using optimal solutions of its

Subproblems.

Example:

Consider the following instance of knapsack problem: $n = 3$, $m = 20$, $(p_1, p_2, p_3) = (25, 24, 15)$ and $(w_1, w_2, w_3) = (18, 15, 10)$

By the greedy approach

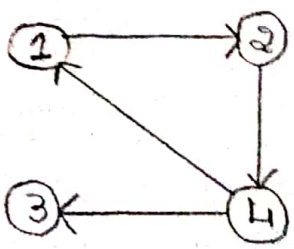
objects	1	2	3
Profit (P)	25	24	15
weights (w)	18	15	10
P/w	1·3	1·6	1·5

x (0 1 0·5) $= 1 \times 24 + 0.5 \times 15$
$= 31.5$

∴ Maximum profit = 31·5

- The solution to the knapsack problem can be viewed as the result of a sequence of decisions.
- We have to decide the values of x_i, $1 \leq i \leq n$.
- First we make a decision on x_1, then on x_2, then on x_3, and so on.
- An optimal sequence of decisions maximizes the objective function $\sum p_i x_i$. It also satisfies the constraints $\sum x_i w_i \leq m$ & $0 \leq x_i \leq 1$.

b) Trace the following graph using Warshall's algorithm. (8 marks)



-> The adjacency matrix for given graph is

$$R^{(0)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Step 1: Consider the path through the vertex '1'.

$R_{ij}^{(k)} = R_{ij}^{(k-1)} \,||\, (R_{ik}^{(k-1)} \,\&\&\, R_{kj}^{(k-1)})$

By comparing all the paths, the changing path is

$R_1(4,2) = R_0(4,2) \,||\, \{R_0(4,1) \,\&\&\, R_0(1,2)\}$

$= 1$

$$R^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Step 2: Consider the path through the vertex '2'.

$R_2(1,4) = R_1(1,4) \,||\, \{R_1(1,2) \,\&\&\, R_1(2,4)\}$

$= 1$

$R_2(4,4) = R_1(4,4) \,||\, \{R_1(4,2) \,\&\&\, R_1(2,4)\}$

$= 1$

$$R^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Step 3: Consider the path through the vertex '3'.

By comparing all the paths, we can observe that their is no changing path exists

$$R^{(3)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right] \end{array}$$

Step 4: Consider the path through the vertex '4'.

$R_4(1,1) = R_3(1,1) \;||\; \{R_3(1,4) \;\&\&\; R_3(4,1)\}$

$= 1$

$R_4(1,3) = R_3(1,3) \;||\; \{R_3(1,4) \;\&\&\; R_3(4,3)\}$

$= 1$

$R_4(2,1) = R_3(2,1) \;||\; \{R_3(2,4) \;\&\&\; R_3(4,1)\}$

$= 1$

$R_4(2,2) = R_3(2,2) \;||\; \{R_3(2,4) \;\&\&\; R_3(4,2)\}$

$= 1$

$R_4(2,3) = R_3(2,3) \;||\; \{R_3(2,4) \;\&\&\; R_3(4,3)\}$

$= 1$

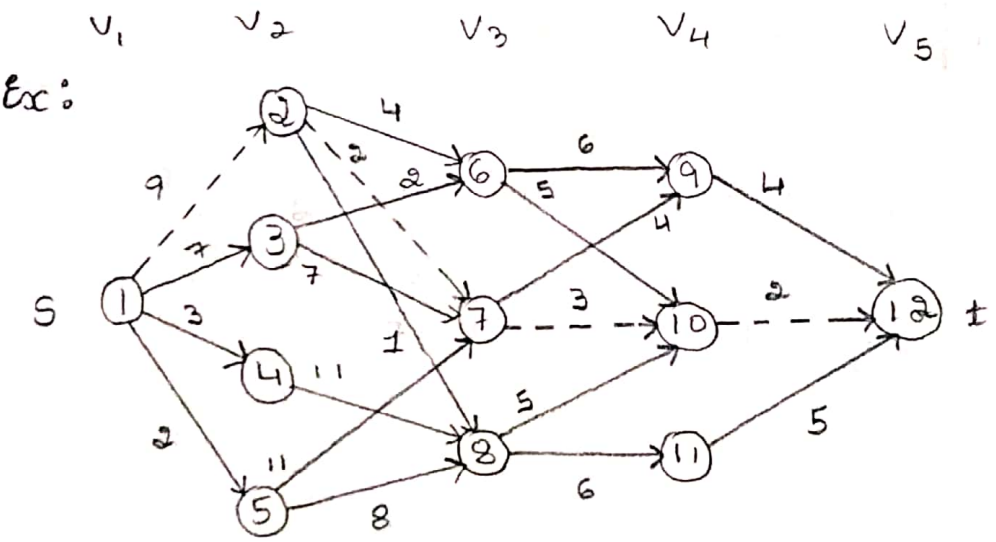
Hence the resultant matrix is

$$R^{(4)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right] \end{array}$$

8 a) Explain multistage graphs with example. Write multistage graph algorithm to forward approach. (8 marks)

→ A multistage graph $G = (V, E)$ is a directed graph in which the vertices are partitioned into $k \geq 2$ disjoint sets V_i, $1 \leq i \leq k$.

Ex:



- In the above diagram S - source, t → destination (or) sink
- $c(i, j)$ → cost of edges (i, j)

  V_i → stages in graph

$$\text{cost}(i, j) = \min_{\substack{l \in V_{i+1} \\ (j, l) \in E}} \{ c(j, l) + \text{cost}(i+1, l) \}$$

cost (ith stage, jth vertex)

Stage = 5

cost (5, 12) = 0

Stage = 4, vertex = 9 = j

$cost(4,9) = \{c(9,12) + cost(5,12)\} = 4+0 = 4$

Stage = 4, vertex = 10, 11 $\quad l = 12$

$cost(4,10) = \{c(10,12) + cost(5,12)\} = 2+0 = 2$

$cost(4,11) = \{c(11,12) + cost(5,12)\} = 5+0 = 5$

Stage = 3, vertex = 6, 7, 8 $\quad l = 9, 10, 11$

$$cost(3,6) = \min\{c(6,9) + cost(4,9),\ c(6,10) + cost(4,10)\}$$
$$= \min\{(6+4), (5+2)\} = 7$$

$$cost(3,7) = \min\{c(7,9) + cost(4,9),\ c(7,10) + cost(4,10)\}$$
$$= \min\{(4+4), (3+2)\} = 5$$

$$cost(3,8) = \min\{c(8,10) + cost(4,10),\ c(8,11) + cost(4,11)\}$$
$$= \min\{(5+2), (6+5)\} = 7$$

