Fourth Semester B.E Degree Examination

Design and Analysis of Algorithms

Model Question Paper - 01

Module - 03

05

a) Apply greedy method to obtain an optimal solution to the knapsack problem where knapsack capacity $m = 15$ [7 marks]

object	1	2	3	4	5	6	7
weight	10	5	15	7	6	8	3
profit	2	3	5	7	1	4	1

=> given that

Knapsack capacity $m = 15$

Number of objects $n = 7$

weight = $(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (10, 5, 15, 7, 6, 8, 3)$

Profit = $(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (2, 3, 5, 7, 1, 4, 1)$

object	1	2	3	4	5	6	7
Profit	2	3	5	7	1	4	1
weight	10	5	15	7	6	8	3
Profit/weight (P_i/w_i)	$\frac{2}{10}$ 0.2	$\frac{3}{5}$ 0.6	$\frac{5}{15}$ 0.33	$\frac{7}{7}$ 1	$\frac{1}{6}$ 0.166	$\frac{4}{8}$ 0.5	$\frac{1}{3}$ 0.33
x	0	1	0	1	0	3/8 = 0.375	0

= 0.375

where x is a fraction of object taken into knapsack.

- Greedy add objects to knapsack while the weight does not exceed capacity.
- At each step fill the knapsack with the object with largest profit.
- object 2 and object 4 are included in the knapsack and this adds up to the weight 5 + 7 = 12.
- The next object we considerd is object 6 which has the weight of 15 but if we include object 6 entirely the sum of weight will be 12 + 8 = 20 which is greater than 15 & it exceeds the capacity of the knapsack.

- we have only 3 weight units left in the knapsack.
- The fraction 3/8 of object 6 is included in the knapsack to fill the knapsack capacity.

The total profit obtained is

$= (1 \times 3) + (1 \times 7) + (0.375 \times 4)$

$= 3 + 7 + 1.5$

$= 11.5$

∴ maximum profit = 11.5 units

05

b) What is job sequencing with deadlines problem? for the given data find the optimal job sequence and maximum profit using greedy approach [6 marks]

Jobs	J_1	J_2	J_3	J_4	J_5
Profits	60	100	20	40	20
Deadlines	2	2	3	1	1

=> Job sequencing with deadline problem

definition :- given a set of n jobs and an integer is associated with job i, deadline $d_i >= 0$ and profit $P_i >= 0$. it is required to find the jobs such that all the chosen jobs should be completed within their deadline and profit earned should be maximum with the following constraints

- only one machine is available for processing jobs.

- only one job must be processed at any point of time.

Given that

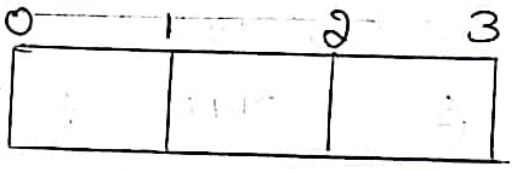
Jobs	J_1	J_2	J_3	J_4	J_5
Profits	60	100	20	40	20
Deadlines	2	2	3	1	1

Step 01: Sort all the given jobs in decreasing order of their profits.

Jobs	J_2	J_1	J_4	J_3	J_5
Deadlines	2	2	1	3	1
profits	100	60	40	20	20

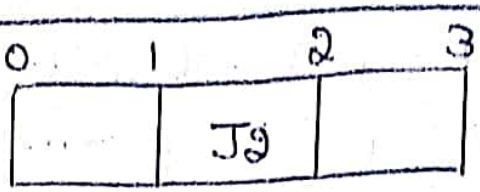
Step 02: Value of maximum deadline = 3

- draw a gantt chart with maximum time on gantt chart = 3 units



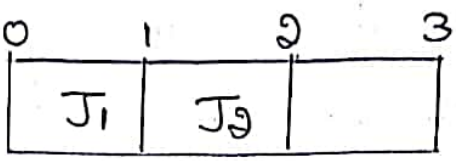
Step 03:

- we take job J_2
- since its deadline is 2, so we place it in the first empty cell before deadline 2.



Step 04:-

- we take job J_1.
- since its deadline is 2, we place it in the first empty cell before deadline 2 as

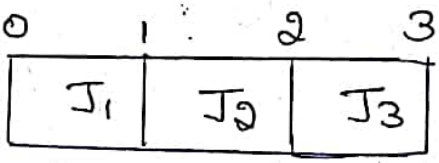


Step 05:-

- we take job J_4 whose deadline is 1.
- But all the slots before deadline 1 are already occupied thus, job J_4 can not be completed.

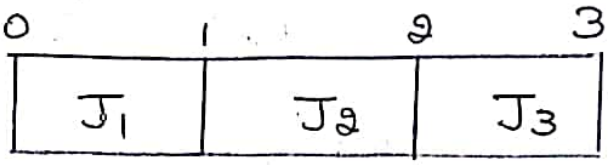
Step 06:-

- we take job J_3
- since its deadline is 3, we place it in the first empty cell before deadline 3 as



Now

- The only job left is job J_5 whose deadline is 1.
- All the slots before deadline 1 are already occupied thus, job J_5 can not be completed.



The optimal schedule is J_1, J_2, J_3 which are being executed within their deadline and gives maximum profit.

Maximum earned profit = Sum of profit of all the jobs in optimal schedule

= profit of J_1 + profit of J_2 + profit of J_3

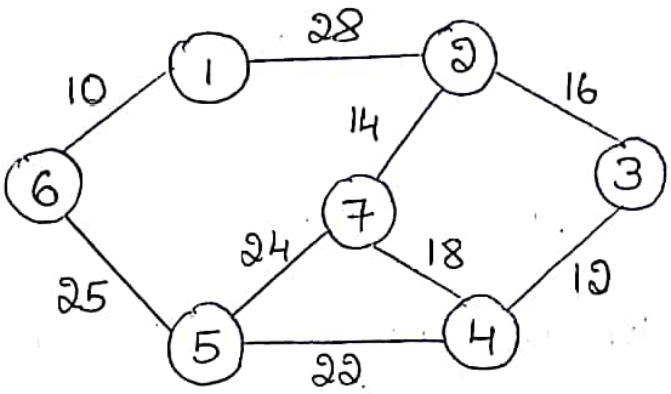
= 100 + 60 + 20

= 180

∴ Total profit of this sequence = 180 units

05

c. Apply prim's algorithm to obtain the minimum cost spanning tree for the given weighted graph [7 marks]



⇒ Prim's Algorithm :-

* prim's algorithm is famous greedy algorithm.
* It is used for finding the minimum spanning Tree (MST) of a given graph.
* To apply prim's algorithm, the given graph must be weighted, connected and undirected.

