CBCS SCHEME

USN 1 C R 2 2 M C 0 1 1

22MCA15

First Semester MCA Degree Examination, Jan./Feb. 2023

Design and Analysis of Algorithms

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Define algorithm. Illustrate the characteristics of an algorithm.	06	L1	CO1
	b.	List and explain the different problem types of an algorithm.	04	L1	CO1
	c.	Explain asymptotic notations, with a diagram and explain with an example.	10	L2	CO1

A)

An Algorithm is defined as finite sequence of unambiguous instructions followed to accomplish particular task.

Characteristics of an algorithm :

* Input - One or more quantity can be supplied
* Output - Atleast one quantity is produced.
* Definitness - Each instruction of the algorithm should be clear & unambiguous
* Finiteness - The process should be terminated after a finite number of steps.
* Effectiveness - Every instruction must be basic enough to be carried out theoretically or by using "pencil" & "paper".

B)

* Sorting - The problem involving the rearrangement of the item of a given list in some particular order is known as "Sorting Problem".

* Searching - The searching problem deals with searching of a given value called "key" in the given list

* String processing - In rapid growing application of computer science, processing of text takes an important role.

Searching for a particular word in a text is known as "String processing".

* Graph problem - Graph is a set of points known as "vertices" some of which may be connected by a line/curve called "edge"

* Combinatorial problem - The problem involving combinatorial elements like permutation, combination, sets etc are known as "combinatorial problem".

* Numerical problem: It involves solving system of equations, computing definite integrals etc

C)

Asymptotic notation: Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

There are types of notations, they are:

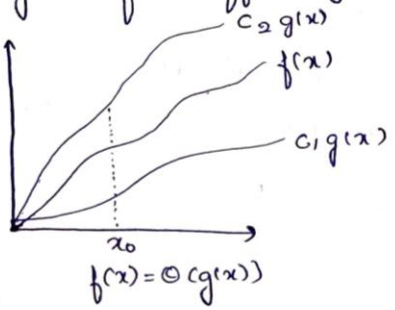
1] Θ - notation
2] O - notation
3] Ω - notation

1] Θ notation

For a given function $g(x)$, we denote $\Theta(g(x))$ the set of functions.

$\therefore \Theta(g(x)) = \{ f(x)$: there exists positive constants c_1, c_2 & x_0 such that $0 \leq c_1 g(x) \leq f(x) \leq c_2 g(x)$ for all $x \geq x_0\}$

A function $f(x)$ belongs to the set $\Theta(g(x))$, if there exists positive constants c_1 & c_2 such that it can be sandwiched b/w $c_1(g(x))$ & $c_2(g(x))$ for sufficiently large x.



2] O - notation

For a given function $g(x)$ we denote by $O(g(x))$ the set of function.

$\therefore O(g(x)) = \{ f(x)$: there exist positive constant c & x_0 such that $0 \leq f(x) \leq c(g(x))$ for all $x \geq x_0 \}$

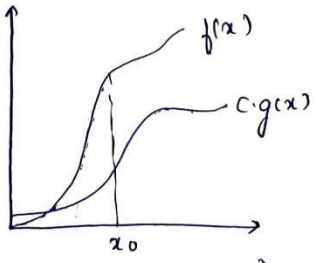
we use 'O' notation to give an appropriate upperbound as function. If $f(x)$ is bounded above by some constant multiple of $g(x)$ for all large x.

3] Ω - notation

For a given function $g(x)$, we denote by $\Omega(g(x))$ the set of functions.

$\therefore \Omega(g(x)) = \{ f(x)$: there exist positive constant c & x_0 such that $0 \leq g(x) \leq f(x)$ for all $x \geq x_0 \}$

we use Ω notation to give an lowerbound for function. If $f(x)$ is bounded below by some constant multiple of $g(x)$ for all large x.



$f(x) = \Omega(g(x))$

Eg: In bubble sort, when the input array is already sorted, the time taken by the algorithm is ~~tea~~ linear ie the best case.

Q.2	a.	Design a general plan for analyzing the non recursive algorithm. Design an algorithm for element uniqueness in an array. Obtain its time complexity.	12	L3	CO2
	b.	List and explain the fundamental data structures with definition and examples.	08	L1	CO2

A)

The general plan for analysis is :

- Based on input size, decide the various parameters to be considered
- Identify the "Basic operation" of algorithm.
- Compute the no of times a basic operation is executed
- Obtain the total no of time a basic operation is executed.
- Simplify using standard formula & compute the order of growth

The formula's used for analysis are

1. $\sum_{i=1}^{u} c \cdot a_i = c \sum_{i=1}^{u} a_i$
2. $\sum_{i=1}^{u} (a_i \pm b_i) = \sum_{i=1}^{u} a_i + \sum_{i=1}^{u} b_i$
3. $\sum_{i=1}^{u} 1 = \text{Upperlimit} - \text{Lowerlimit} + 1$
4. $\sum_{i=0}^{u} i = \sum_{i=1}^{u} 1 = 1+2+\cdots n = \frac{n(n+1)}{2}$
5. $\sum_{i=1}^{u} i^2 = 1^2 + 2^2 + \cdots + n^2$
   $= \frac{n(n+1)(2n+1)}{6}$
   $= \frac{1}{3} n^3$

Algorithm for element uniqueness in an array :

```
Algorithm Unique A([0.... n-1])
i/p: An array A[0.... n-1]
o/p: True or False
    for i ← 0 to n-2 do
        for j ← i+1 to n-1 do
            if A[i] == A[j]
                return False
    return TRUE
```

Analysis:

- The parameter to be considered here is input size 'n'
- The basic operation is "comparison"
- The algorithm stops, when encounter duplicate in list, thus algorithm depends on 'n'

But we can't give the position of duplicate, so we will assume the worstcase treating all comparisons are made

Can be written as $\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1}$

$$\therefore T(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} \left[(n-1) - (i+1) + 1 \right]$$

$$= \sum_{i=0}^{n-2} \{ n - 1 - i \}$$

$$= (n-1) + (n-2) + (n-3) \cdots 3 + 2 + 1 + 0$$

$$= 1 + 2 + 3 + \cdots (n-1)$$

$$= n(n-1)/2$$

$$= n^2/2$$

$\therefore$ Time Complexity, $T(n) = O(n^2)$ //

a) List and explain the fundamental data structures with definition and examples.

definition and examples.

Most of algorithms operates on data, so organization of data plays an important role in design & analysis of algorithm.

Types of datastructures are:

1. Stack
2. Queue
3. Graphs
4. Tree
5. Linked List

5. Linked list:

1. Stack: It is an abstract data type (ADT) commonly used in most programming language.
A stack is a list of elements in which an element may be inserted / deleted only at one end called "top" of the stack.
Stack follow LIFO concept as the items can be added or removed only from the top.
Two basic operations are: 1] Push 2] Pop

2. Queue: A queue is a linear list of elements in which deletion can take place only at one end called "front" & insertion can take place only at the other end called "rear".

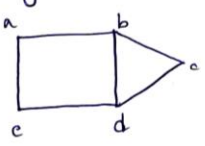
Two basic operation associated with it 1] insert 2] delete

3] Graphs: A graph 'G' is defined as a pair of two sets of V & E denoted by G = (V, E)
Where V is set of vertices, E is set of edges.
Two types of graphs are:
1] Directed graph - Edge is directed
2] Undirected graph - Edge is undirected

4] Tree: Tree is a data structure used to represent linear relationship existing among several data item. Here each data item is referred as node.
Basic operations include: 1] Search 2] Insertion 3] Traversal

5] Linked list: In algorithm, graphs are represented using
1] Adjacency matrix 2] Adjacency linked list
Adjacency matrix will be interms of '0' & '1'

Ex:

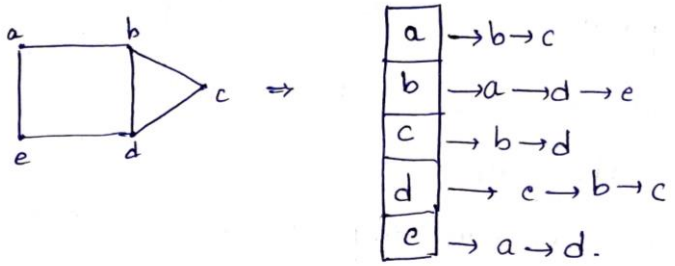


=>

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

Adjacency linked list of a graph is a collection of linked list, one for each vertex that contains all the vertices adjacent to the list vertex.

