CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test - II

Sub:	**DESIGN & ANALYSIS of ALGORITHMS**						Code:	15CS43
Date:	09 / 05/ 2017	Duration:	90 mins	Max Marks:	50	Sem: IV	Branch:	CSE/ISE

Answer Any **FIVE** Complete Questions. Please note that some figures/tables given in a question may actually be required in some other question. Hence please look very carefully at Table and Figure numbers.

		Marks	CO	RBT

1 Write the complete algorithm for HeapSort. Build the heap for the following elements that are inserted in sequence into a maxheap: **4, 1, 3, 2, 16, 9, 10, 14, 8, 7** [10] CO2 L1

OR

Construct an Optimal Binary Search Tree, given in **Table 1** [10] CO4 L3

Table 1

Set of keys	10	12	16	21
Search freq.	4	2	6	3

Table 2

Item	1	2	3
Weight	18	15	10
Profit	25	24	15

2 (a) Solve the following instance of knapsack problem using greedy algorithm. Knapsack weight M=20 given in Table 2. [05] CO4 L3

(b) Solve the above problem for the 0/1 Knapsack using Dynamic Programming [05] CO4 L3

3 (a) (i)Using Prim's algorithm, determine the minimum cost spanning tree for the graph given in **Fig.1**.
(ii) What do you mean by relaxation of an edge? Explain with example.What is the need to relax an edge? Where does this concept find its utility? [05+05] CO4 L3

Fig. 1

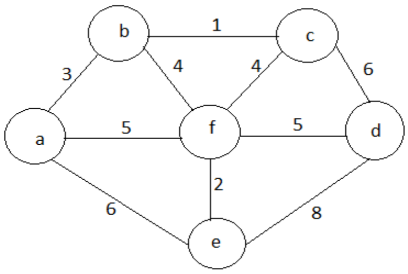
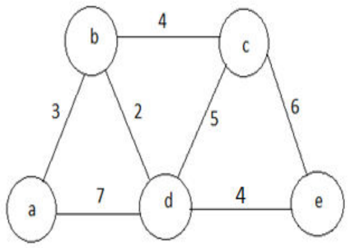


Fig.2



OR

(b) (i)What is Kruskal's technique to find the Minimun cost spanning tree? Write the algorithm.
(ii) What is the concept of Negative weight edges? What does the Bellman Ford Algorithm achieve? How is it different from Dijkstra's algorithm? [05+05] CO3 L4

4 Define the transitive closure of a graph. Describe the Warshall's algorithm to find it. Apply the same on the graph defined by the adjacency matrix in **Table 3**. [10] CO4 L3

Table 3.

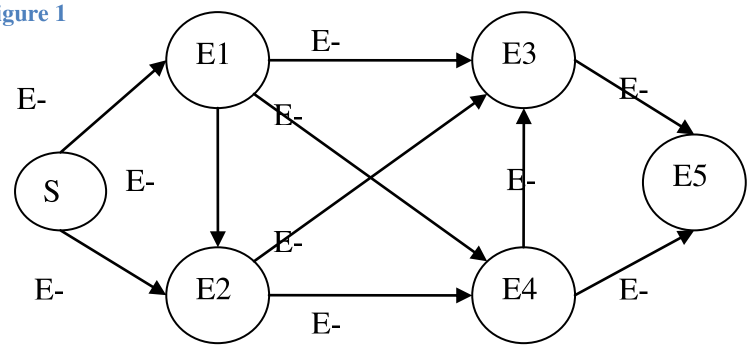
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 4.