By applying prim's algorithm we can find minimum cost spanning tree by following below steps

Step 01 :-

- Randomly choose any vertex.
- The vertex connecting to the edge having least weight is usually selected.

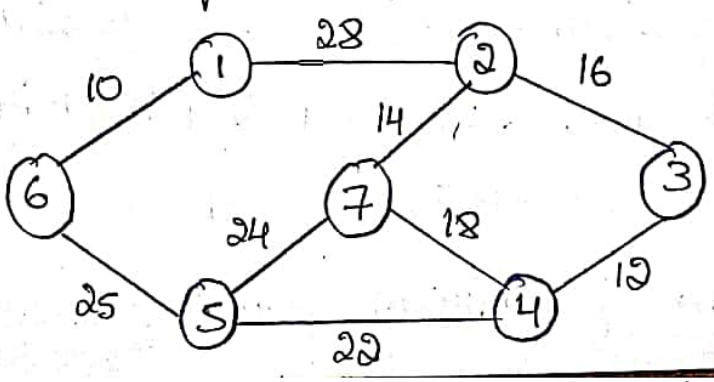
Step 02:-

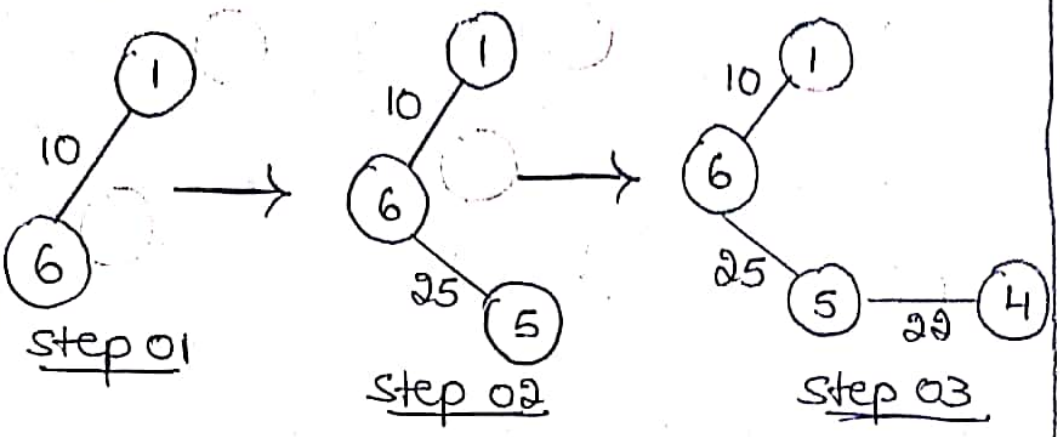
- Find all the edges that connect the tree to new vertices.
- Find the least weight edge among those edges and include it in the existing tree.
- If including that edge creates a cycle then reject that edge and look for the next least weight edge.

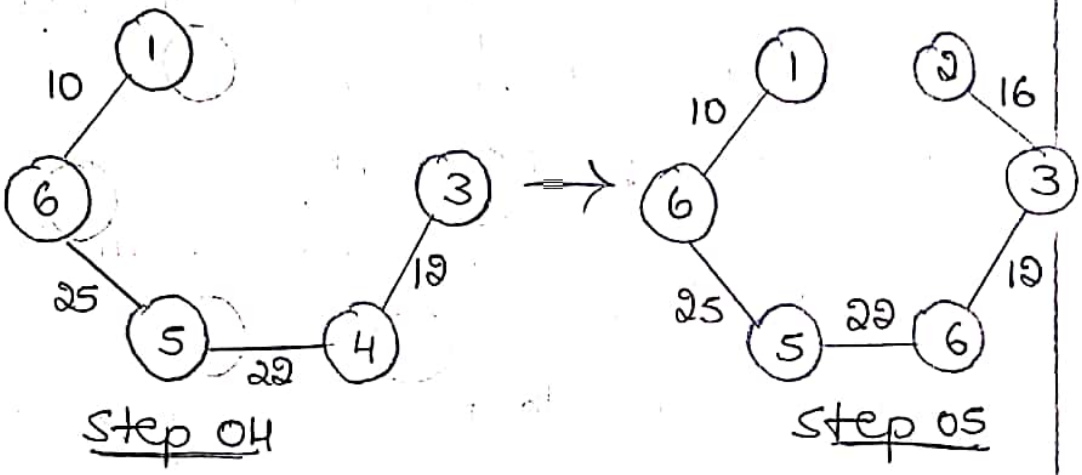
Step 03:-

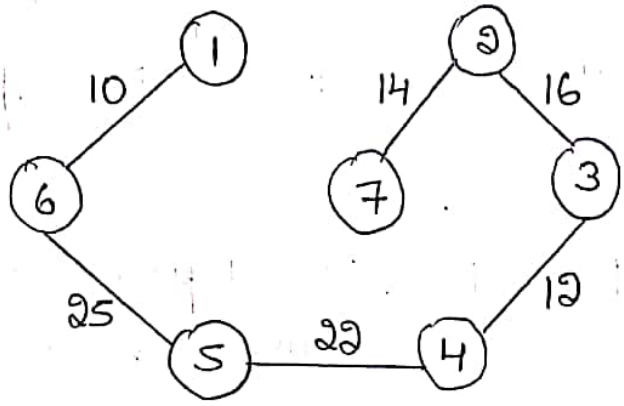
- Keep repeating step-02 until all the vertices are included and minimum spanning tree (MST) is obtained.

Consider a given graph









Now, Cost of minimum spanning Tree

= Sum of all edge weights

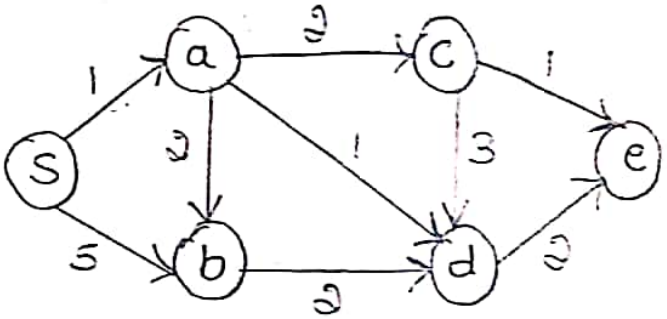
= 10 + 25 + 22 + 12 + 16 + 14

= 99 units

∴ Cost of minimum spanning tree = 99 units

06.

a) Design Dijkstra's algorithm and apply the same to find single source shortest path for the given graph by considering 's' as the source vertex. [8 marks]