Ex:



		Module – 2			
Q.3	a.	Design a binary search algorithm and derive its time complexity and apply the same to search an element "42" from the given elements. 3, 14, 27, 31, 40, 42, 55, 66	12	L3	CO2
	b.	Sort the following elements using Quick Sort. Show only 1st partition. 50, 30, 10, 90, 60, 40, 35, 62	08	L3	CO2

A)

⇒ Algorithm

BinarySearch(A[0...n-1], k)

```
int bin_search(int arr[], int x){
  int start, end;
  int mid = (start+end)/2;
  if (arr[mid] == x){
        return mid;
  }
  if (arr[mid] > x){
        return end = mid-1;
  }
  else {
        return start=mid+1;
  }
  return -1;
}
```

Pseudo Code

BinarySearch(A[0..n-1], k)

- Input: An array A[0..n-1]
- Sorted in ascending order
- A search key K

```
l ← 0;
r ← n-1
while l <= r do
    m ← (l+r)/2
    if K = A[m] return m
    else if K < A[m]
        return r ← m-1
    else
        l ← m+1
return -1
```

Time complexity of Binary Search Algorithm is :- $O(\log n)$

⇒ 3, 14, 27, 31, 40, 42, 55, 66

Target element : 42

Step 1: start : 0
end : 7
midpoint index : (0+7)/2 = 3
- Midpoint value = 31

Step 2 : Compare the midpoint value with target value
- Midpoint value (31) < Target value (42)
- Since midpoint value is less than target value, we search on the right half of the array.

New search range : [40, 42, 55, 66]

Step 3: Identify new midpoint
- Start index : 4
- end index : 7
- midpoint index : (4+7)/2 = 5
- midpoint value : 42

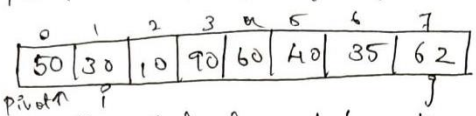
Step 4:
Compare the new midpoint value with the target value.
- Midpoint value (42 = Target value (42)
- Element Found at index 5 in the sorted array.

∴ By using Binary search, we have found the element 42 in the given array [3, 14, 27, 31, 40, 42, 55, 66] at index 5.

B)

50, 30, 10, 90, 60, 40, 35, 62.

=>



① Choose the pivot element from the array

② Let pivot element be the first element in the array

③ Choose the next element of pivot as 'i' element

Step ④ Choose the last element of the array index as 'j' element.

Step ⑤ Increment 'i' until it is greater than the pivot element

a[3] = 90 a[pivot] = 50

a[3] > a[pivot]

Step ⑥ Decrement 'j' until it is less than equal to pivot element.

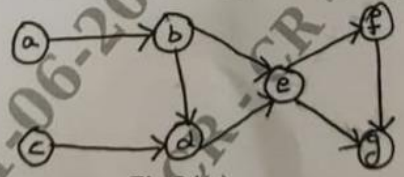
a[6] = 35, a[pivot] = 50

a[6] < a[pivot]

Step ⑦ After 'i' and 'j' has met its conditions then the user needs to swap a[i] and a[j] element then again the above steps needs to be done

Step ⑧ Once the increment is done in the array if 'i' > 'j' and 'i' element should be greater pivot, stop swap the a[j] and a[pivot]. Now the pivot element position is found.

Step ⑨ In the array a[pivot], left side of it will be less than pivot.

Q.4	a.	Define Topological Sorting. Obtain the topological ordering of elements using DFS and Source Removal Method, for the following graph Fig.Q4(a):  Fig.Q4(a)	10	L3	CO2
	b.	Define Heap. Explain different types of heap. Sort the following elements using heap sort technique by creating bottom-up max heap tree. 26, 14, 18, 42, 6, 9, 38	10	L3	CO1

A)

Topological sorting is a way of ordering the nodes in a directed graph so that for every directed edge uv, the node u comes before the node v. This can be done using the following steps:

1. Perform a DFS on the graph.
2. For each node that is visited, add it to a stack.
3. Pop the elements from the stack in reverse order.

The source removal method is a way of finding the topological ordering of elements in a directed graph with no cycles. It works as follows:

1. Initialize a set of visited nodes.
2. While the set of visited nodes is not equal to the set of all nodes:
   - Find a node that has no incoming edges and add it to the set of visited nodes.
   - For each node that is adjacent to the node that was just added, remove the edge from the graph.

The topological ordering of elements in the graph in the image can be found using the following steps:

- Perform a DFS on the graph.
- The nodes that are visited in the reverse order of the DFS traversal will be the topological ordering of the elements.

B)

⇒ A Heap is a specialized tree-based data structure that satisfies the heap property.
The heap property states that for a given node in the heap, the value of that node is either greater than or equal to (in a max heap) or less than or equal to in a min heap the values of its children nodes.
There are two main types of heaps:

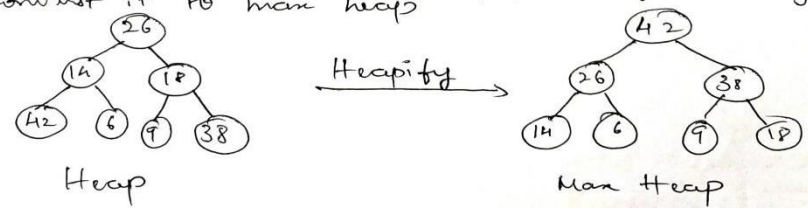
① Max Heap: In a max heap, the value of each node is greater than or equal to the values of its children. This means that the root node of a max heap contains the maximum element in the heap. The max heap property allows efficient retrieval of the maximum element in O(1), while inserting and deleting elements take O(log n) time complexity.

② Min Heap: In a min heap, the value of each node is less than or equal to the values of its children. This means that the root node of a min heap contains the minimum element in the heap. Similar to the max heap, a min heap allows efficient retrival of the minimum element in O(1) time, O(log n) time complexity

⇒

26	14	18	42	6	9	38

① First we have to construct a heap from the given array and then convert it to max heap

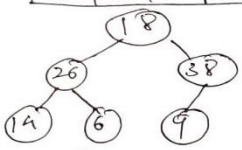


(2) | 42 | 26 | 38 | 14 | 6 | 9 | 18 |

Swap 42 and 18

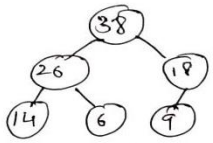
42 is sorted at its position.