Device	D0	D1	D2
Reliability	0.9	0.8	0.5
Cost	\$300	\$150	\$200

5	What is the concept of the dynamic programming (DP) approach to solve problems? Take an example of your choice and discuss the Multistage Graph problem solved using dynamic programming. Justify why you would choose DP over the greedy technique and the brute force.	[10]	CO4	L2
	OR			
	Explain what you understand by the term 'prefix codes'. State the Huffman's algorithm for designing the optimal prefix code.	[10]	CO2	L1
6	Describe the Dijkstra's algorithm and apply the same to find the single source shortest paths problem for the graph in the Fig. 2taking vertex '**a**' as source.	[10]	CO2	L3
7(a)	Solve the problem of Job Sequencing with deadlines, given n=7 jobs with profit (P1, P2, P3, P4, P5, P6, P7)= (3, 5, 18, 20, 6, 1, 38) and deadline (d1, d2, d3, d4, d5, d6, d7)=(1, 3, 3, 4, 1, 2, 1)	[05]	CO4	L3
(b)	What is the Reliability Design problem? Design a reliable system with the data in **Table 4** with a total Budget of $1050.	[05]	CO4	L3
8 (a)	A special IQ test is framed for 5 minutes having a total of 4 questions. Question 1 will take 2 minutes to solve, and carries 2 points. Likewise, questions 2, 3 and 4 will take 2, 3, and 1 minute(s) to solve, respectively, each carrying 5, 8, and 1 mark(s) in that order. No partial points shall be given. In the given time limit, what questions should be answered to get the maximum score in the IQ test? Apply dynamic programming to solve this.	[05]	CO5	L3
(b)	A test is conducted in a controlled environment to simulate particle motion with 5 energy levels from a particle source S. When a particle goes from one state to the other, it generally loses some amount of energy. However, in the given system, some states are at conditionally excited energy levels. Thus moving to these states from specific previous states, particle gains energy. The energy state diagram is shown below. Apply dynamic programming to plot the total energy spent by particles to move from particle source to each of the energy levels conserving maximum energy.	[05]	CO5	L3