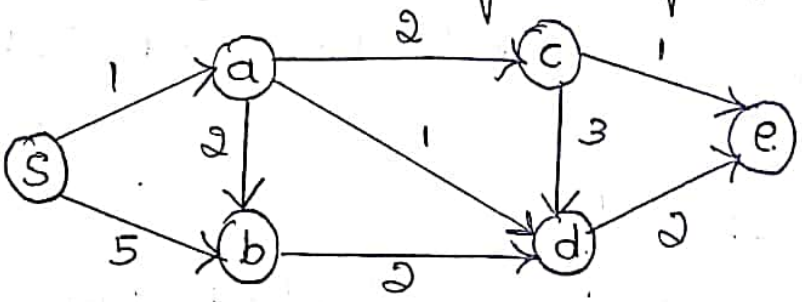
⇒ Dijkstra's Algorithm :-

- Djkstra's algorithm is a very famous greedy algorithm
- It is used for solving the single source shortest path problem.
- It computes the shortest path from one particular source node to all other remaining nodes of the graph.

```
Algorithm ShortestPaths(v, cost, dist, n)
// dist[j], 1<=j<=n, is set to the length
  of the shortest
// path from vertex v to vertex j in
  a digraph G with n vertices.
// vertices, dist[v] is set to zero. G
  is represented by its
// cost adjacency matrix cost[1:n,1:n]
{
   for i:=1 to n do
    {
      // initialize S.
      S[i] := false;
      dist[i] := cost[v,i];
    }
    S[v] := true;
    dist[v] := 0.0; //put v in S
    for num := 2 to n-1 do
    {
      //Determine n-1 paths from v
       Choose u from among those
       vertices not in S such that
       dist[u] is minimum;
```

```
S[u] := true ; // put u in S
for(each W adjacent to u with
        S[W] = false) do
// update distances
if(dist[W] > dist[u] + cost[u,W]) then
    dist[W] := dist[u] + cost[u,W];
}
}
```

Now consider the given graph



Step 01:- The following two sets are considered

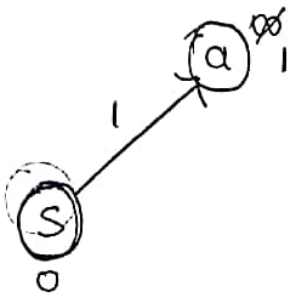
- Unvisited set = { s, a, b, c, d, e }
- Visited set = { }

Step 02 :- The two variables π and d are created for each vertex and initialized as

- $\pi[s] = \pi[a] = \pi[b] = \pi[c] = \pi[d] = \pi[e] = NIL$
- $d[s] = 0$

- $d[a] = d[b] = d[c] = d[d] = d[e] = \infty$
- $\pi[v]$ denotes the predecessor of vertex 'v' and $d[v]$ denotes the shortest path estimated from source vertex to other vertices.

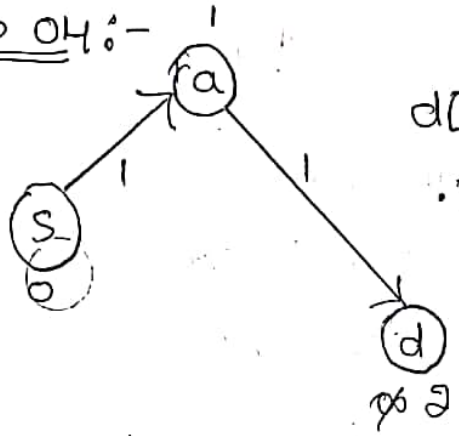
Step 03 :- Consider vertex s source vertex.