Stage = 2, $\quad l = 6, 7, 8$

$$cost(2,2) = \min\{c(2,6) + cost(3,6),\ c(2,7) + cost(3,7),\ c(2,8) + cost(3,8)\}$$
$$= \min\{(4+7), (2+5), (1+7)\}$$
$$= 7$$

$$\text{Cost}(2,3) = \min\{ c(3,6) + \text{cost}(3,6),\ c(3,7) + \text{cost}(3,7) \}$$

$$= \min\{(2+7),\ (7+5)\} = 9$$

$$\text{cost}(2,4) = \min\{ c(4,8) + \text{cost}(3,8) \}$$

$$= \{11+7\} = 18$$

$$\text{cost}(2,5) = \{ c(5,8) + \text{cost}(3,8) \}$$

$$= (8+7) = 15$$

Stage = 1, $i = 2, 3, 4, 5$

$$c(1,1) = \min\{ c(1,2) + \text{cost}(2,2),\ c(1,3) + \text{cost}(2,3),\ c(1,4) + \text{cost}(2,4),\ c(1,5) + \text{cost}(2,5) \}$$

$$= \min\{(9+7),\ (7+9),\ (3+18),\ (2+15)\}$$

$$= 16$$

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
Cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2	7	6	8	8	10	10	10	12	12	12	12

$p[1] = 1;\ p[5] = 12;\ p[j] = d[p[j-1]];$

p	[1]	[2]	[3]	[4]	[5]
	1	2	7	10	12

→ Path Table

Algorithm for 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 {i, j}. 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 {j, r} 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]];
}
```

b) Solve the following instance of knapsack problem using dynamic programming. Knapsack capacity is 5. (8 marks)

Item	Weight	Value
1	2	$ 12
2	1	$ 10
3	3	$ 20
4	2	$ 15

→ Step 1: Construct the profit table of N+1 rows and W+1 columns, initialize $V[i,j] = 0$ when $i = 0$ (or) $j = 0$.

N = No. of items = 4

W = Knapsack capacity = 5

		capacity j						
		i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
2	12	1	0					
1	10	2	0					
3	20	3	0					
2	15	4	0					

Step 2: Using the below relation compute the entries.

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

When $i = 1$, $w = 2$, $v = 12$

$V[1,1] = V[0,1]$ $\because j - w = 1 - 2 = -1 < 0$

$V[1,2] = \max\{V[0,2], 12 + V[0,0]\} = 12$

$V[1,3] = \max\{0, 12\} = 12$; $V[1,4] = \max\{0, 12\} = 12$

$V[1,5] = \max\{0, 12\} = 12$

When $i = 2$, $w = 1$, $v = 10$

$V[2,1] = \max\{V[1,1], 10 + V[1,0]\} = 10$

$V[2,2] = \max\{12, 0\} = 12$; $V[2,3] = \max\{12, 22\} = 22$

$V[2,4] = \max\{12, 22\} = 22$; $V[2,5] = \max\{12, 22\} = 22$

When $i = 3$, $w = 3$, $v = 20$

$V[3,1] = V[2,1] = 10$; $V[3,2] = V[2,2] = 12$

$V[3,3] = \max\{22, 20\} = 22$; $V[3,4] = \max\{22, 30\} = 30$

$V[3,5] = \max\{22, 32\} = 32$;

When $i = 4$, $w = 2$, $v = 15$

$V[4,1] = V[3,1] = 10$; $V[4,2] = \max\{12, 15\} = 15$

$V[4,3] = \max\{22, 25\} = 25$; $V[4,4] = \max\{30, 27\} = 30$

$V[4,5] = \max\{32, 37\} = 37$;

Hence update the above entries in the profit table

capacity j

		i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
2	12	1	0	0	12	12	12	12
1	10	2	0	10	12	22	22	22
3	20	3	0	10	12	22	30	32
2	15	4	0	10	15	25	30	37

The objects selected are :

w_i	v_i
2	12
1	10
3	20
2	15

Selected	Profit
Yes	12-12 = 0
Yes	22-10 = 12
No	
Yes	37-15 = 22

Weight
= 2 + 1 + 2
= 5

Thus the maximal value is $V(4,5) = \$37$

The final part of optimal solution is

{ item 1, item 2, item 4 }

Fourth Semester B.E Degree Examination, Dec 2018 / Jan. 2019

Design and Analysis of Algorithm

Module-04

7a) Define transitive closure of a graph. Write Warshall's algorithm to compute transitive closure of a directed graph. Apply the same on the graph defined by the following adjacency matrix

$$R = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

[8 Marks]

→ **Transitive closure :-**

The transitive closure of a directed graph with n vertex can be defined as the nxn boolean matrix $T = \{t_{ij}\}$ in which the elements in the i^{th} row and j^{th} column is 1 if there exist a nontrivial path from the i^{th} vertex to the j^{th} vertex otherwise t_{ij} is 0.

Warshall's Algorithm:

// implement Warshall's Algorithm for computing the transitive closure
// input: The adjacency matrix A of digraph with n vertices
// output: The transitive closure of the digraph