Figure 1

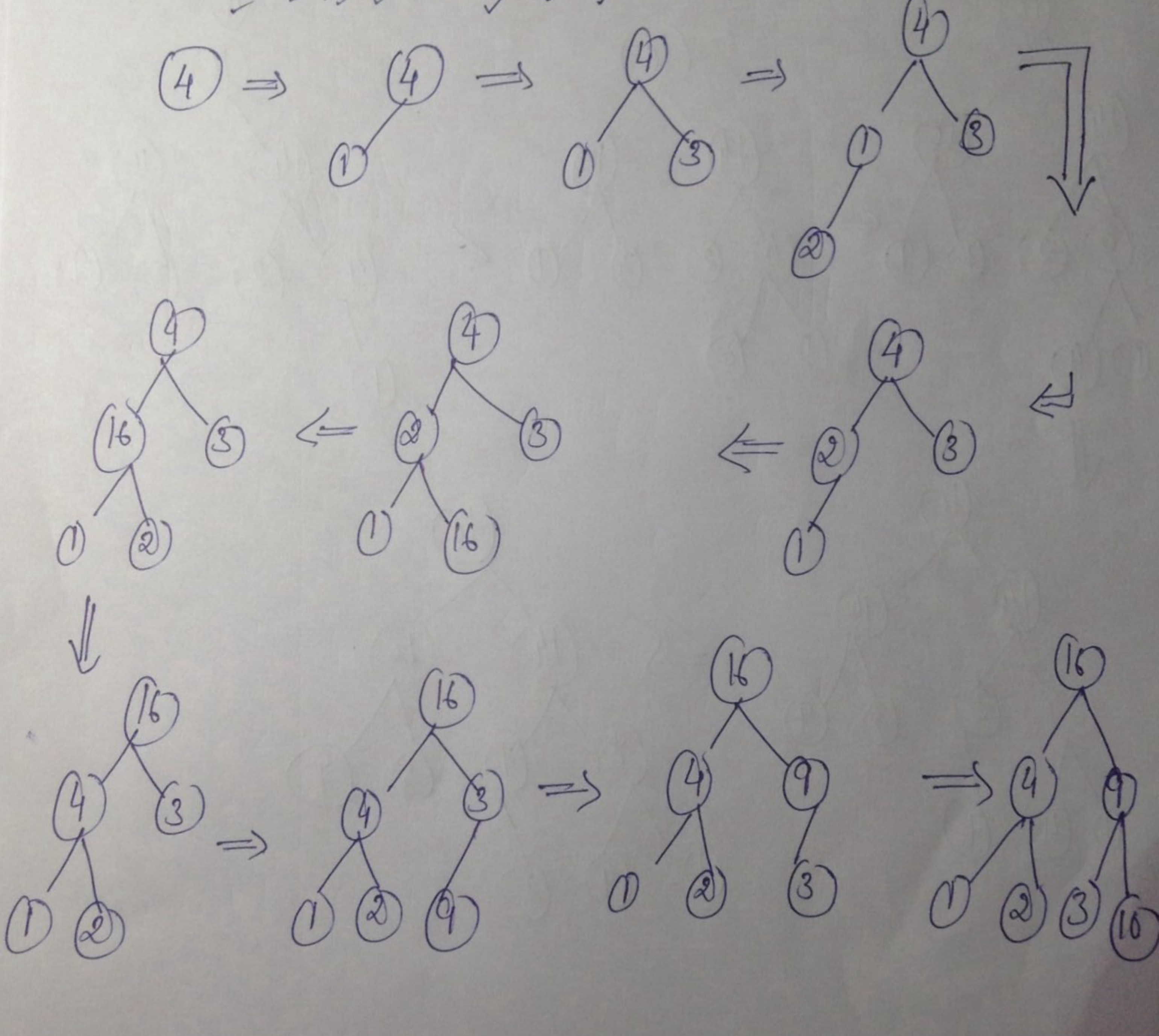


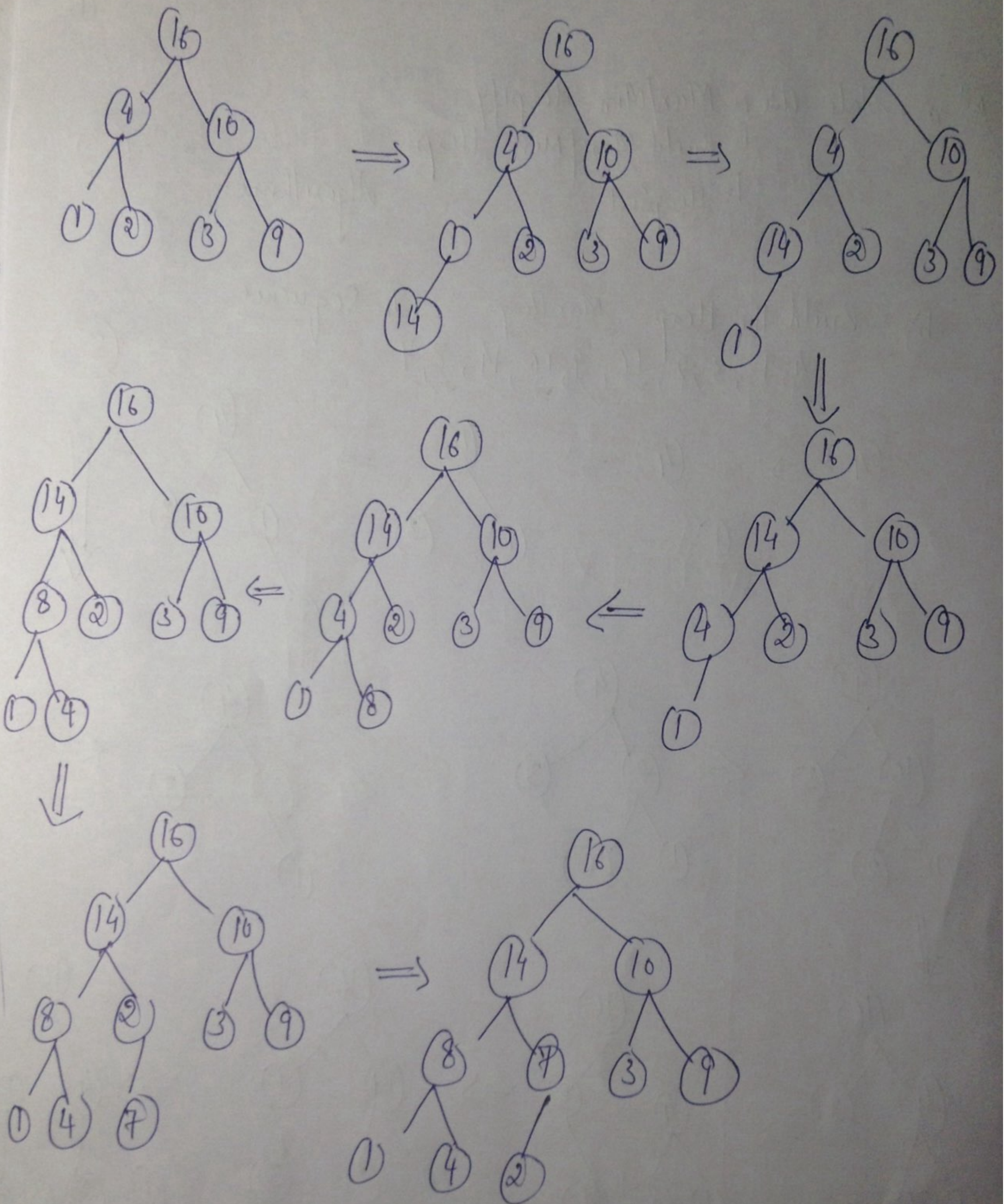
Q.1. a) State the
- Max/Min-Heapify
- Build Max/Min-Heap
- Heapsort