$d[s] + 1 = 0 + 1 = 1 < \infty$

$\therefore d[a] = 1$ and $\pi[a] = s$

Step 04 :-

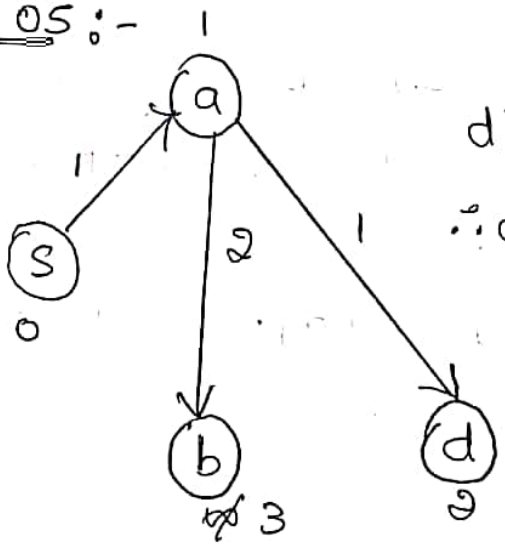


$d[a] + 1 = 1 + 1 = 2 < \infty$

$\therefore d[d] = 2$ and

$\pi[d] = a$

Step 05 :-

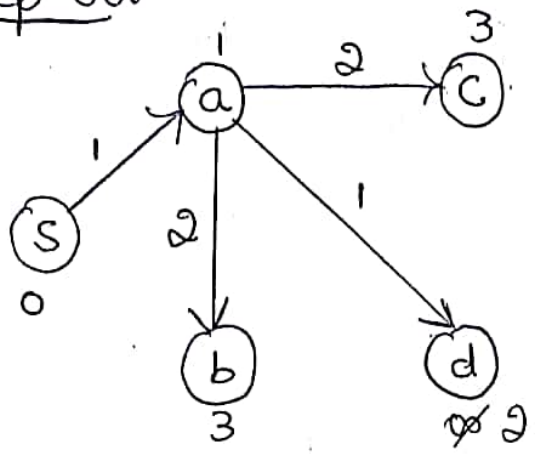


$d[a] + 2 = 1 + 2 = 3 < \infty$

$\therefore d[b] = 3$ and

$\pi[b] = a$

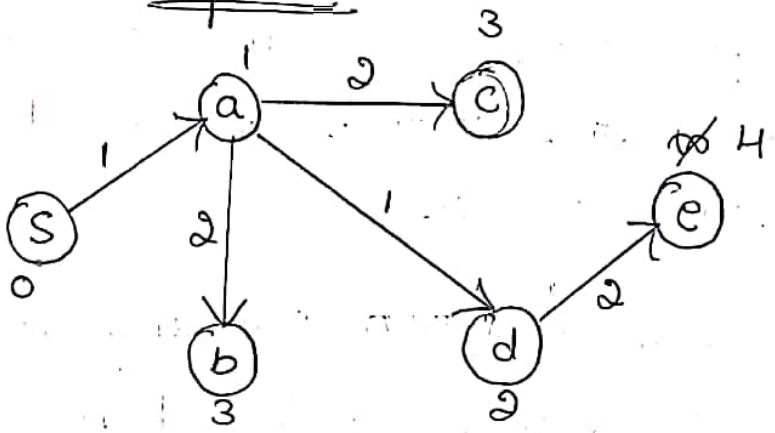
Step 06:-



$d[a] + 2 = 1 + 2 = 3 < \infty$

$\therefore d[c] = 3$ and $\Pi[c] = a$

Step 07:-



$d[d] + 2 = 2 + 2 = 4 < \infty$

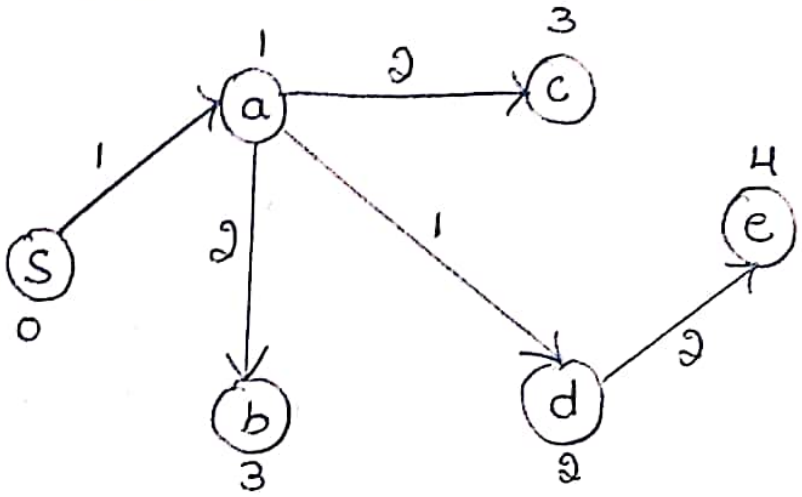
$\therefore d[e] = 4$ and $\Pi[e] = d$

By considering all the steps, now the sets are updated as

- Unvisited set = { }
- Visited set = {s, a, b, c, d, e}

Now,

- All vertices of the graph are processed
- The final shortest path tree is as shown below



- It represents the shortest path from source vertex 'S' to all other remaining vertices.

The order in which all the vertices are processed is S, a, d, b, C, e

06

b. Construct the Huffman tree for the following data [5 marks]

Character	A	B	C	D	E	-
probability	0.5	0.35	0.5	0.1	0.4	0.2

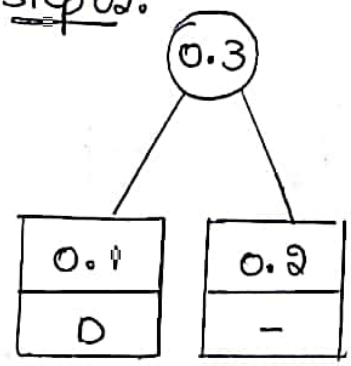
Encode: a) BED b) AB_CD

⇒ Huffman tree construction

Step 01: arrange the probabilities in decreasing order

0.1	0.2	0.35	0.4	0.5	0.5
D	-	B	E	A	C

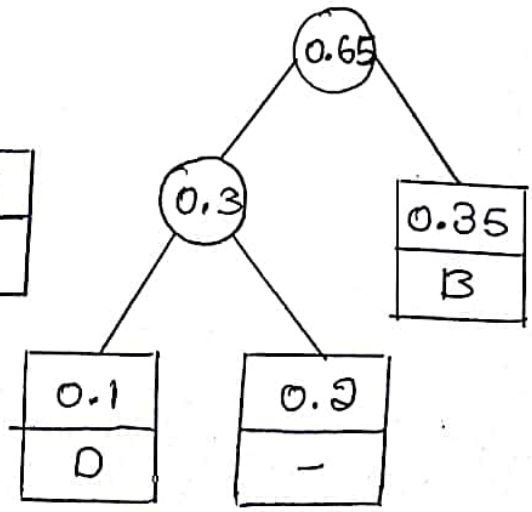
Step 02:



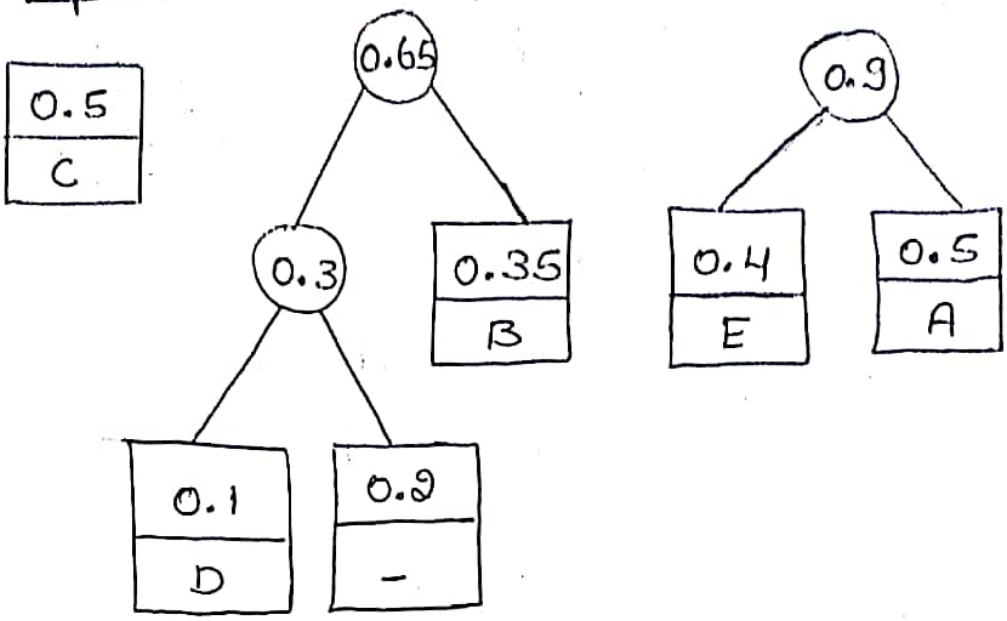
0.35	0.4	0.5	0.5
B	E	A	C

Step 03:

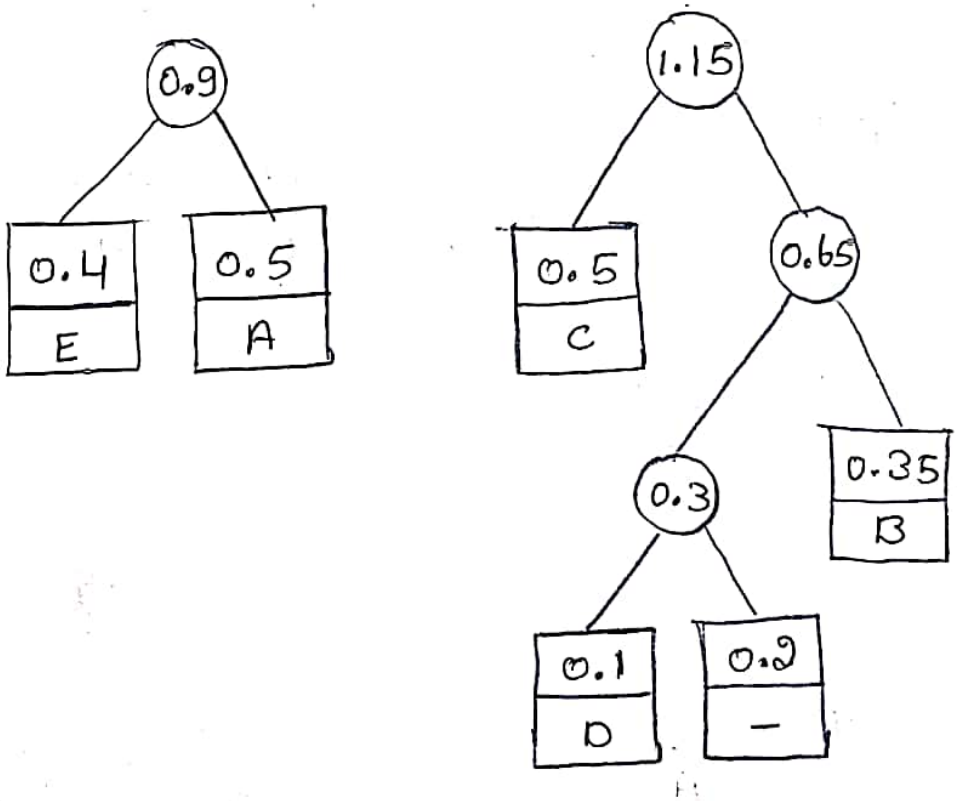
0.4	0.5	0.5
E	A	C

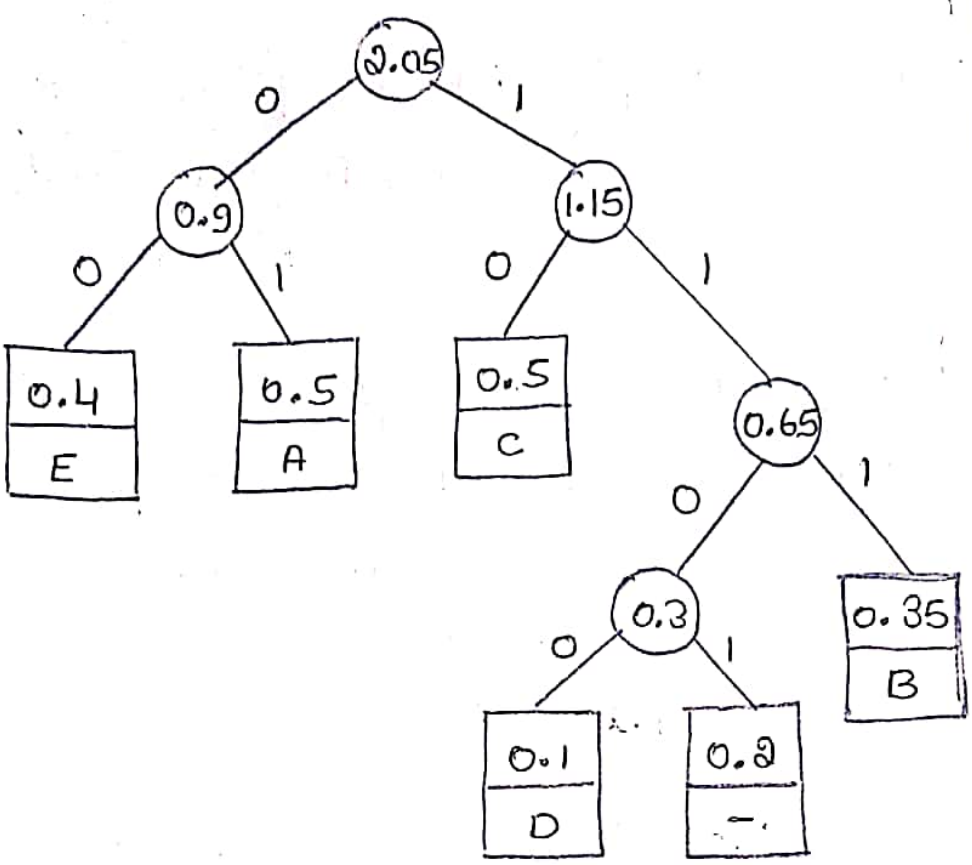


Step 04 :-



Step 05 :-





The resulting codewords are as yellow

Symbol	A	B	C	D	E	-
Probability	0.5	0.35	0.5	0.1	0.4	0.2
Codeword	01	111	10	1100	00	1101

hence

BED is encoded as 111001100

AB_CD is encoded as 011111101101100

06

C. Define Heap. Sort the given list of elements using heap sort : 2, 9, 7, 6, 5, 8

[8 marks]

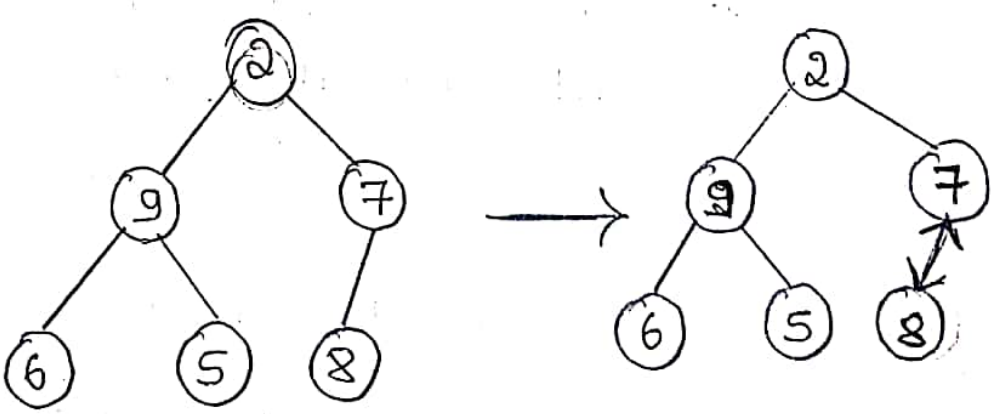
=> Defination :- A heap is a complete binary tree with the property that the value at each node is at least as large as the values at its children.