```
R(0) ← A
for K ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            R^(K)[i,j] ← R^(K-1)[j,j] or (R^(K-1)[i,K] and R^(K-1)[K,j])
return R(n)
```

The given adjacency matrix

$R_0 =$

	a	b	c	d
a	0	1	0	0
b	0	0	1	0
c	0	0	0	1
d	0	0	0	0

Step 1:- Consider the smallest path distance through the vertex 'a'

Consider ath row and ath column from the above adjacency matrix

$$R_{ij}(K) = \min\{R_{ij}^{(K-1)} \,||\, (R_{ik}^{(K-1)} \,\&\&\, R_{kj}^{(K-1)})\}$$

$R^{(1)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

Step 2:- Consider the shortest path distance through the vertex 'b'

$R_2(a,c) = \min\{R_1(a,c) \,||\, (R_1(a,b) \,\&\&\, R_1(b,c))\}$
$= \min\{0 \,||\, (1 \,\&\&\, 1)\}$
$= 1$

$R^{(2)}$ =

	a	b	c	d
a	0	1	1	0
b	0	0	1	0
c	0	0	0	1
d	0	0	0	0

Step 3:- Consider the shortest path distance through the vertex 'c'

$R_3(a,d) = \min\{R_2(a,d) \,||\, R_2(a,c) \,\&\&\, R_2(c,d)\}$
$= \min\{0 \,||\, (1 \,\&\&\, 1)\}$
$= 1$

$R_3(b,d) = \min\{R_2(b,d) \,||\, R_2(b,c) \,\&\&\, R_2(c,d)\}$
$= \min\{0 \,||\, (1 \,\&\&\, 1)\}$
$= 1$

$$R^{(3)} = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Step 4:- Consider the shortest path distance through the vertex 'd'

As there is no change in elements

∴ The transitive closure of the given adjacency matrix is

$$T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

b) Using dynamic programming, solve the below instance of Knapsack problem. [8 Marks]

item	Weight	Value
1	2	12
2	1	10
3	3	20
4	2	15

Capacity $W = 5$

→ Given :- Knapsack capacity (W) = 5 kg
Number of items (n) = 4

Step 1:- Draw a table 'T' Write $(n+1) = 4+1 = 5$
∴ 5 → number of rows
$(W+1) = 5+1 = 6$ → number of colums
Initilize 0th row and 0th column as 0

Step 2:- Start filling the table row wise top to bottom from left to right using the formula

$$F(i,j) = \begin{cases} \max\{F(i-1,j), v_i + F(i-1, j-w_i)\} & \text{if } j - w_i \geq 0 \\ F(i-1,j) & \text{if } j - w_i < 0 \end{cases}$$

$w_1 = 2, v_1 = 12$

T → Weights: 0 1 2 3 4 5

Items ↓: 0 1 2 3 4

0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0					
3	0					
4	0					

Here the weight of the first item is 2 and the value is 12. Hence fill the value of the first item in the table i.e (12)

In the table, the weight increases. If the weight increasing but there is no profit. So retain the same value i.e (12)

$W_1 = 2, V_1 = 12$
$W_2 = 1, V_2 = 10$

T	0	1	2	3	4	5 → Weight
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	10	12	22	22	22
3	0					
4	0					

↓ Item

As we have considered the first and second item, the weight of the second item is 1 and its value is 10. So write the value of the 2nd item in the table i.e $T(2,1) = 10$

The weight of the 1st item is 2 and its value is 12. So write the value of the 1st item in the table i.e $T(2,2) = 12$

Hence we have not considered the weight 3 from the given. We are going to add the considered items i.e W_1 and W_2 and fill the table. i.e $T(2,3) = T(2,1) + T(2,2)$

$$T(2,3) = 10 + 12$$

$$T(2,3) = 22$$

As the weight increases there is no increase in the profit. Hence retain the value. i.e (22)

$w_1 = 2, \quad v_1 = 12$
$w_2 = 1, \quad v_2 = 10$
$w_3 = 3, \quad v_3 = 20$

Considering the previous steps

$\therefore T(3,1) = 10$

$T(3,2) = 12$

$T(3,3) = 22$ $T(3,4) = 30$ $T(3,5) = 32$

$w_1 = 2, \quad v_1 = 12$
$w_2 = 1, \quad v_2 = 10$
$w_3 = 3, \quad v_3 = 20$
$w_4 = 2, \quad v_4 = 15$

Considering the previous steps

$T(4,1) = 10$

$T(4,2) = 15$

$T(4,3) = 25$

$T(4,4) = 30$

$T(4,5) = 37$

Tracing :-

$$F(i,j) = \max F(i-1,j),\ v_i + F(i-1, j-w_i)$$

$F(4,5) = \max$ 32 , $15 + F(4-1, 5-2)$

$F(4,5) = \max$ 32 , $15 + 22$ 37

Weight	Value
$w_1 = 2$	$v_1 = 12$
$w_2 = 1$	$v_2 = 10$
$w_3 = 3$	$v_3 = 20$
$w_4 = 2$	$v_4 = 15$

T	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	12	12	12	12
2	0	10	12	22	22	22
3	0	10	12	22	30	32
4	0	10	15	25	30	37

Selected	Profit
Yes	12 - 12 = 0
Yes	22 - 10 = 12
No	
Yes	37 - 15 = 22

Thus, the maximum value is $F(4,5) = \$37$

The final part of the optimal solution is

{item 1, item 2, item 4}

$\therefore$

8a) Obtain a optimal binary search tree for the following four-key set. [8 Marks]

Key	A	B	C	D
Probability	0.1	0.2	0.4	0.3

→ At initial

main Table

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.3	
4				0	0.4
5					0

Root table

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