| 18 | 26 | 38 | 14 | 6 | 9 | 42 |



Heap

Heapify →

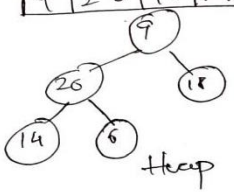


Max Heap

(3) | 38 | 26 | 18 | 14 | 6 | 9 | 42 |

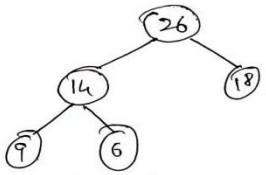
Swap 38 and 9

| 9 | 26 | 18 | 14 | 6 | 38 | 42 |



Heap

Heapify →

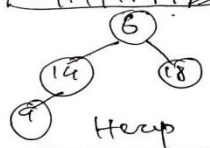


Max Heap

(4) | 26 | 14 | 18 | 9 | 6 | 38 | 42 |

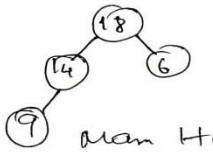
Swap 26 and 6

| 6 | 14 | 18 | 9 | 26 | 38 | 42 |



Heap

Heapify →

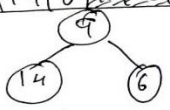


Max Heap

(5) | 18 | 14 | 6 | 9 | 26 | 38 | 42 |

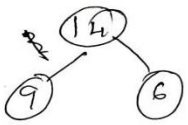
18 swap and 9

| 9 | 14 | 6 | 18 | 26 | 38 | 42 |



Heap

Heapify →



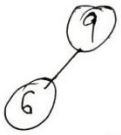
(6) | 14 | 9 | 6 | 18 | 26 | 38 | 42 |

Swap 6 and 14

| 6 | 9 | 14 | 18 | 26 | 38 | 42 |

Heap

Heapify →



Max Heap

(7) | 9 | 6 | 14 | 18 | 26 | 38 | 42 |

Swap 9 and 6

| 6 | 9 | 14 | 18 | 26 | 38 | 42 |

(6)

Heapify

(6)

∴ The sorted Elements are [6, 9, 14, 18, 26, 38, 42].

Module – 3

Q.5	a.	Consider the following jobs with their profits and deadlines. Find the executing job sequence using greedy to obtain max profit. <$p_1, p_2, p_3, p_4, p_5, p_6$> = <23, 45, 6, 18, 60, 5> <$d_1, d_2, d_3, d_4, d_5, d_6$> = <3, 2, 1, 4, 2, 1>	10	L3	CO2
	b.	Apply Kruskal algorithm to find Minimum Spanning tree for the below graph, Fig.Q5(b).	10	L3	

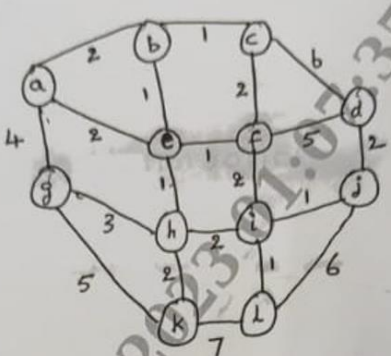


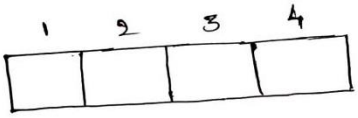
Fig.Q5(b)

A)

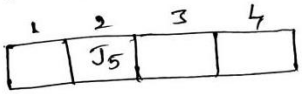
Firstly, Sort all given jobs in the decreasing order of their profit.

Jobs	J_5	J_2	J_1	J_4	J_3	J_6
Deadline	60	45	23	18	6	5
Profits	2	2	3	4	1	1

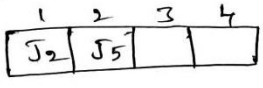
The max deadline = 4 so draw a chart with max time on chart = 4 units.



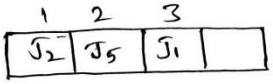
→ Now, take each job one by one in order & place them as far as possible within deadline, Firstly we take job 5 and place it in its deadline 2.



→ Next J_2, its deadline is 2 but, 2 is occupied with J_5. so, place it in deadline 1.



→ Next J_1, its deadline is 3.



→ Next J_4, its deadline is 4.

1	2	3	4
J_2	J_5	J_1	J_4

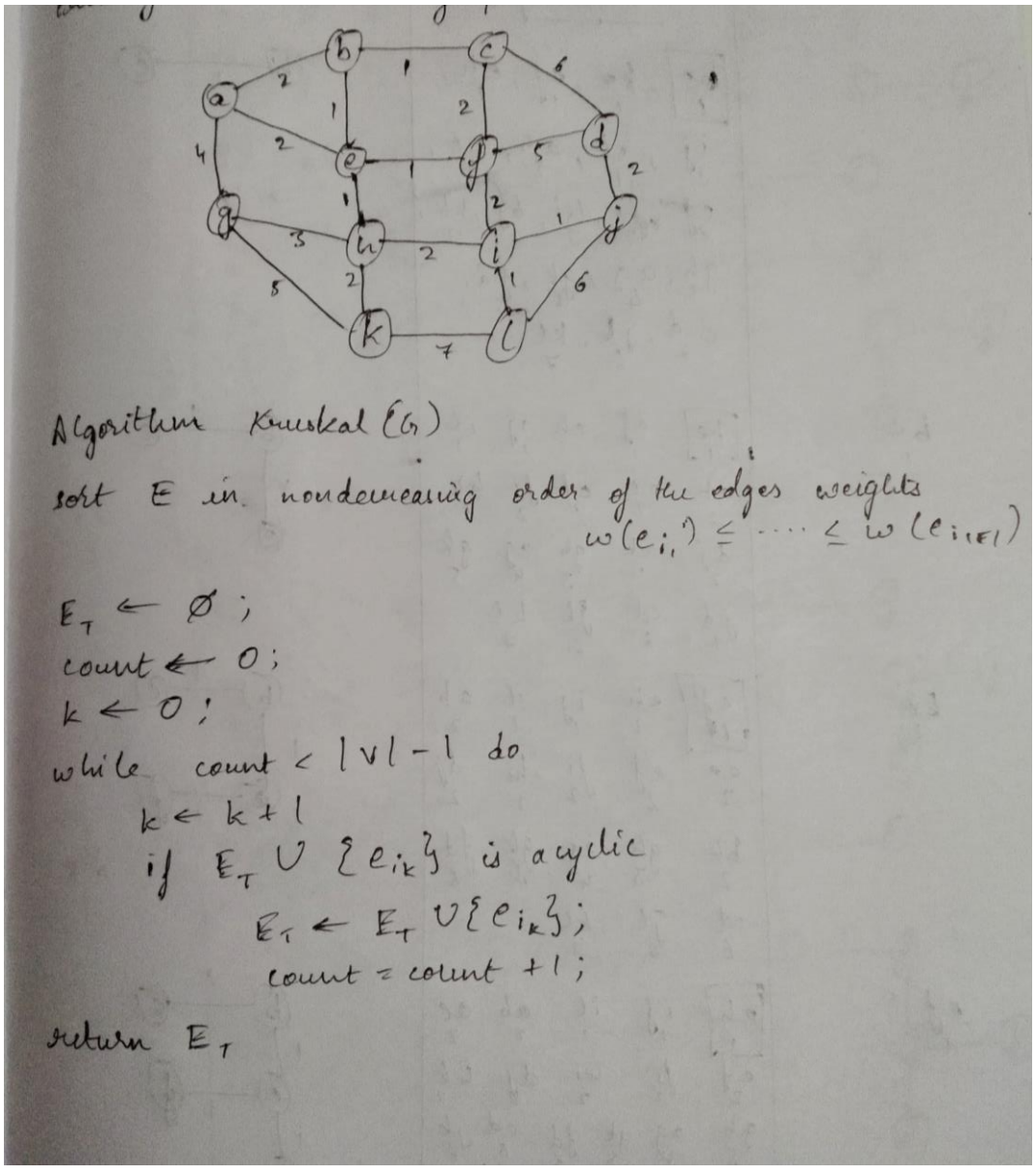
∴ The optimal schedule is

(J_2, J_5, J_1, J_4) //

∴ Total profit earned is

$45 + 60 + 23 + 18 = 146$

B)



Algorithm Kruskal (G)

sort E in nondecreasing order of the edges weights

$w(e_{i_1}) \leq \cdots \leq w(e_{i_{|E|}})$

$E_T \leftarrow \emptyset$;

count $\leftarrow 0$;