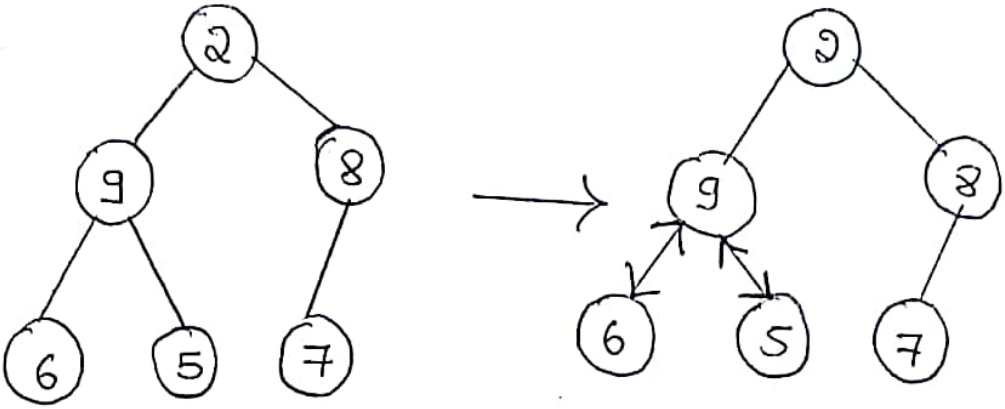
Consider the given elements

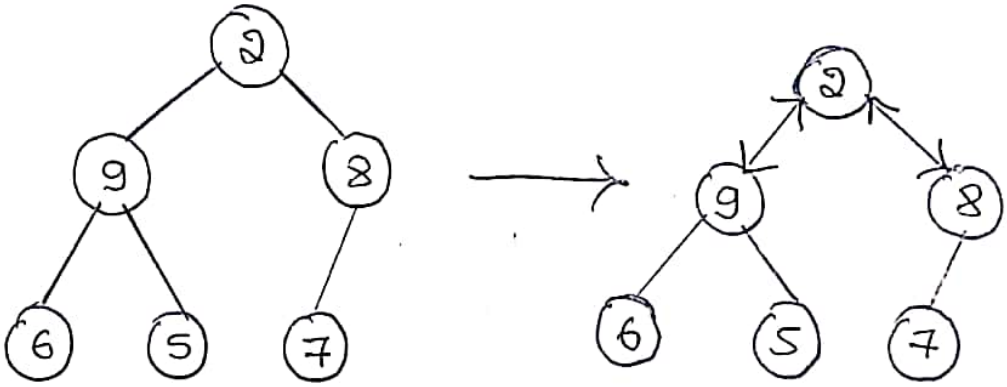
2, 9, 7, 6, 5, 8

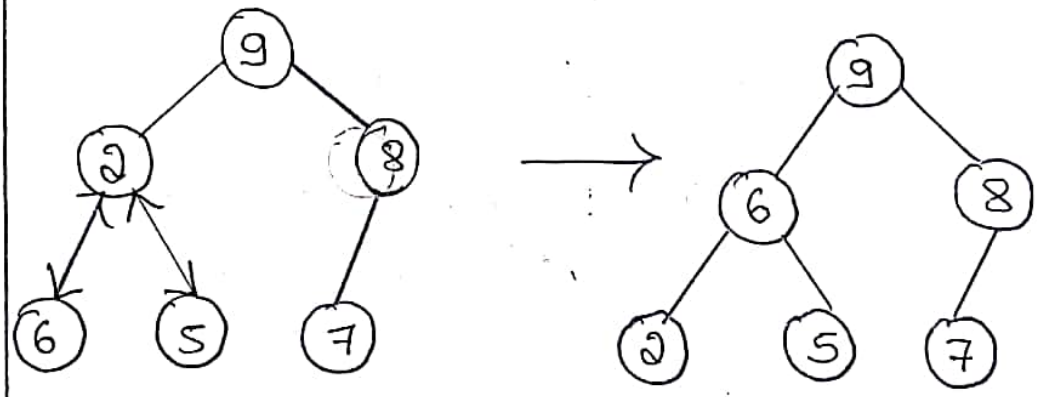
Stage 1 :- (Heap Construction)

Construct a heap for a given array









Step 1 : 2 9 7 6 5 8

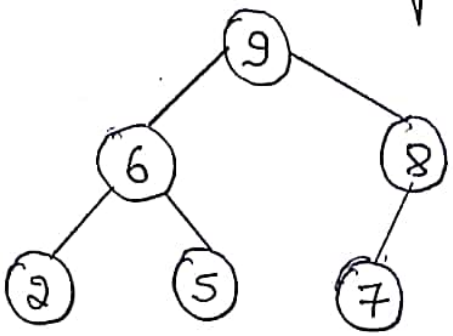
step 2 : 2 9 8 6 5 7

step 3 : 2 9 8 6 5 7

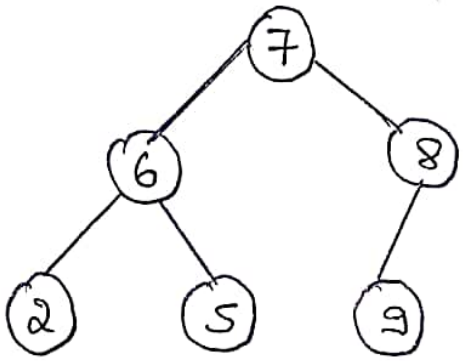
step 4 : 9 2 8 6 5 7

step 5 : 9 6 8 2 5 7

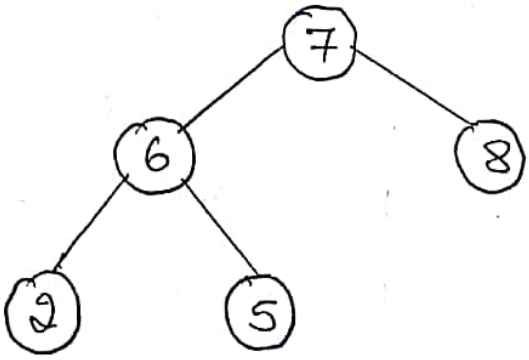
Stage 2 (maximum deletion)
apply the root - deletion operation n-1 times to the remaining heap.



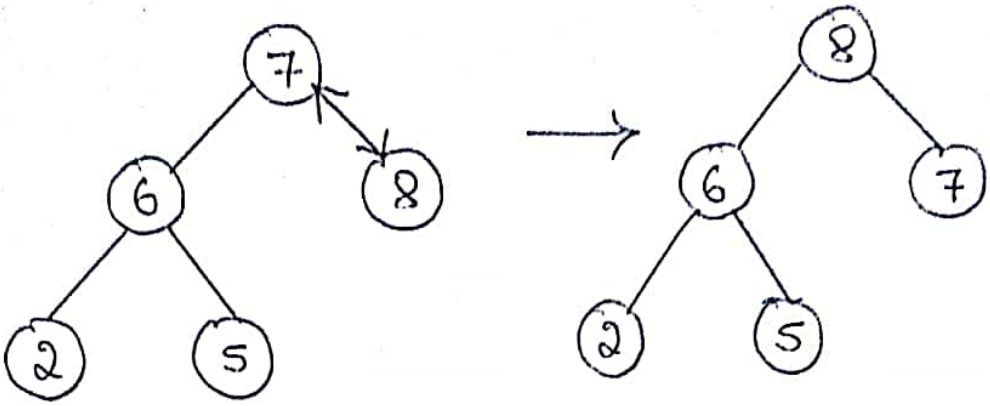
Step 01 : Replace the root node with the leaf node



Step 02 : Delete the last leaf node i.e node 9 is deleted.