We compute each data with the formula

$$c(i,j) = \min_{i \le k \le j} \{c(i,k-1) + c(k+1,j)\} + \sum_{s=i}^{j} P_s \quad \text{for } 1 \le i \le j \le n$$

When $i=1;\ j=2;\ k=1,2$

$$c(1,2) = \min \begin{cases} k=1;\ c(1,0)+c(2,2)+\sum_{s=1}^{2} P_s = 0+0.2+0.3 = 0.5 \\ k=2;\ c(1,1)+c(3,2)+\sum_{s=1}^{2} P_s = 0.1+0.3+0 = 0.4 \end{cases}$$

$$= 0.4$$

When $i=2;\ j=3;\ k=2,3$

$$c(2,3) = \min \begin{cases} k=2;\ c(2,1)+c(3,3)+\sum_{s=2}^{3} P_s = 0+0.4+0.6 = 1.0 \\ k=3;\ c(2,2)+c(4,3)+\sum_{s=2}^{3} P_s = 0.2+0+0.6 = 0.8 \end{cases}$$

$$= 0.8$$

When $i = 3$; $j = 4$; $K = 3, 4$

$$c(3,4) = \min \begin{cases} k=3 ; c(3,2) + c(4,4) + \sum_{s=3}^{4} P_s = 0 + 0.3 + 0.7 = 1.0 \\ k=4 ; c(3,3) + c(5,4) + \sum_{s=3}^{4} P_s = 0.4 + 0 + 0.7 = 1.1 \end{cases}$$

$= 1.0$

When $i = 1$; $j = 3$; $K = 1, 2, 3$

$$c(1,3) = \min \begin{cases} K=1 ; c(1,0) + c(2,3) + \sum_{s=1}^{3} P_s = 0 + 0.8 + 0.7 = 1.5 \\ K=2 ; c(1,1) + c(3,3) + \sum_{s=1}^{3} P_s = 0.1 + 0.4 + 0.7 = 1.2 \\ K=3 ; c(1,2) + c(3,4) + \sum_{s=1}^{3} P_s = 0.4 + 0 + 0.7 = 1.1 \end{cases}$$

$= 1.1$

When $i = 2$; $j = 4$; $K = 2, 3, 4$

$$c(2,4) = \min \begin{cases} K=2 ; c(2,1) + c(3,4) + \sum_{s=2}^{4} P_s = 0 + 1.0 + 0.9 = 1.9 \\ K=3 ; c(2,2) + c(4,4) + \sum_{s=2}^{4} P_s = 0.2 + 0.3 + 0.9 = 1.4 \\ K=4 ; c(2,3) + c(5,4) + \sum_{s=2}^{4} P_s = 0.8 + 0 + 0.9 = 1.7 \end{cases}$$

$= 1.4$

When $i = 1$; $j = 4$; $K = 1, 2, 3, 4$

$$c(1,4) = \min \begin{cases} K=1 ; c(1,0) + c(2,4) + \sum_{s=1}^{4} P_s = 0 + 1.0 + 1.0 = 2.0 \\ K=2 ; c(1,1) + c(3,4) + \sum_{s=1}^{4} P_s = 0.1 + 1.0 + 1.0 = 2.1 \\ K_3=3 ; c(1,2) + c(4,4) + \sum_{s=1}^{4} P_s = 0.4 + 0.3 + 1.0 = 1.7 \\ K_4=4 ; c(1,3) + c(5,4) + \sum_{s=1}^{4} P_s = 1.1 + 0 + 1.0 = 2.0 \end{cases}$$

$= 1.7$

main table

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.6
4				0	0.3
5					0

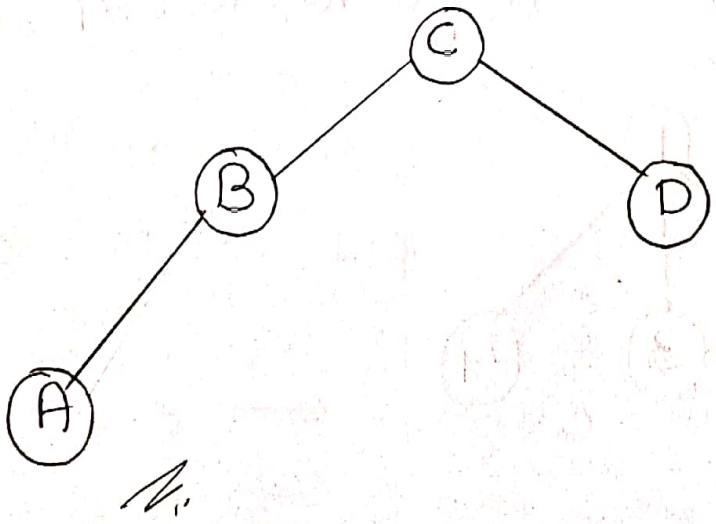
Root table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

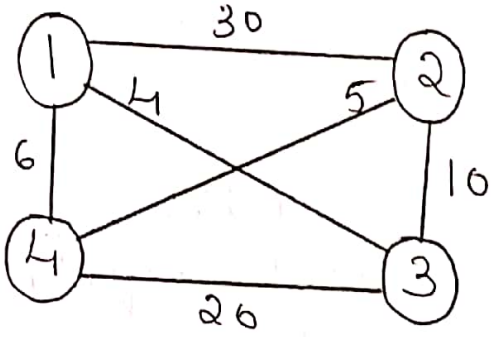
Thus the average number of key comparision in the optimal tree is equal to 1.7

With the help of root table we can construct the binary search tree such that

* Root is key no. 3 [R(1,4) = 3] i.e C where as key 1 is the part of left subtree and key = 4 is the part of right subtree.
* Construct subtree between 2 and 1, make 2 as the root and 1 as the left node
* R(1,2) = 2 (B) R(1,1) = 1 (A) R(4,4) = 4 (D)



b) Solve the following travelling sales person problem represented as a graph shown in fig using dynamic programming. [8 Marks]

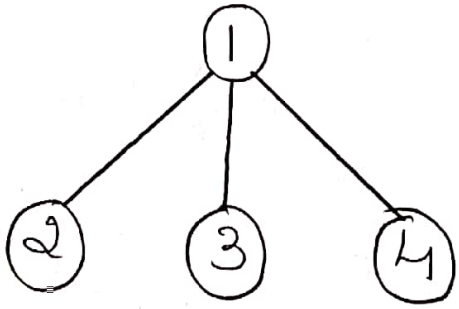


→ The adjacency matrix for the given graph is

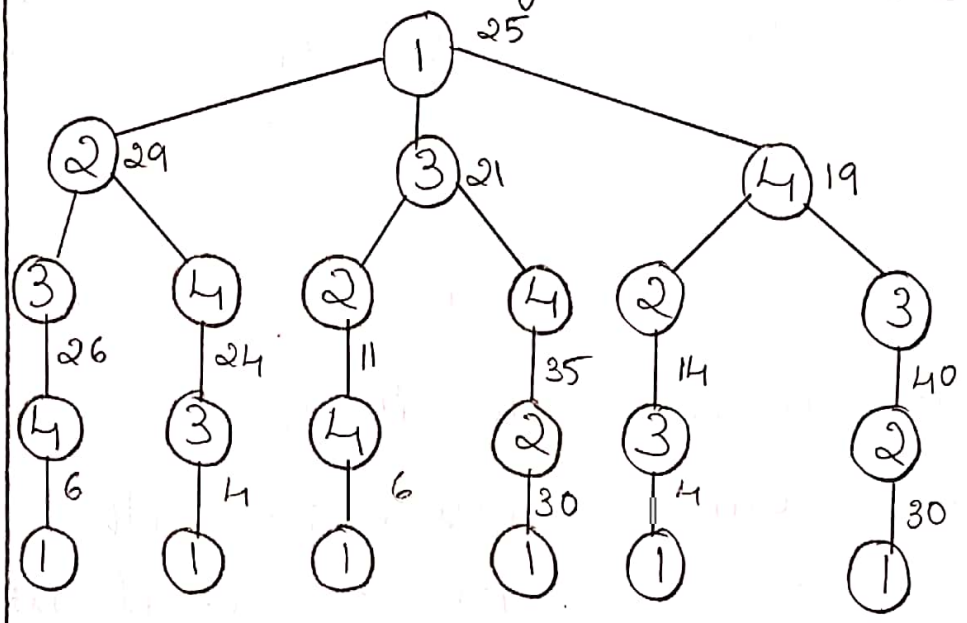
$$\begin{bmatrix} 0 & 30 & 4 & 6 \\ 30 & 0 & 10 & 5 \\ 4 & 10 & 0 & 20 \\ 6 & 5 & 20 & 0 \end{bmatrix}$$

For the given graph, we have chosen "1" as source vertex

Here from "1" we can go to 2, 3, 4. It can be shown as



from 2 we can go to 3 and 4
from 3 we can go to 2 and 4
from 4 we can go to 2 and 3



	1	2	3	4
1	0	30	4	6
2	30	0	10	5
3	4	10	0	20
4	6	5	20	0

From node 2:

2 → 3 → 4 → 1 = 296
2 → 4 → 3 → 1 = 29
29 is minimum

From node 3:

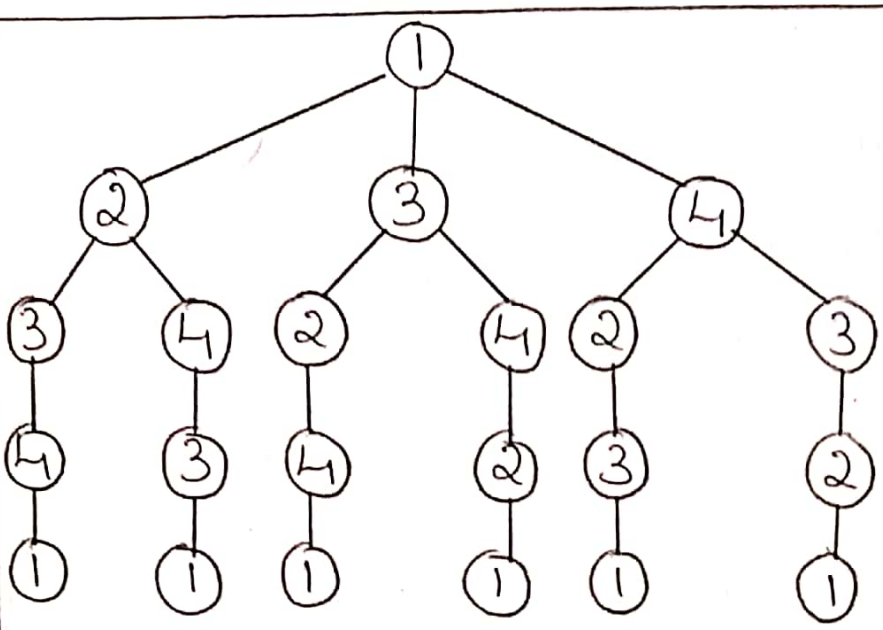
3 → 2 → 4 → 1 = 321
3 → 4 → 2 → 1 = 55
21 is minimum

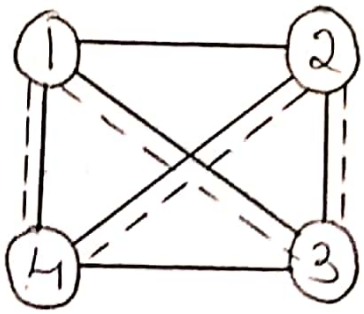
From node 4:

4 → 2 → 3 → 1 = 19
4 → 3 → 2 → 1 = 60
19 is minimum

From node 1:

1 → 2 → 4 → 3 → 1 = 59
1 → 3 → 4 → 2 → 1 = 25
1 → 4 → 2 → 3 → 1 = 25
25 is minimum





The minimum value is 25

If the salesman travels the path $1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$ he can visit all the nodes and return to the initial nodes with minimum cost.

Fourth Semester B.E Degree Examination, Dec 2017/Jan. 2018

Design and Analysis of Algorithm

Module-04

7a) Define transitive closure. Write Warshall's algorithm to compute transitive closure. Find its efficiency. [8 Marks]

→ **Transitive closure :-**

The transitive closure of a directed graph with n vertices can be defined as the nxn boolean matrix $T = \{t_{ij}\}$ in which the elements in the i^{th} row and j^{th} column is 1 if there exist a nontrivial path from the i^{th} vertex to the j^{th} vertex otherwise t_{ij} is 0.

Warshall's Algorithm :-

// implement Warshall's Algorithm for transitive closure
// input : The adjacency matrix A of digraph with n vertices
// output : Transitive closure of a digraph