Algorithms. — 5

b) Build the Heap. MaxHeap In Sequence — 5

4, 1, 3, 2, 16, 9, 10, 14, 8, 7





Q.1

OR

Optimal BST. Use Dynamic Programming

Given Keys: 10 12 16 21

Search freq: 4 2 6 3

Rearrange

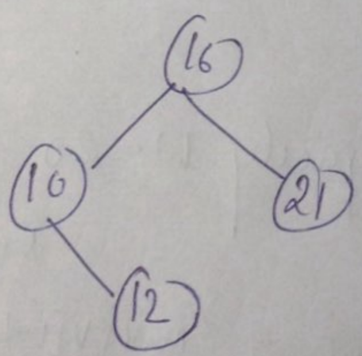
0	1	2	3
10	12	16	21
4	2	6	3

Table

	0	1	2	3
0	4	$8^{(0)}$	$20^{(2)}$	$26^{(2)}$
1		2	$10^{(2)}$	$16^{(2)}$
2			6	$12^{(2)}$
3				3

Give each step legibly calculating the table entries.

Tree: ⇒



Total Cost = 26

— 16

Q.2(a) Greedy Knapsack. $M = 20$

Item	1	2	3
wt.	18	15	10
Profit	25	24	15

~~$P[i]/w[i]$~~ =

$P_1/w_1 = 25/18 = 1.38$

$P_2/w_2 = 24/15 = 1.6$

$P_3/w_3 = 15/10 = 1.5$

Arrange: $P[i]/w[i] \geq P[i+1]/w[i+1]$

$P_2 = 24$, $P_3 = 15$, $P_1 = 25$
$w_2 = 15$, $w_3 = 10$, $w_1 = 18$

5

Item2: $\Rightarrow$ 1 $\Rightarrow \overset{P_i}{25}$ $M - 18 = 2$.

$\frac{2}{15} \times 24 : 3.2$

Total profit $= 25 + 3.2 = 28.2$

b) 0/1 Knapsack DP. ; Capacity $W = 20$

Item	wt.	value
1	18	25
2	15	24
3	10	15

Recurrence.

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

5

	i	0 ……	10	11	12	13	14	15	16	17	18	19 ~	20
	0	0	0	0	0	0	0	0	0	0	0	0	0
$w_1 = 18, v_1 = 25$	1	0	0	0	0	0	0	0	0	0	25	25	25
$w_2 = 15, v_1 = 24$	2	0	0	0	0	0	0	24	24	24	25	49	49
$w_3 = 10, v_3 = 15$	3	0	15	15	15	15	15	24	39	39	39	49	64

Q.3 (b). OR.

(i) Kruskal's Technique:

↳ Sorts E in nondecreasing order of edge wts. $w(e_{i_1}) \leq \cdots \leq w(e_{i_{|E|}})$

↳ Checks for $E_T \cup \{e_{ik}\}$ is acyclic.

↳ Explain the concept of Disjoint Sets. — 2

↳ makeSet

↳ Union

↳ findSet

ALGO to achieve the test for acyclic condn.

// I/P: Wtd. connected graph $G = (V, E)$.

// O/P: E_T: set of edges composing MST of G.

Sort such that $w(e_{i_1}) \leq \cdots \leq w(e_{i_{|E|}})$ — 3

$E_T \leftarrow \phi$; ecounter $\leftarrow 0$.