$k \leftarrow 0$;

while count $< |V| - 1$ do

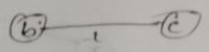
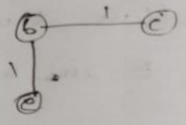
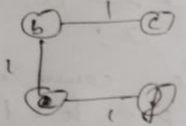
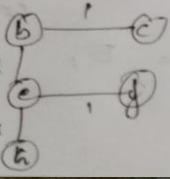
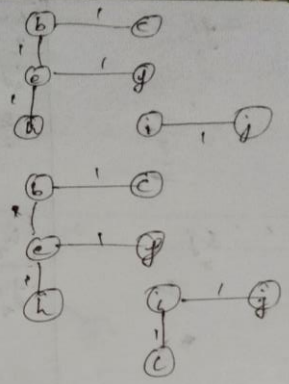
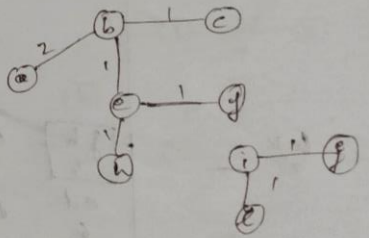
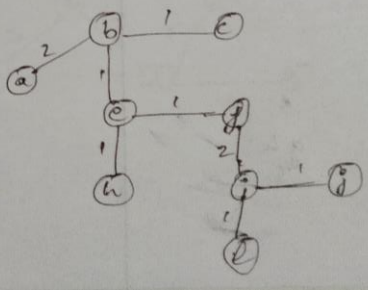
$k \leftarrow k + 1$

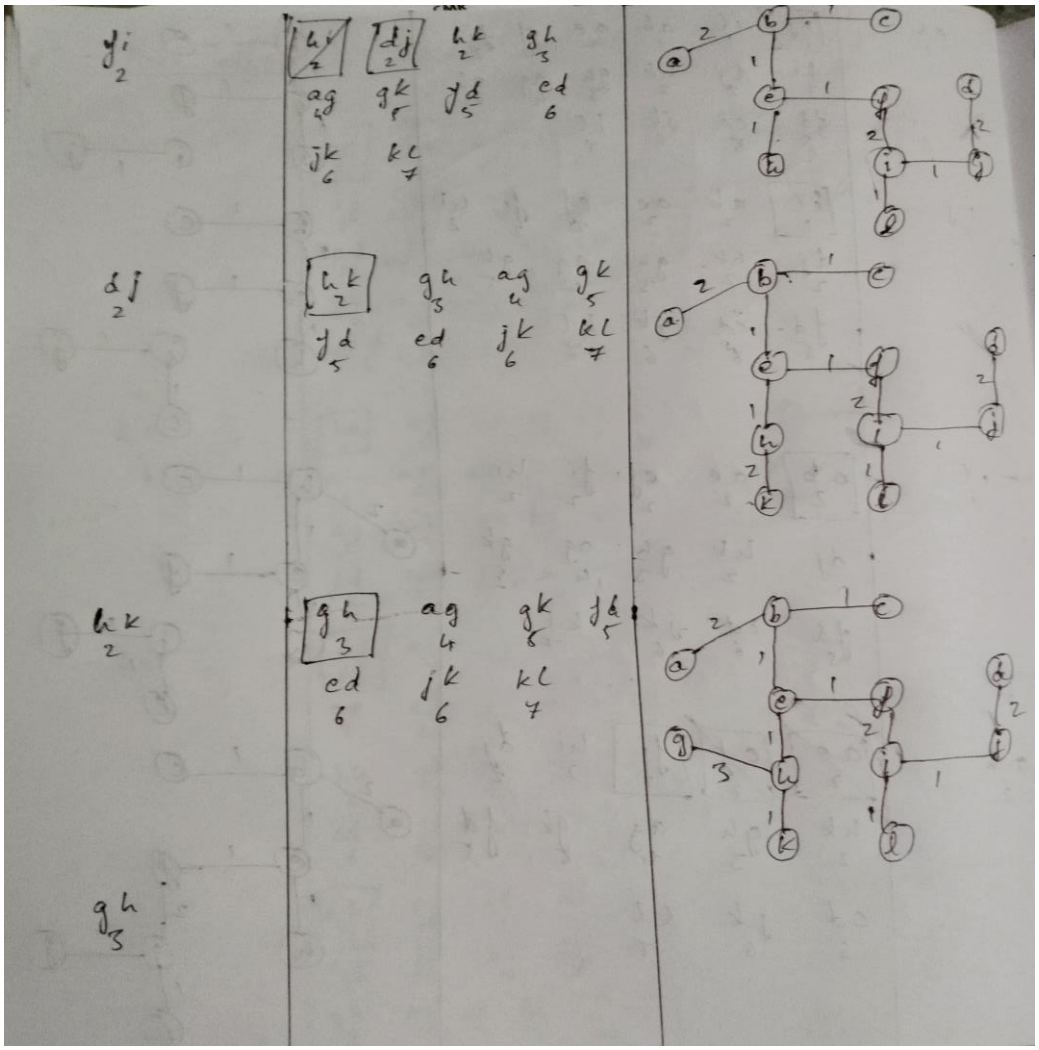
if $E_T \cup \{e_{i_k}\}$ is acyclic

$E_T \leftarrow E_T \cup \{e_{i_k}\}$;

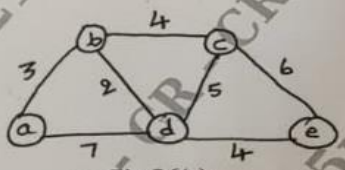
count = count + 1;

return E_T

Tree edges	Sorted list of edges	Graph
	[be]$_1$, bf$_1$, ef$_1$, eh$_1$, ij$_1$, il$_1$, ab$_2$, ae$_2$, ef$_2$, fi$_2$, hi$_2$, dj$_2$, hk$_2$, gh$_3$, ag$_4$, gk$_5$, fd$_5$, cd$_6$, jl$_6$, kl$_7$	
be$_1$	[be]$_1$ ef$_1$ eh$_1$ ij$_1$ il$_1$ ab$_2$ ae$_2$ ef$_2$ fi$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jl$_6$ kl$_7$	
ef$_1$	[ef]$_1$ eh$_1$ ij$_1$ il$_1$ ab$_2$ ae$_2$ fi$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jl$_6$ kl$_4$	
eh$_1$	[eh]$_1$ ij$_1$ il$_1$ ab$_2$ ae$_2$ ef$_2$ fi$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jk$_6$ kl$_7$	
ij$_1$	[ij]$_1$ il$_1$ ab$_2$ ae$_2$ ef$_2$ fi$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jk$_6$ kl$_7$	
il$_1$	[il]$_1$ ab$_2$ ae$_2$ ef$_2$ fi$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jk$_6$ kl$_7$	
ab$_2$	[ab]$_2$ ae$_2$ ef$_2$ fi$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jk$_6$ kl$_7$	
ae$_2$ ef$_2$ fi$_2$	[ae]$_2$ [ef]$_2$ [fi]$_2$ hi$_2$ dj$_2$ hk$_2$ gh$_3$ ag$_4$ gk$_5$ fd$_5$ cd$_6$ jk$_6$ kl$_7$	


hi 2
dj 2
hk 2
gh 3
ag 4
gk 5
jd 5
ed 6
jk 6
kl 7
dj 2
hk 2
gh 3
ag 4
gk 5
jd 5
ed 6
jk 6
kl 7
hk 2
gh 3
ag 4
gk 8
jd 5
ed 6
jk 6
kl 7
gh 3

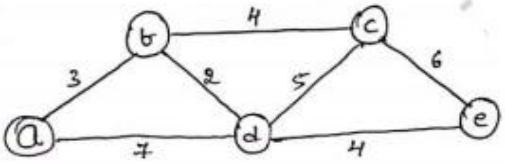
OR

Q.6	a.	Find the shortest path from the vertex "d" to all other vertices for the below graph, Fig.Q6(a).  Fig.Q6(a)	10	L3	CO2
	b.	Construct a Huffman code for the following data:	10	L3	CO2