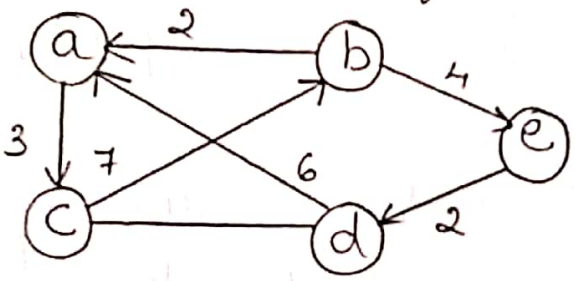
```
R(0) ← A
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            R(k)[i,j] ← R(k-1)[i,j] or (R(k-1)[i,k] and R(k-1)[k,j])
return R(n)
```

The time efficiency of Warshall's algorithm is $\Theta(n)$

Space efficiency :- Although seperate matrices for recording intermediate results of the algorithm are used, that can be avoided.

b) Apply Floyd's algorithm to find all pair shortest path for the graph.

[8 Marks]



→ The cost adjacency matrix for the given graph is as follows:

$R(0) =$

	a	b	c	d	e
a	0	∞	3	∞	∞
b	2	0	∞	∞	4
c	∞	7	0	1	∞
d	6	∞	∞	0	∞
e	∞	∞	∞	2	0

Step 1 :- Consider the smallest path distance through the vertex 'a'

Consider a^{th} row and a^{th} column from the above adjacency matrix

$$R_{ij}(K) = \min\{ R_{ij}^{(K-1)}, (R_{iK}^{(K-1)} + R_{Kj}^{(K-1)}) \}$$

$R_1(b,c) = \min\{R_0(b,c), R_0(b,a) + R_0(a,c)\}$
$= \min\{\infty, 2+3\}$
$= 5$

$R_1(d,c) = \min\{R_0(d,c), R_0(d,a) + R_0(a,c)\}$
$= \min\{\infty, 6+3\}$
$= 9$

$R^{(1)} =$

	a	b	c	d	e
a	0	∞	3	∞	∞
b	2	0	5	∞	~~∞~~ 4
c	∞	7	0	1	∞
d	6	∞	9	0	∞
e	∞	∞	∞	2	0

Step 2 :- Consider the shortest path distance through the vertex 'b'

$R_2(c,a) = \min\{R_1(c,a), R_1(c,b) + R_1(b,a)\}$
$= \min\{\infty, 7+2\}$
$= 9$

$R_2(c,e) = \min\{R_1(c,e), R_1(c,b) + R_1(b,e)\}$
$= \min\{\infty, 7+4\} = 11$
$= 11$

$R^{(2)} =$

	a	b	c	d	e
a	0	∞	3	∞	∞
b	2	0	5	∞	4
c	9	7	0	1	11
d	6	∞	9	0	∞
e	∞	∞	∞	2	0

Step 3:- Consider the shortest path distance through the vertex 'c'

$R_3(a,b) = \min\{R_2(a,b), R_2(a,c)+R_2(c,b)\}$
$= \min\{\infty, 3+7\}$
$= 10$

$R_3(a,d) = \min\{R_2(a,d), R_2(a,c)+R_2(c,d)\}$
$= \min\{\infty, 3+1\}$
$= 4$

$R_3(a,e) = \min\{R_2(a,e), R_2(a,c)+R_2(c,e)\}$
$= \min\{\infty, 3+11\}$
$= 14$

$R_3(b,d) = \min\{R_2(b,d), R_2(b,c)+R_2(c,d)\}$
$= \min\{\infty, 5+1\}$
$= 6$

$R_3(d,b) = \min\{R_2(d,b), R_2(d,c)+R_2(c,b)\}$
$= \min\{\infty, 9+7\}$
$= 16$

$R_3(d,e) = \min\{R_2(d,e), R_2(d,c)+R_2(c,e)\}$
$= \min\{\infty, 9+11\}$
$= 20$

$R^{(3)} =$

	a	b	c	d	e
a	0	10	3	4	14
b	2	0	5	6	4
c	9	7	0	1	11
d	6	16	9	0	20
e	∞	∞	∞	2	0

Step 4:- Consider the shortest path distance through the vertex 'd'

$R_4(c,a) = \min\{R_3(c,a), R_3(c,d) + R_3(d,a)\}$
$= \min\{9, 1+6\}$
$= 7$

$R_4(e,b) = \min\{R_3(e,b), R_3(e,d) + R_3(d,b)\}$
$= \min\{\infty, 2+16\}$
$= 18$

$R_4(e,c) = \min\{R_3(e,c), R_4(e,d) + R_3(d,c)\}$
$= \min\{\infty, 2+9\}$
$= 11$

$R^{(4)} =$

	a	b	c	d	e
a	0	10	3	4	14
b	2	0	5	6	4
c	7	7	0	1	11
d	6	16	9	0	20
e	8	18	11	2	0

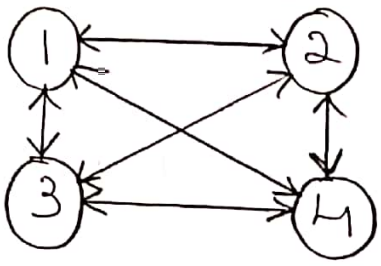
Step 5:- Consider the shortest distance through the vertex 'e'

~~$R_5(\ ,\ ) = \min\{$~~ As there is no change in elements.

∴ By Floyd's algorithm all pair shortest path for the given graph is

$$\begin{bmatrix} 0 & 10 & 3 & 4 & 14 \\ 2 & 0 & 5 & 6 & 4 \\ 7 & 7 & 0 & 1 & 11 \\ 6 & 16 & 9 & 0 & 20 \\ 8 & 18 & 11 & 2 & 0 \end{bmatrix}$$ //.

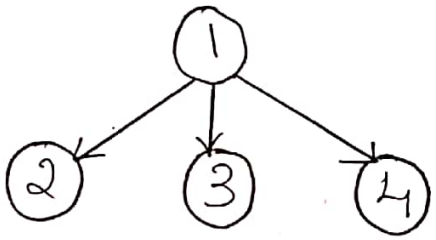
8a) For the given cost matrix, obtain optimal cost tour using dynamic programming. [8 Marks]



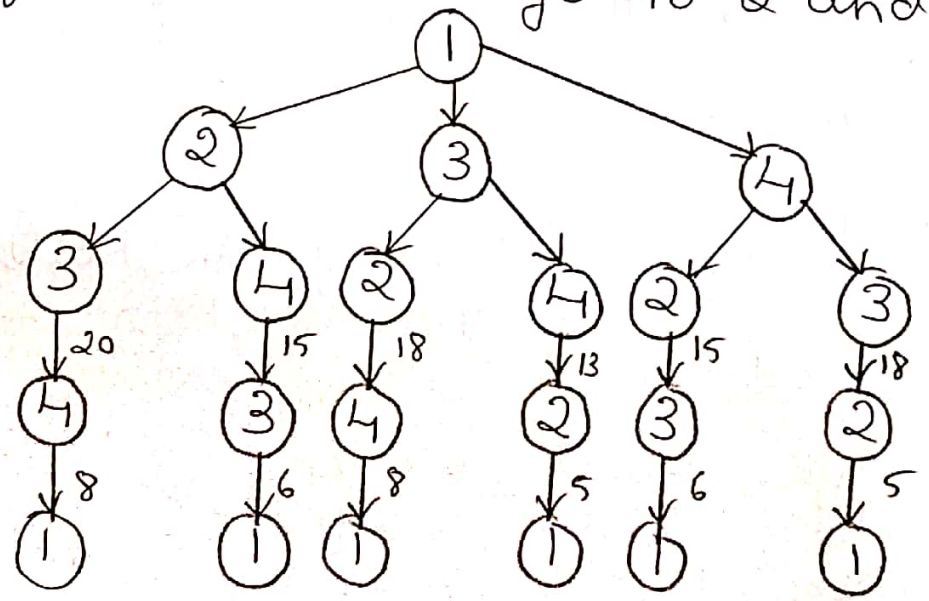
$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

→ For the given graph, we have choosen "1" as source vertex

Here from "1" we can go to 2, 3, 4. It can be shown as



from 2 we can go to 3 and 4
from 3 we can go to 2 and 4
from 4 we can go to 2 and 3



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

From node 2:

2 → 3 → 4 → 1 = 29

2 → 4 → 3 → 1 = 25

25 is minimum

From node 3:

3 → 2 → 4 → 1 = 31

3 → 4 → 2 → 1 = 25

25 is minimum

From node 4:

4 → 2 → 3 → 1 = 23

4 → 3 → 2 → 1 = 27

23 is minimum

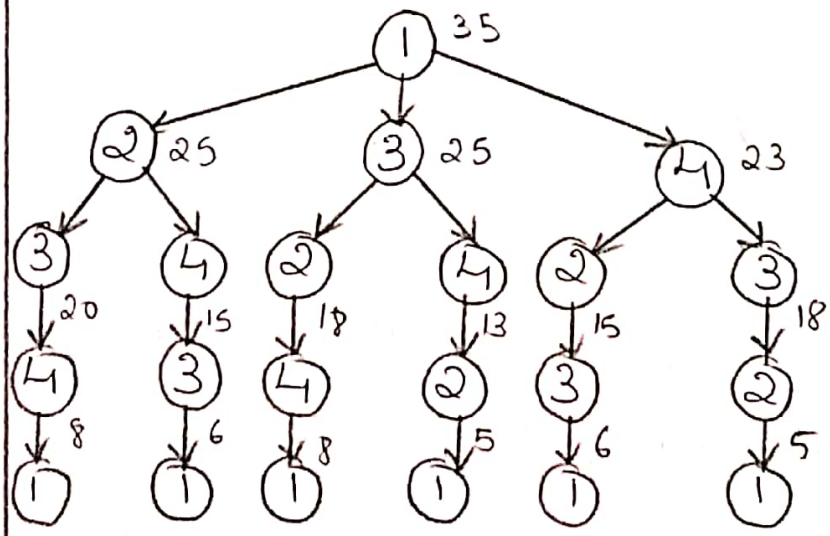
From node 1:

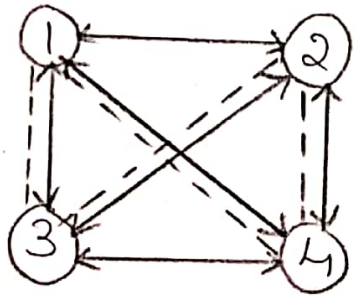
1 → 2 → 4 → 3 → 1 = 35

1 → 3 → 4 → 2 → 1 = 40

1 → 4 → 2 → 3 → 1 = 43

35 is minimum





The minimum value is 35

If the salesman travels the path 1 → 4 → 2 → 3 → 1 he can visit all the nodes and return to the initial nodes with minimum cost.

b) Write a pseudocode to find an optimal binary search tree by dynamic programming. [8 Marks]

→

Algorithm :- OptimalBST(P[1...n])

// Find an optimal binary search tree by dynamic programming

// input : An array P[1...n] of search probabilities for a sorted list of n keys

// Output : Average number of comparisons in succesfull searches in the

// optimal BST and table R of subtrees roots in the optimal BST for i ← 1 n do