Step 03: Compare the nodes and sort the elements



Final sorted array is 2, 5, 6, 7, 8

Therefore we have deleted root key from heap. the key to be deleted is swapped with the last key. After which the smaller tree is "heapified" by exchanging the new key in its root with the larger key in its children until the parental dominance requirment is satisfied.

Model Question Paper - 2 with effect from 2019 - 20 (CBCS Scheme)

Fourth Semester B.E. Degree Examination

Design and Analysis of Algorithms

Module - 3

5 @ Solve the following instance of greedy knapsack problem where $n = 4$, $m = 10$, $p = (40, 42, 25, 12)$ and $w = (4, 7, 5, 3)$ [6 marks]

→ Given that:

Knapsack capacity $m = 10$

number of objects $n = 4$

Profit $= (P_1, P_2, P_3, P_4) = (40, 42, 25, 12)$

Weight $= (w_1, w_2, w_3, w_4) = (4, 7, 5, 3)$

object	1	2	3	4
Profit	40	42	25	12
Weight	4	7	5	3

Profit / Weight (P_i/W_i)	$\frac{40}{4}$ = 10	$\frac{42}{7}$ = 6	$\frac{25}{5}$ = 5	$\frac{12}{3}$ = 4
x	1	$\frac{6}{7}$ = 0.8571	0	0

where x is a fraction of object taken into knapsack.

* Object 1 is included in knapsack which has capacity of 10, remaining knapsack capacity = 10 - weight of object 1
  10 - 4 = 6

* The next object we considered is objt2 which has weight of 7, but if we include object 2 completely the sum of weights will be 4+7 = 11 > 10, it exceeds capacity of knapsack which is only 10.

* We have only 6 weight units left in knapsack, the fraction $\frac{6}{7}$ object 2 is included to fill its capacity.

∴ The profit obtained is,

= 1 × 40 + 0.8571 × 42

= 175.9982.

∴ The total profit earned = 76 units

5 (b) Write the problem statement for job sequencing with deadline? Let $n = 5$, profits (10, 3, 33, 11, 40) and deadlines (3, 1, 1, 2, 2). Find the optimal sequence of execution of job solution using greedy algorithm. [6 M]

→ The sequencing of jobs on a single processor with deadline constraints is called as Job sequencing with deadlines.

Given a set of n jobs and an integer d_i is associated with job i, deadline $d_i >= 0$ and profit $p_i >= 0$. It is required to find jobs such that chosen job should be completed within their deadline and profit earned should be maximum.

Given that :

number of jobs, $n = 5$

profits = [10, 3, 33, 11, 40]

deadlines = [3, 1, 1, 2, 2]

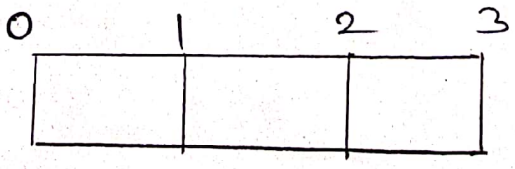
Jobs	J_1	J_2	J_3	J_4	J_5
Deadlines	3	1	1	2	2
Profits	10	3	33	11	40

Step 1 : Sort all the given jobs in decreasing order of their profits.

Jobs	J_5	J_3	J_4	J_1	J_2
Deadlines	2	1	2	3	1
Profits	40	33	11	10	3

Step 2 : * Value of maximum deadline = 3

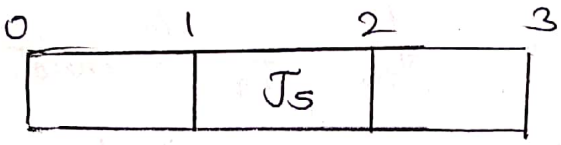
* So, draw a gantt chart with maximum time on gantt chart = 3 units.



gantt chart

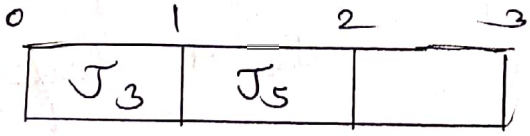
Step 3 :

* We take job J_5.
* Since its deadline is 2, so we place it in the first empty cell before deadline 2 as,



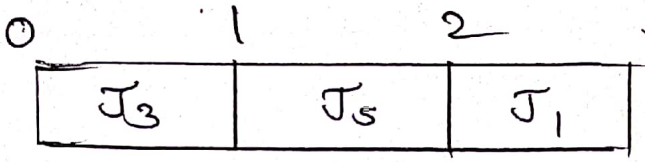
Step 4 :

* We take job J_3
* Since its deadline is 1, so we place it in first empty cell before deadline 1 as,



Step 5 : * We take job J_4 where deadline is 2 but all the slots before deadline 2 are already occupied.
* Thus, J_4 cannot be filled.

Step 6 : * We take Job J_1
* Since its deadline is 3, so we place it in first emptycell before deadline 3 as



* The only job left is Job J_2 where deadline is 1 but all the slot before deadline 1 are already occupied.
* Thus, Job J_2 cannot be filled.

The optimal schedule is J_3, J_5, J_1.

Maximum earned profit = Sum of profit of all jobs in optimal schedule.

Maximum profit = Profit of J_3 + Profit of J_5 + Profit of J_1

= 33 + 40 + 10

= 83 units.

∴ Maximum earned profit = 83 units.

5) (c) Define minimum cost spanning tree. Write Prim's algorithm to find minimum cost spanning tree. [8 marks]

→ **Minimum cost spanning tree:**

A spanning tree of an undirected connected graph is its connected acyclic subgraph that contains all the vertices of the graph.

If such graph has weights assigned to its edges, a minimum spanning tree is its spanning tree of smallest weight, where the weight of a tree is defined as sum of weights on all its edges.

Prim's algorithm:

Algorithm Prim (G)

// Prim's algorithm for constructing a minimum spanning tree.

// Input : A weighted connected graph

$G = V, E$

// Output : E_T, the set of edges composing a minimum spanning tree of G.

// $V_T \leftarrow \{v_0\}$ the set of tree vertices can be initialized with any vertex.

$E_T \leftarrow \phi$

for $i \leftarrow 1$ to $|V| - 1$ do

find a minimum - weight edge $e^* = (v^*, u^*)$ among all the edges (v, u) such that v is in V_T and u is in $V - V_T$.