Character	A	B	C	D	-
Prob	0.35	0.1	0.2	0.2	0.15

Find: (i) Huffman tree (ii) Decode the string "1001101101001101"

A)



⇒

Tree vertices	Remaining vertices
* a (-, 0)	b (a, 3), c (-, ∞), d (a, 7), e (-, ∞)
* b (a, 3)	c (b, 3+4), d (b, 3+2), e (-, ∞)
* d (b, 5)	c (b, 7), e (d, 5+4)
* c (b, 7)	e (d, 9)
* e (d, 9)	— × —

→ The Shortest path Set is given as

From A to B : A → B dist = 3

A to C : A → B → C dist = 7

A to D : A → B → D dist = 5

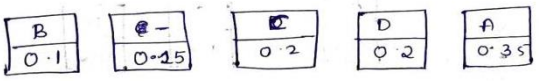
A to E : A → B → D → E dist = 9

B)

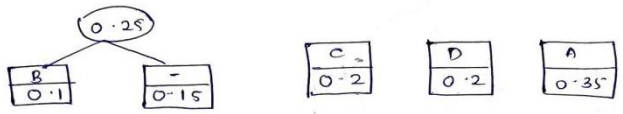
* Pick the last prob of characters first and continue to same.

1) A = 0.35
B = 0.1
C = 0.2
D = 0.2
- = 0.15

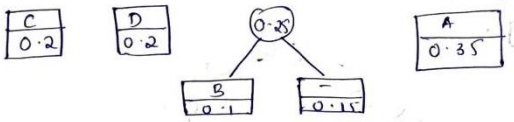
First step, create 5 one-node tree as shown below in ascending order.



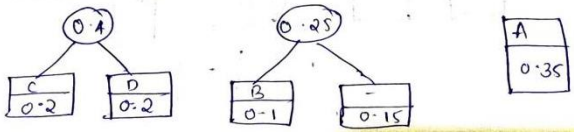
Choose the smallest two nodes/subtree & construct a new tree with root node consisting of their sum



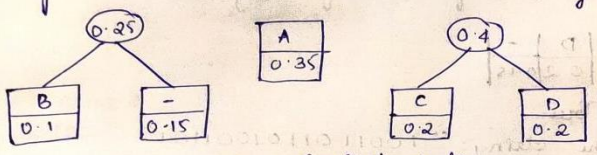
2) Again sort the subtree in ascending order



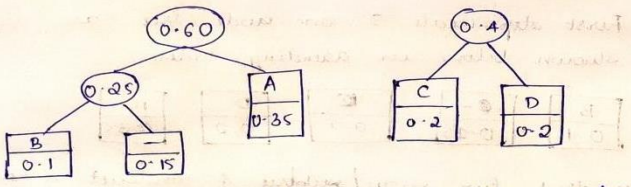
Choose the smallest two nodes



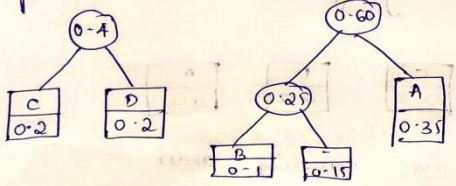
3) Again sort the subtree in ascending order



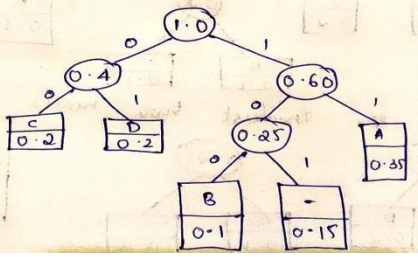
Choose the two smallest node



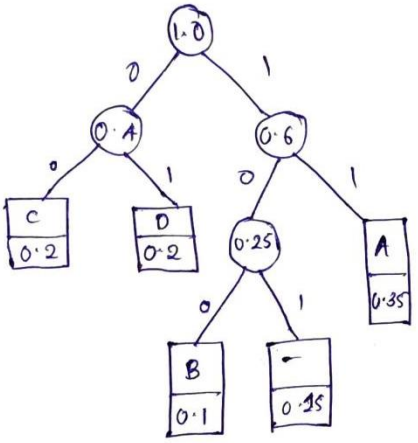
4) Again sort the subtree in ascending order



Finally after combining,



i) Huffman Tree



A → 11
B → 100
C → 00
D → 01
- → 101

ii) Decode 1001101101001101

B A D - C A D

=> BAD-CAD

Module – 4

Q.7	a.	Apply Forward approach Multistage graph technique to find shortest path from S to t.	10	L3	CO3

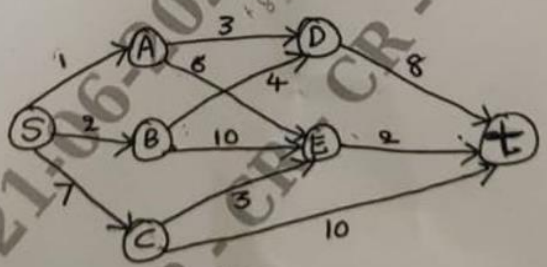


Fig.Q7(a)

	b.	Find all pair shortest path for the below graph, Fig.Q7(b).	10	L3	CO2

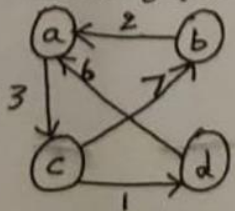
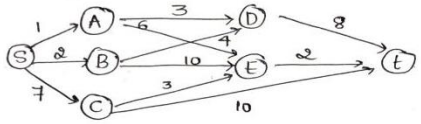


Fig.Q7(b)

A) Q.7

a] Apply Forward approach Multistage graph technique to find shortest path from S to t.



S1 S2 S3 S4

V	S	A	B	C	D	E	t
Cost	9	8	12	5	8	2	0
d	A	E	D/E	E	t	t	t

Formula:

$$Cost(i,j) = \min\{c(j,l) + Cost(i+1,l)\}$$

i → Stage
j → vertex
l → Next connecting vertex

Calculate the distances from S1 to S4.

→ At Stage 4 ie t to t the distance is zero so mark the distance as zero.

→ At Stage 3, use the formula

i] $Cost(3,D) = \min\{c(D,t) + Cost(4,d)\}$
$= \min\{8+0\}$
$Cost(3,D) = 8$

ii] $Cost(3,E) = \min\{c(E,t) + Cost(4,t)\}$
$= \min\{2+0\}$
$Cost(3,E) = 2$

→ At stage 2

$Cost(2,A) = \min\{c(A,D) + Cost(3,D), c(A,E) + Cost(3,E)\}$
$= \min(8+8, 6+2)$
$Cost(2,A) = 8$ //

$Cost(2,B) = \min\{c(B,D) + Cost(3,D), c(B,E) + Cost(3,E)\}$
$= \min\{4+8, 10+2\}$
$Cost(2,B) = 12$ //

$Cost(2,C) = \min\{c(C,E) + Cost(3,E), c(C,t) + Cost(3,t)\}$
$= \min\{3+2, 10+0\}$
$Cost(2,C) = 5$ //

→ At stage 1

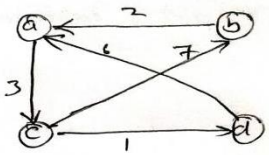
$Cost(1,S) = \min\{c(1,A) + Cost(2,A), c(1,B) + Cost(2,B), c(1,C) + Cost(2,C)\}$
$= \min\{1+8, 2+12, 7+5\}$
$= \min\{9, 14, 12\}$
$Cost(1,S) = 9$ //

∴ Now the table is filled.

The final path is: S → A → E → t

B)

b. Find all pair shortest Path for the below graph.