```
C[i, i-1] ← 0
C[n+1, n] ← 0
C[i, i] ← P[i]
R[i, i] ← i
for d ← 1 to n-1 do // diagonal count
  for i ← 1 to n-d do
      j ← i + d minval ← ∞
for k ← i to j do
   if c[i, k-1] + c[k+1, j] < minval
      minval ← c[i, k-1] + c[k+1, j]; kmin ← k R[i, j] ← kmin
sum ← P[i]; for s ← i+1 to j do sum ← sum + P[s]
                                c[i, j] ← minval + sum
return c[1, n], R
```

Fourth Semester B.E Degree Examination, June/July 2018

Design and Analysis of Algorithm

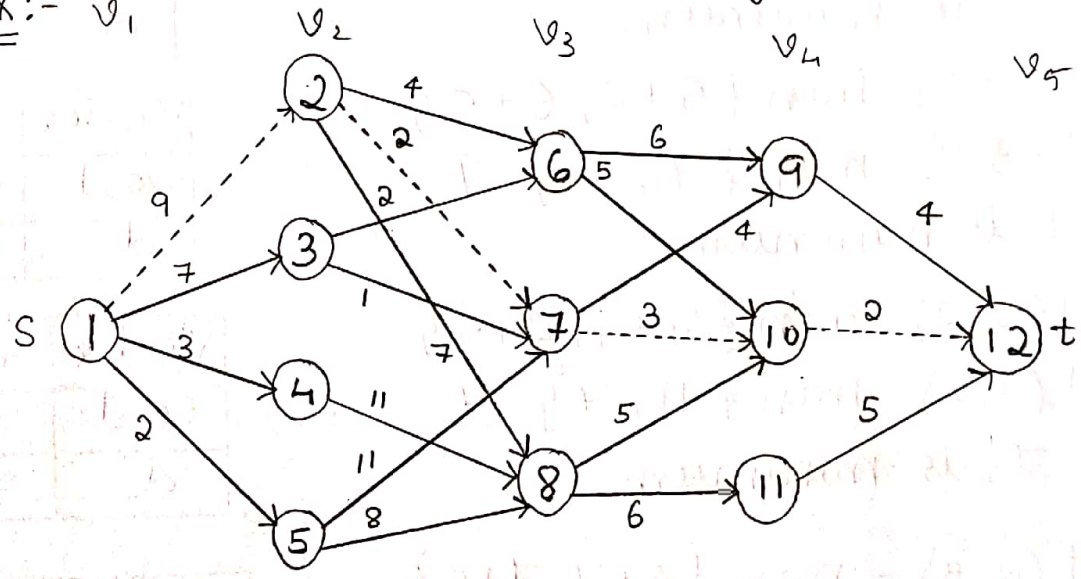
Module-04

7 a) Explain multistage graph with an example. Write multistage graph algorithm using backward approach. [8 Marks]

→ **Multistage graph :-**

A multistage graph $G=(V,E)$ is a direct graph in which the vertices are partitioned into $K>2$ disjoint sets V_i.

Ex:-



S → source, t → sink/destination

* Minimum cost from s to t = Sum of the cost of the edges on the path

cost (i^{th} stage, j^{th} vertex)

cost (5, 12) = 0

cost (4, 9) = 4

cost (4, 10) = 2

cost (4, 11) = 5

p	[1]	[2]	[3]	[4]	[5]
	1	2	7	10	12

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2	7	6	8	8	10	10	10	12	12	12	12

cost (i^{th} stage, j^{th} vertex)

cost (i, j) = min {c(j, r) + cost (i+1, r)}

cost (3,6) = min { 6+4, 5+2 }

cost (3,6) = min { 7 }

∴ 7 is minimum

Vertex	6
cost	7
d	10

cost (3,7) = min { 4+4, 3+2 }

cost (3,7) = min { 8, 5 } = 5

∴ 5 is minimum

Vertex	7
cost	5
d	10

cost (3,8) = min { 5+2, 6+5 }

cost (3,8) = min { 7, 11 } = 7

∴ 7 is minimum

Vertex	8
cost	7
d	10

cost (2,2) = min { 4+7, 2+5 }

cost (2,2) = min { 11, 7 } = 7

∴ 7 is minimum

Vertex	2
cost	7
d	7

cost (2,3) = min { 2+7, 7+5 }

cost (2,3) = min { 9, 12 } = 9

∴ 9 is minimum

Vertex	3
cost	9
d	6

$Cost(2,4) = min\{11+7\} = 18$

$\therefore$ 18 is minimum

Vertex	4
Cost	18
d	8

$cost(2,5) = min\{11+5, 8+7\}$

$cost(2,5) = min\{16, 15\} = 15$

$\therefore$ 15 is minimum

Vertex	5
cost	15
d	8

$cost(1,1) = min\{9+7, 7+9, 3+18, 2+15\}$

$cost(1,1) = min\{16, 16, 21, 17\} = 16$

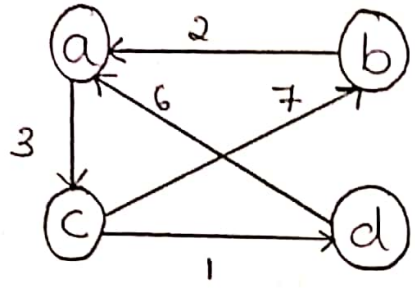
$\therefore$ 16 is minimum

Vertex	1
Cost	16
d	2

Algorithm using backward approach:

```
Algorithm BGraph (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 (i,j), p[1:k] is a minimum
   cost path
{
   bcost[1] := 0.0;
   for j := 2 to n do
   { // compute bcost[j]
     Let r be such that (r,j) is an edge of G
     and bcost[r] + c[r,j] is minimum;
     bcost[j] := bcost[r] + c[r,j];
     d[j] := r;
   }
   // find a minimum-cost path
   p[1] := 1; p[k] := n;
} for j := k-1 to 2 do p[j] := d[p[j+1]];
```

b) Floyd's algorithm to solve all pair shortest path problem for the graph given below in the figure. [8 Marks]



→ The cost adjacency matrix for the given graph is as follows

$D(0) =$

	a	b	c	d
a	0	∞	3	∞
b	2	0	∞	∞
c	∞	7	0	1
d	6	∞	∞	0

Step 1:- Consider the smallest path distance through the vertex 'a'

Consider ath row and ath column from the above adjacency matrix

$$D_{ij}(K) = \min\{D_{ij}^{(K-1)}, D_{iK}^{(K-1)} + D_{Kj}^{(K-1)}\}$$

$D_1(b,c) = \min\{D_0(b,c), D_0(b,a) + D_0(a,c)\}$

$= \min\{\infty, 2+3\} = 5$

$D_1(d,c) = \min\{D_0(d,c), D_0(d,a) + D_0(a,c)\}$

$= \min\{\infty, 6+3\}$

$= 9$

$D^{(1)} =$

	a	b	c	d
a	0	∞	3	∞
b	2	0	5	∞
c	∞	7	0	1
d	6	∞	9	0

Step 2:- Consider the shortest path distance through the vertex 'b'

$D_2(c,a) = \min\{D_1(c,a), D_1(c,b) + D_1(b,a)\}$

$= \min\{\infty, 7+2\}$

$= 9$

$D^{(2)} =$

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	9	7	0	1
d	6	16	9	0

Step 3:- Consider the shortest path distance through the vertex 'c'

$D_3(c,a) = \min\{D_2(c,a), D_2(c,d) + D_2(d,a)\}$

$= \min\{9, 1+6\}$

$= 7$

$D^{(3)} =$

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	7	7	0	1
d	6	16	9	0

8a) Explain Bellman Ford al to find shortest path from single source to all destination for a directed graph with negative edge cost [8 Marks]

→ **Bellman-Ford Algorithm:**

The Bellman-Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph.

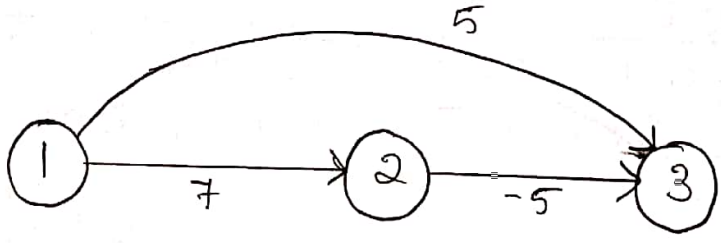
Bellman-Ford algorithm is capable of handling graphs in which some of the edges weights are negative numbers.

Bellman-Ford is simpler than Dijstra and suits well for distributed system. But time complexity of Bellman Ford is $O(V E)$ which is more than Dijstra.

Algorithm :