$k \leftarrow 0$

while ecounter $< |V| - 1$ do

$k \leftarrow k + 1$

if $E_T \cup \{e_{ik}\}$ is acyclic

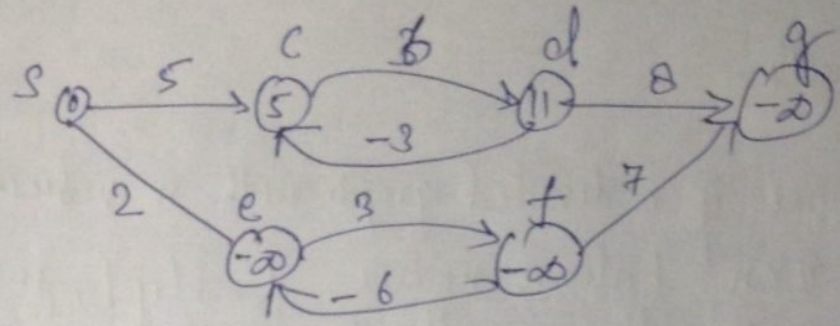
$E_T \leftarrow E_T \cup \{e_{ik}\}$; ecounter ++ ;

return E_T.

ii) Negative weight edges

If the $G = (V, E)$ contains no (-ve) wt. cycle reachable from the source s, then $\forall\ v \in V$, the shortest path wt. $\delta(s, v)$ remains well-defined, even if it has a (-ve) ~~wt.~~ value.

eg:



— 3

(1) $\langle s,c \rangle \Rightarrow$ paths
- $\rightarrow \langle s,c \rangle$
- $\rightarrow \langle s,c,d,c \rangle$
- $\rightarrow \langle s,c,d,c,d,c \rangle$ and so on

} ∞ly many paths.

Cycle $\langle cdc \rangle$ has wt $= 6-3 = 3$.

$\therefore$ Shortest path $\langle s,c \rangle = \delta(s,c) = 5$.

(2) s to f: $\Rightarrow$
- $\rightarrow \langle s,e,f \rangle$
- $\rightarrow \langle s,e,f,e,f \rangle$

} ∞ly many.

⋮ so on.

$\therefore \delta(s,f) = -\infty$ $\quad\because$ Cycle $\langle efe \rangle = 3-6 = -3$ Negative.

~~Bellf~~ Bellman-Ford :- Solves the prob. of single source shortest path. Unlike Dijkstra which has (+ve) wt. edges, Bellman-Ford considers (−ve) wt. edges as well.

Hence, uses the concept of Negative wt. edges/cycle

— 2

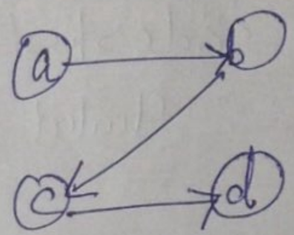
Q4. Warshall's -

Transitive closure of a directed graph with n vertices can be defined as $n \times n$ boolean matrix $T = \{t_{ij}\}$, in which the elem. in the i^{th} row, j^{th} col $(1 \le i \le n;\ 1 \le j \le n)$ is 1 if there exists a non-trivial directed path from i^{th} to j^{th} vertex; otherwise $t_{ij} = 0$.

2

Given: Adjacency Matrix

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



8

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \text{ OR } \left(r_{ik}^{(k-1)} \text{ AND } r_{kj}^{(k-1)} \right)$$

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

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

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

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

Q.5.

↳ Discuss the Dynamic Programming Technique explaining what kind of problems it solves and how.

↳ Take an example of Multistage Graph Prob.

↳ Explain the prob.

↳ Solve it using Greedy, DP & brute force — (10)

↳ Compare.

OR.

Prefix Codes: No codeword is also a prefix of some other codeword. Desirably as they simplify coding & are unambiguous. — (2)

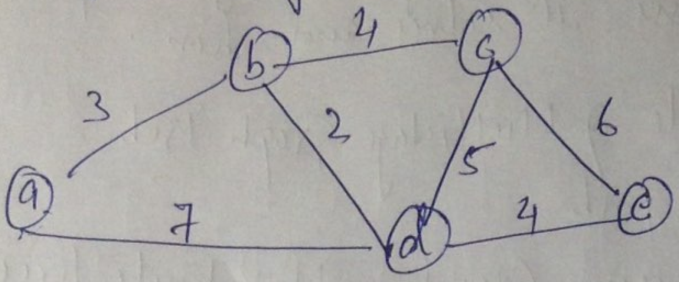
// C : set of chars.
// f[c] :- defined freq.
// Q :- Min-Priority Q keyed on f.