=> Matrix

$$A^0 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & 0 & \infty & 0 \end{bmatrix} \end{array}$$

consider a=1, b=2, C=3, d=4

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

Formula to find all pair Shortest Path is

$$A^k[i,j] = \min\{A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j]\}$$

① K=1

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & 0 & 9 & 0 \end{bmatrix} \end{array}$$

For A^1, write 1st row & column same as the previous Matrix

$A^1[2,3] = \min\{A^0[2,3], A^0[2,1] + A^0[1,3]\}$
$= \min\{\infty, 2+3\}$
$= 5$

$A^1[2,4] = \min\{A^0[2,4], A^0[2,1] + A^0[1,4]\}$
$= \min\{\infty, 1+\infty\}$
$= \infty$

$A^1[3,2] = \min\{A^0[3,2], A^0[3,1] + A^0[1,2]\}$
$= \min\{7, \infty+\infty\}$
$= 7$

$A^1[3,4] = \min\{A^0[3,4], A^0[3,1] + A^0[1,4]\}$
$= \min\{1, \infty+\infty\}$
$= 1$

$A^1[4,2] = \min\{A^0[4,2], A^0[4,1] + A^0[1,2]\}$
$= \min\{0, 6+\infty\}$
$= 0$

$A^1[4,3] = \min\{A^0[4,3], A^0[4,1] + A^0[1,3]\}$
$= \min\{\infty, 6+3\}$
$= 9$

② K=2

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ 9 & 7 & 0 & 1 \\ 2 & 0 & 5 & 0 \end{bmatrix} \end{array}$$

For A^2, write 2nd row & column same as the previous Matrix

$A^2[1,3] = \min\{A^1[1,3], A^1[1,2] + A^1[2,3]\}$
$= \min\{3, \infty+5\}$
$= 3$

$A^2[1,4] = \min\{A^1[1,4], A^1[1,2] + A^1[2,4]\}$
$= \min\{\infty, \infty+\infty\}$
$= \infty$

$A^2[3,1] = \min\{A^1[3,1], A^1[3,2] + A^1[2,1]\}$
$= \min\{\infty, 7+2\}$
$= 9$

$A^2[3,4] = \min\{A^1[3,4], A^1[3,2] + A^1[2,4]\}$
$= \min\{1, 2+\infty\}$
$= 1$

$A^2[4,1] = \min\{A^1[4,1], A^1[4,2] + A^1[2,1]\}$
$= \min\{6, 0+2\}$
$= 2$

$A^2[4,3] = \min\{A^1[4,3], A^1[4,2] + A^1[2,3]\}$
$= \min\{9, 0+5\}$
$= 5$

(3) $K=3$.

$$A^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 2 & 0 & 5 & 0 \end{bmatrix} \end{array}$$

For A^3, write 3rd row & Column same as the Previous Matrix

$A^3[1,2] = \min\{A^2[1,2], A^2[1,3] + A^2[3,2]\}$
$= \min\{\infty, 3+7\}$
$= 10$

$A^3[1,4] = \min\{A^2[1,4], A^2[1,3] + A^2[3,4]\}$
$= \min\{\infty, 3+1\}$
$= 4$

$A^3[2,1] = \min\{A^2[2,1], A^2[2,3] + A^2[3,1]\}$
$= \min\{2, 5+9\}$
$= 2$

$A^3[2,4] = \min\{A^2[2,4], A^2[2,3] + A^2[3,4]\}$
$= \min\{\infty, 5+1\}$
$= 6$

$A^3[4,1] = \min\{A^2[4,1], A^2[4,3] + A^2[3,1]\}$
$= \min\{2, 5+9\}$
$= 2$

$A^3[4,2] = \min\{A^2[4,2], A^2[4,3] + A^2[3,2]\}$
$= \min\{0, 5+7\}$
$= 0$

(4) $K=4$

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

For A^4, write 4th row and Column same as the previous Matrix.

$A^4[1,2] = \min\{A^3[1,2], A^3[1,4] + A^3[4,2]\}$
$= \min\{10, 4+0\}$
$= 4$

$A^4[1,3] = \min\{A^3[1,3], A^3[1,4] + A^3[4,3]\}$
$= \min\{3, 4+5\}$
$= 3$

$A^4[2,1] = \min\{A^3[2,1], A^3[2,4] + A^3[4,1]\}$
$= \min\{2, 6+2\}$
$= 2$

$A^4[2,3] = \min\{A^3[2,3], A^3[2,4] + A^3[4,3]\}$
$= \min\{5, 6+5\}$
$= 5$

$A^4[3,1] = \min\{A^3[3,1], A^3[3,4] + A^3[4,1]\}$
$= \min\{9, 1+2\}$
$= 3$

$A^4[3,2] = \min\{A^3[3,2], A^3[3,4] + A^3[4,2]\}$
$= \min\{7, 1+0\}$
$= 1$

∴ The All pairs Shortest Path of the Graph is

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 3 & 1 & 0 & 1 \\ 2 & 0 & 5 & 0 \end{bmatrix} \end{array} \quad \text{or} \quad \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 4 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 3 & 1 & 0 & 1 \\ 2 & 0 & 5 & 0 \end{bmatrix} \end{array}$$

OR

Q.8	a.	Applying dynamic programming, find the optimal solution for the instance given below, with knapsack capacity W = 5.	10	L3	CO3

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

	b.	Why we need Bellman-Ford algorithm? Apply the same to find the shortest path for the graph.	10	L3	CO3

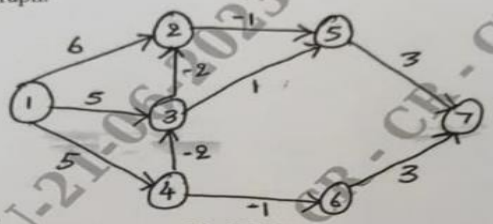


Fig.Q8(b)

A)

Weight of bag → j

i ↓ j →

Bag weight = 5

p	w	V	0	1	2	3	4	5
		0	0	0	0	0	0	0
12	2	1	0	0	12	12	12	12
10	1	2	0	10	12	22	22	22
20	3	3	0	10	12	22	30	32
15	2	4	0	10	15	22	30	(37)

Fill the first row and column with 0s, As we know that $V[0,j] = 0$ & $V[i,0] = 0$

Now for remaining values perform computation based on relation.

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

* $i = 1$:- $i = 1$ & $w_1 = 2$ profit $= 12$

* $V[1,1] = V[i-1, j] = V[0,1] = 0$ // $\because j - w_i = 1 - 2 < 0$

* $V[1,2] = V[i-1, j]$

$= \max\{V[i-1, j], v_i + V[i-1, j-w_i]\}$

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

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

$= \max\{0, 12 + 0\}$

$= 12$

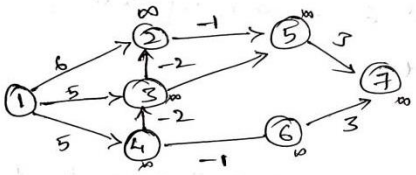
By continuous solving this using the relation, we can fill the entire table.

∴ By observing the table, the solution is $\{1, 3, 4\}$ with the max profit 37.

B)

The Bellman-Ford algorithm emulates the shortest paths from a single source vertex to all other vertices in a weighted digraph.
It is slower than Dijkstra's algorithm for the same problem but more versatile because it can handle graphs with some edge weights that are negative numbers.

The give graph,



	1	2	3	4	5	6	7

1

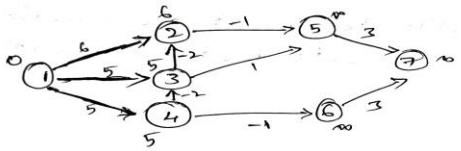
1	2	3	4	5	6	7

edge →	1	2	3	4	5	6	7
d	0	∞	∞	∞	∞	∞	∞
π	/	/	/	/	/	/	/

using vertex 1 as the source, we initialise all other distances to ∞.