```
Algorithm BellmanFord (v, cost, dist, n)
// single-source/all-destinations shortest
// paths with negative edge costs
{
 for i := 1 to n do // Initialize dist
     dist[i] := cost[v, i];
 for k := 2 to n-1 do
    for each u such that u ≠ v and u has
                at least one incoming edge do
      for each (i, u) in the graph do
        if dist[u] > dist[i] + cost[i, u] then
            dist[u] := dist[i] + cost[i, u];
}
```

Consider an example :



Here 1 is the source node

∴ dist[2] = 7 and dist[3] = 5

$$dist^{K}[u] = \min(dist^{K-1}[u], dist^{K-1}[w] + cost[w,u])$$
for each neighbour w to u

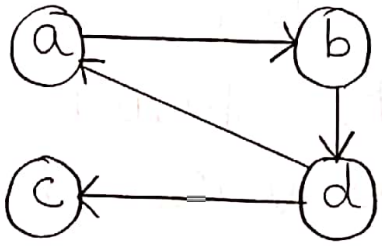
Path length K

	1	2	3 → Vertices
0	0	∞	∞
1	0	7	5
2	0	7	2
3	0	7	2

∴

Source node = 1	
Vertex	Shortest distance
2	7
3	2

b) Apply Warshall's algorithm to the digraph given below in fig. and find the transitive closure. [8 Marks]



→ The ~~cost~~ adjacency matrix for the given graph is as follows:

$R(0) =$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

Step 1:- Consider the smallest path distance through the vertex 'a'

Consider a^{th} row & a^{th} column from the above adjacency matrix

$$R_{ij}(K) = \min \{ R_{ij}^{(K-1)} \,||\, (R_{ik}^{(K-1)} \,\&\&\, R_{kj}^{(K-1)}) \}$$

$R_1(d,b) = \min \{ R_0(d,b) \,||\, (R_0(d,a) \,\&\&\, R_0(a,b)) \}$

$= \min \{ 0 \,||\, (1 \,\&\&\, 1) \}$

$= 1$

$R^{(1)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

Step 2 :- Consider the shortest path distance through the vertex 'b'

$R_2(a,d) = \min\{R_1(a,d) \,||\, (R_1(a,b) \xi\xi R_1(b,d)\}$
$= \min\{0 \,||\, (1 \xi\xi 1)\}$
$= 1$

$R_2(d,d) = \min\{R_1(d,d) \,||\, (R_1(d,b) \xi\xi R_1(b,d)\}$
$= \min\{0 \,||\, (1 \xi\xi 1)\}$
$= 1$

$R^{(2)}$ =

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

Step 3 :- Consider the shortest path distance through the vertex 'c'

R_3 ($\gamma = \min\{R_2($

As there is no change it remains the same

$R^{(3)}$ =

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

Step 4:- Consider the shortest path distance through the vertex 'd'

$R_4(a,a) = \min\{(R_3(a,a) \,||\, (R_3(a,d) \,\&\&\, R_3(d,a)\}$
$= \min\{ 0 \,||\, (1 \,\&\&\, 1) = 1$
$= 1$

$R_4(a,c) = \min\{R_3(a,c) \,||\, (R_3(a,d) \,\&\&\, R_3(d,c)\}$
$= \min\{ 0 \,||\, (1 \,\&\&\, 1) = 1$

$R_4(b,a) = \min\{R_3(b,a) \,||\, (R_3(b,d) \,\&\&\, R_3(d,a)\}$
$= \min\{ 0 \,||\, (1 \,\&\&\, 1)$
$= 1$

$R_4(b,b) = \min\{R_3(b,b) \,||\, (R_3(b,d) \,\&\&\, R_3(d,b)\}$
$= \min\{ 0 \,||\, (1 \,\&\&\, 1)\}$
$= 1$

$R_4(b,c) = \min\{R_3(b,c) \,||\, (R_3(b,d) \,\&\&\, R_3(d,c)\}$
$= \min\{0 \,||\, (1 \,\&\&\, 1)\}$
$= 1$

∴ The transitive closure for the given graph is

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

//.