HUFFMAN (C)

```
n ← |C|
Q ← C
for i ← 1 to (n-1)
  do allocate a new node z
     left[z] ← x ← EXTRACT_MIN(Q)
     right[z] ← y ← EXTRACT_MIN(Q)
     f[z] ← f[x] + f[y]
     INSERT (Q, z)
return EXTRACT_MIN (Q).
```

(8)

Q6. Dijkstra's Algo.



↳ INITIALIZE-SINGLE-SOURCE (G, S)
↳ RELAX (u, v, w)
↳ DIJKSTRA (G, w, s)

} Algo. — 5

Solution: Shortest paths

a–b : 3
a–b–d : 5
a–b–c : 7
a–b–d–e : 9

} Detail the steps. ← 5

Scanned by CamScanner

Q.7(a) $n = 7$. Job Sequencing with deadlines

Profit :	38	20	18	6	5	3	1
Job :	P_7	P_4	P_3	P_5	P_2	P_1	P_6
Deadline :	1	4	3	1	3	1	2

O/P Array J[ ]. — 5

1	2	3	4	5	6	7
P_7						

P_7			P_4			

P_7		P_3	P_4			

P_7		P_3	P_4			

Discard P_5

P_7	P_2	P_3	P_4			

Discard P_1, P_6

P_7	P_2	P_3	P_4			

Final seq: → $P_7 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$.

Profit = 38 + 5 + 18 + 20 = 81

b). Solve using the Dynamic Programming approach. — 5

D_0	D_1	D_2
1	2	2
\$300	\$300	\$400

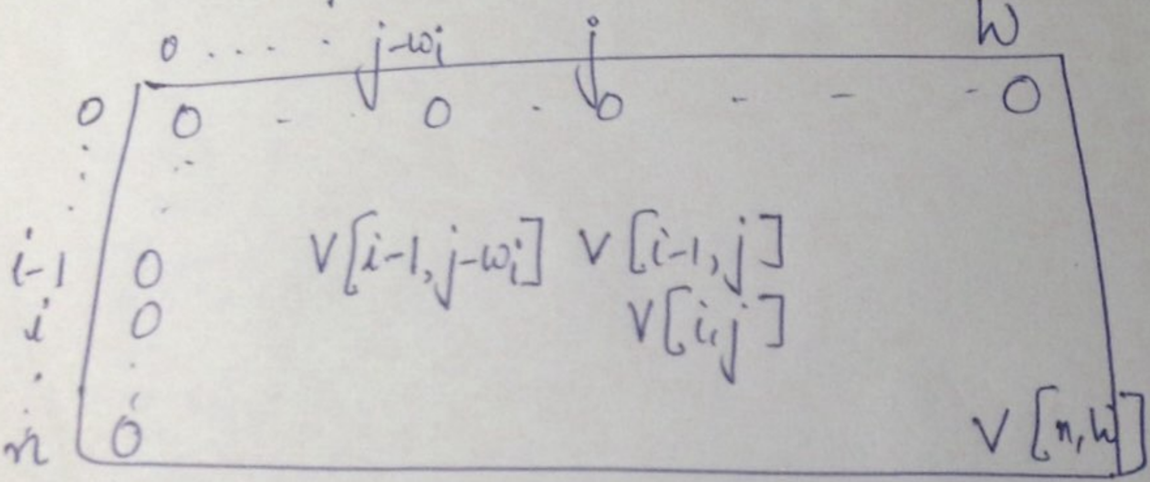
Reliability $\Pi = 0.648$

Cost incurred \$1050

Q.8(a). 0/1 Knapsack Problem. $W = 5$ mins.

Question	Time	Points
1	2	2
2	2	5
3	3	8
4	1	1

5



capacity j

	i	0	1	2	3	4
	0	0	0	0	0	0
$w_1=2, v_1=2$	1	0	0	2	2	2
$w_2=2, v_2=5$	2	0	0	5	7	7
$w_3=3, v_3=8$	3	0	0	5	13	15
$w_4=1, v_4=1$	4	0	1	5	13	15

8(b): Approach this connected, weighted (-ve) edges Graph, G = (V, E) as a Single Source Shortest Path problem.
Use Bellman Ford. — 5