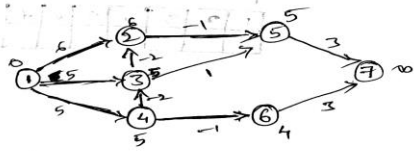
* Iteration 1:- Edges 2 and 3 & 4 relax updating the distance 6

* Iteration 1 :- Edges 2, 3 and 4 relax updating the distance 6, 5 and 5 respectively



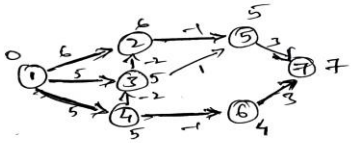
	1	2	3	4	5	6	7
d	0	6	5	5	∞	∞	∞
π	/	1	1	1	/	/	/

* Iteration 2:- edges 5 and 6 relax updating the distance 5 and 4.



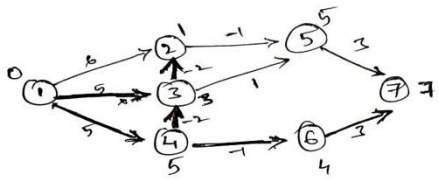
	1	2	3	4	5	6	7
d	0	6	5	5	5	4	∞
π	/	1	1	1	2	4	/

* Iteration 3:- Edge 7 relax updating the distance 7.



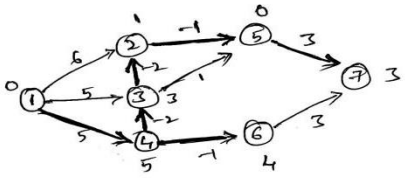
	1	2	3	4	5	6	7
d	0	6	5	5	5	4	7
π	/	1	1	1	2	4	6

Iteration 4:- Edges (4,3) & (3,2) relax updating distance 3 & 1



	1	2	3	4	5	6	7
d	0	1	3	5	5	4	7
π	1	3	4	1	2	4	6

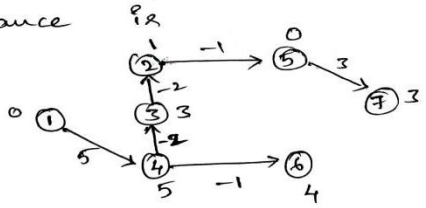
Iteration 5 :- Edges (2,5) and (5,7) relax updating distance 0 and 3.



	1	2	3	4	5	6	7
d	0	1	3	5	0	4	3
π	1	3	4	1	2	5	5

Iteration 6 :- No edges relax.

→ The final shortest path from vertex 1 with corresponding distance is

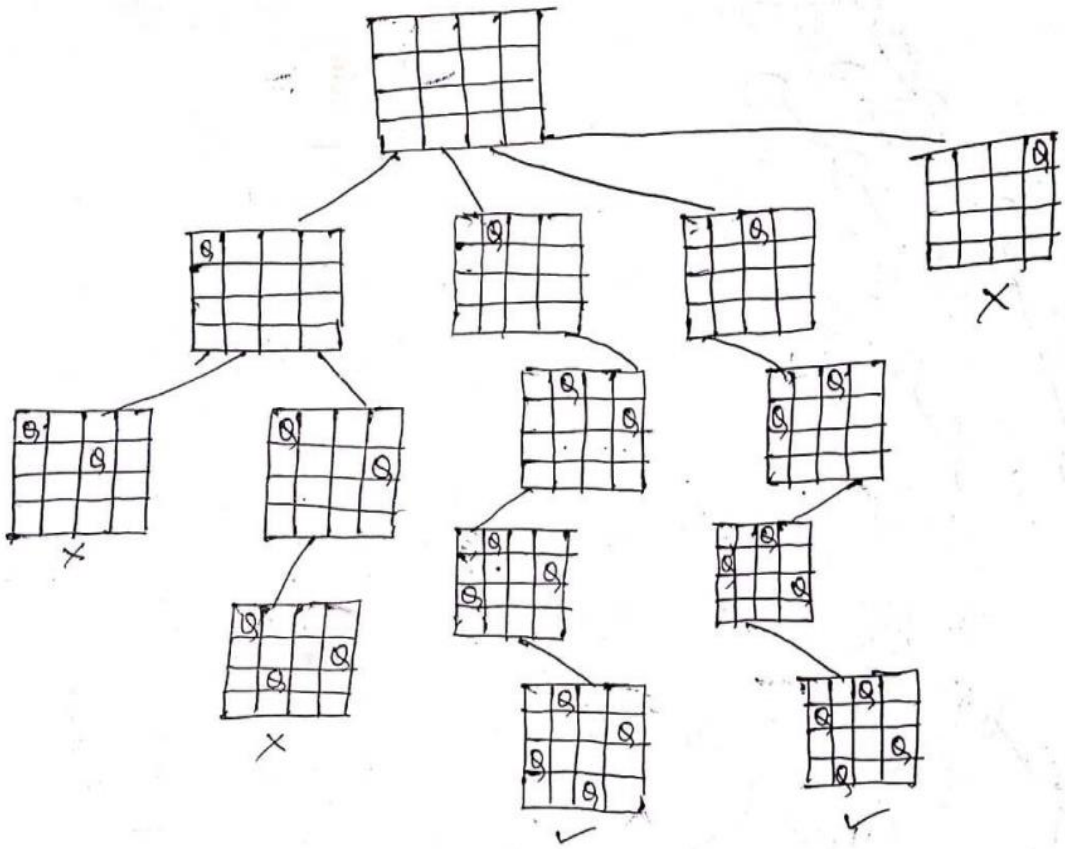


		Module – 5			
Q.9	a.	Define Backtracking. Find the solution space tree for 4-queens problem.	10	L2	CO1
	b.	Find the subset from the given set with d = 15, s = {3, 7, 5, 6} by constructing a state space tree.	10	L3	CO2

A)

Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.

The solution space tree for 4 queens problem



(4-queens problem)

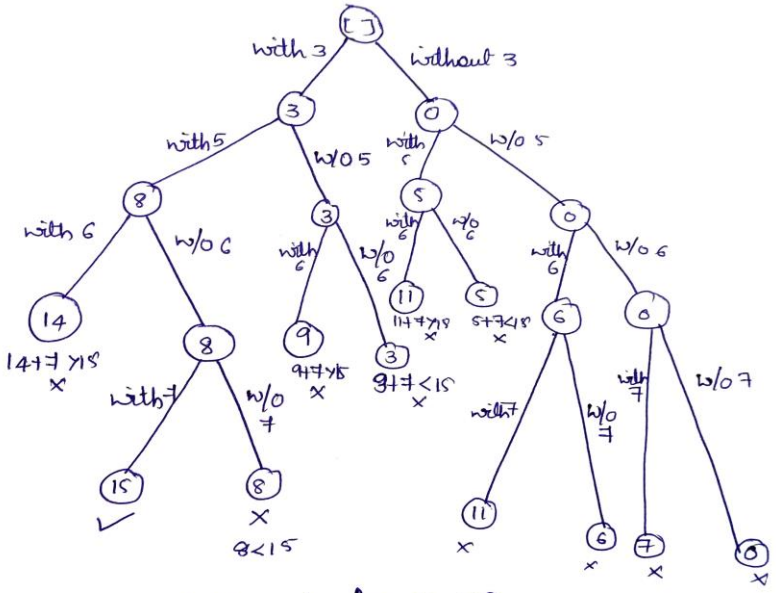
B)

→ • Firstly check the elements in the given set S it must be in sorted manner,

  * Value of first element in S=3 must be less than d
  * Sum of all elements in s must be greater than d

• Start constructing the state-space tree.

• Sorted, S={3,5,6,7}



Q.10							
	a.	Solve the assignment problem using branch and bound technique.				10	L3	CO3

	d_1	d_2	d_3	d_4
p_1	9	2	7	8
p_2	6	4	3	7
p_3	5	8	1	8
p_4	7	6	9	4

b.	Solve the following travelling salesman problem using branch and bound technique by using the constraint "Visit City "b" before "c" ".	10	L3	CO3

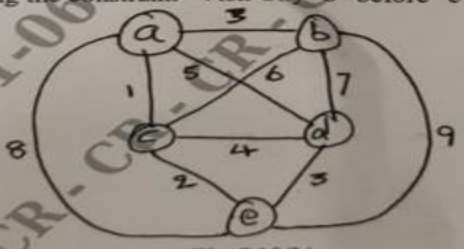
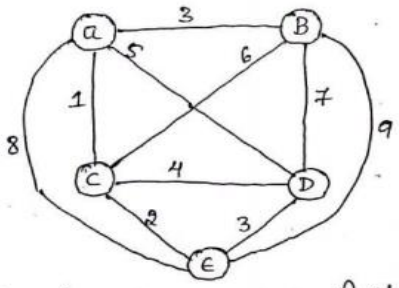


Fig.Q10(b)

A)



Src = A

⇒ Compute the Lower Bound (LB), Initially by choosing Min cost Edge from Each vertex & total Sum should be divided by 2

$$LB = [\underbrace{(1+3)}_{A} + \underbrace{(3+6)}_{B} + \underbrace{(1+2)}_{C} + \underbrace{(4+3)}_{D} + \underbrace{(2+3)}_{E}]/2 = 28/2 = 14$$

→ Next, Make calculated Lower Bound (LB) as root node & Start check -ing all possibilities.

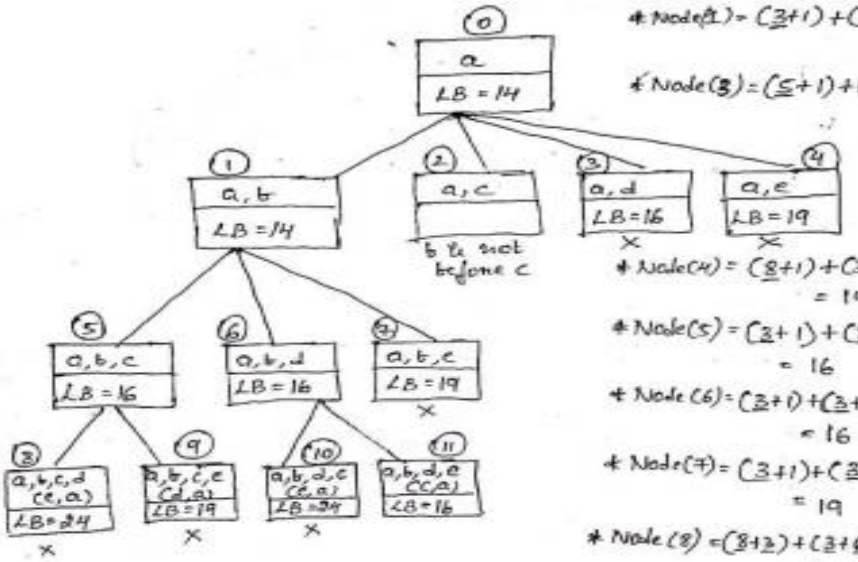
While checking the lower Bound (LB) of paths, keep the value of path, that you're considered same & for remaining vertices, you can choose Min cost Edges.

Ex: Consider path "ABC", Here the cost of vertices are

$\underbrace{(AB + min)}_{A} + \underbrace{(AB + AC)}_{B} + \underbrace{(BC + min)}_{C} + \underbrace{(min + min)}_{D} + \underbrace{(min + min)}_{E}$

Where "min" → min cost Edge from that vertex

→ The State-Space-tree is constructed with the calculated LowerBound = 14 & nodes are enumerated Sequentially



* Node(1) = (3+1) + (3+6) + (1+2) + (4+3) + (2+3)/2
= 14

* Node(3) = (5+1) + (3+6) + (1+2) + (5+3) + (2+3)/2
= 15.5 (16)

* Node(4) = (8+1) + (3+6) + (1+2) + (4+3) + (2+3)/2
= 19

* Node(5) = (3+1) + (3+6) + (6+1) + (4+3) + (2+3)/2
= 16

* Node(6) = (3+1) + (3+7) + (1+2) + (7+3) + (2+3)/2
= 16

* Node(7) = (3+1) + (3+9) + (1+2) + (4+3) + (9+2)/2
= 19

* Node(8) = (8+2) + (3+6) + (6+9) + (9+3) + (3+8)/2
= 24

* Node(9) = (5+3) + (3+6) + (6+2) + (2+3) + (3+5)/2
= 19

* Node(10) = (8+3) + (3+7) + (7+4) + (4+2) + (2+8)/2
= 24

* Node(11) = (1+3) + (3+7) + (7+8) + (3+2) + (2+1)/2
= 16

∴ the path is

a → b → d → e → c → a

The cost of path is 16 //

B)

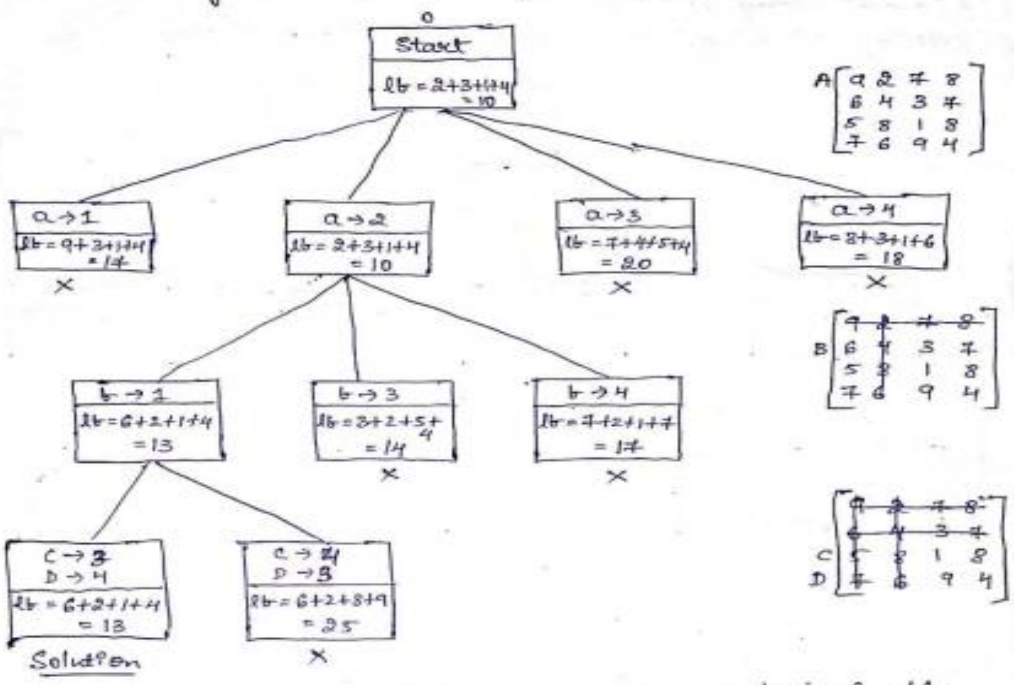
→ Consider the following Cost Matrix

Jobs / Persons	J_1	J_2	J_3	J_4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

⇒ Firstly, Compute the lower bound by adding the Smallest Elements in Every row.

* Lower Bound would be (lb) = 2 + 3 + 1 + 4 = 10 //

→ Now at Every Step, we have to keep finding Minimal Soln without Violating the Constraints of the problem



→ ∴ the cost of given Assignment problem is '13' & the Job Allocation would be

person A → Job 2
person B → Job 1
person C → Job 3
person D → Job 4 //