$V_T \leftarrow V_T \cup \{u^*\}$ $\quad E_T \leftarrow E_T \cup \{e^*\}$

return E_T.

(or)

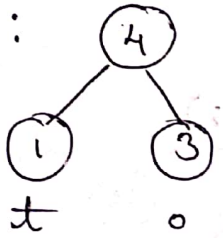
6) a) Obtain the Huffman tree and the code for following data. [4 marks]

Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

→ Arrange the frequencies in ascending order.

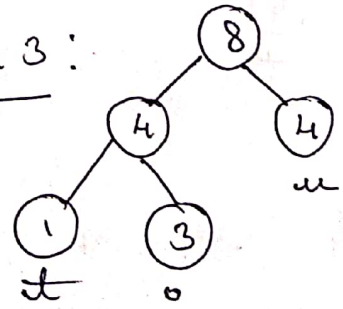
Step 1: 1 3 4 10 12 13 15
t o u a i s e

Step 2:



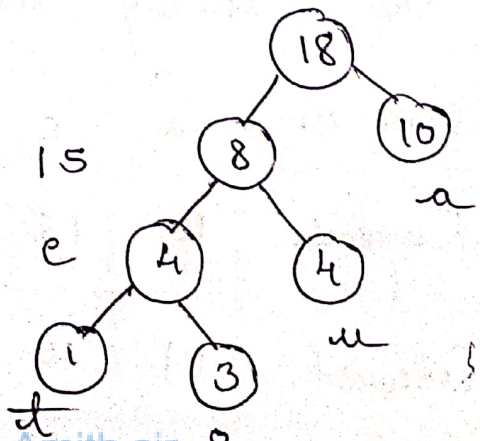
4 10 12 13 15
u a i s e

Step 3:

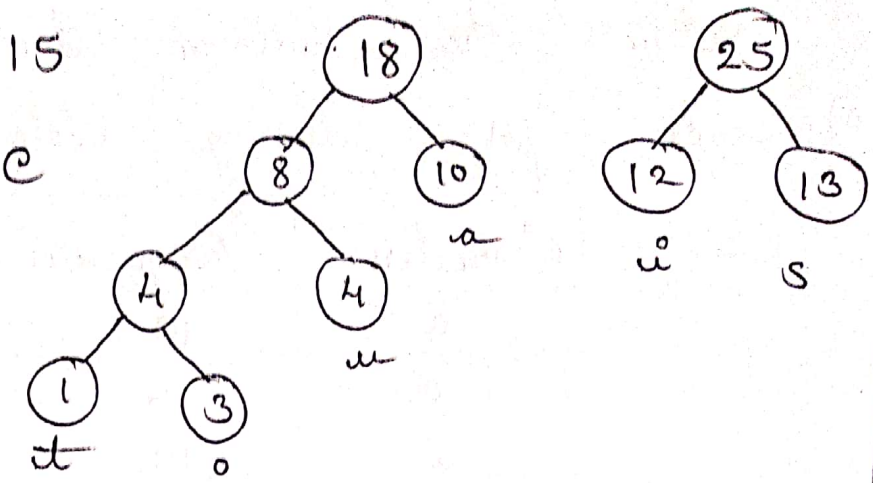


10 12 13 15
a i s e

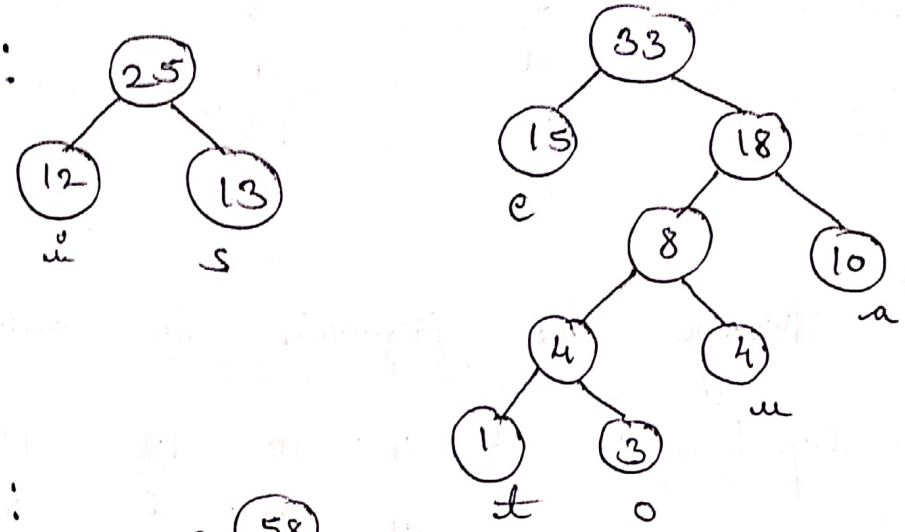
Step 4: 12 13 15
i s e



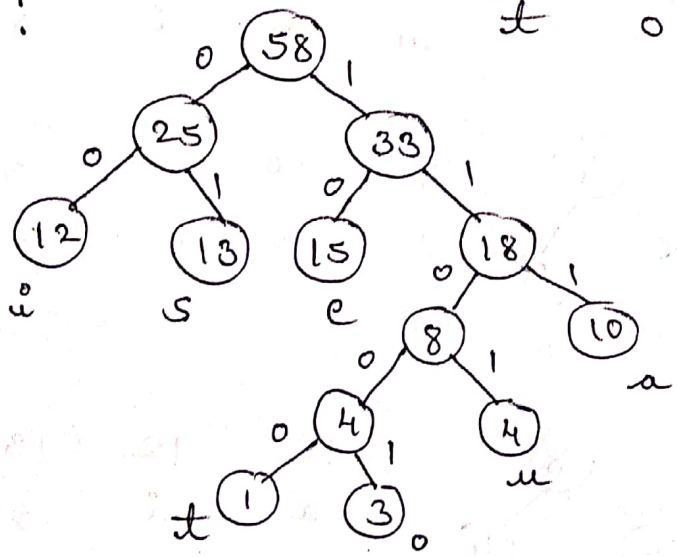
Step 5 :



Step 6 :



Step 7 :



The resulting codewords are as follows,

Characters	a	e	i	o	m	s	t
Frequencies	10	15	12	3	4	13	1
Codeword	111	10	00	11001	1101	01	11000

6) b) Write an algorithm to find single source shortest path for a graph G whose edge weights are positive. [8 marks]

→ Dijkstra's Algorithm is used for finding single source shortest path problem.

* Dijkstra Algorithm is a very famous greedy algorithm.

* It computes shortest path from one particular source node to all other remaining nodes of the graph.

Algorithm:

Algorithm Dijkstra (G, S)

// dist[j], 1 <= j <= n, is set to the length of the shortest.

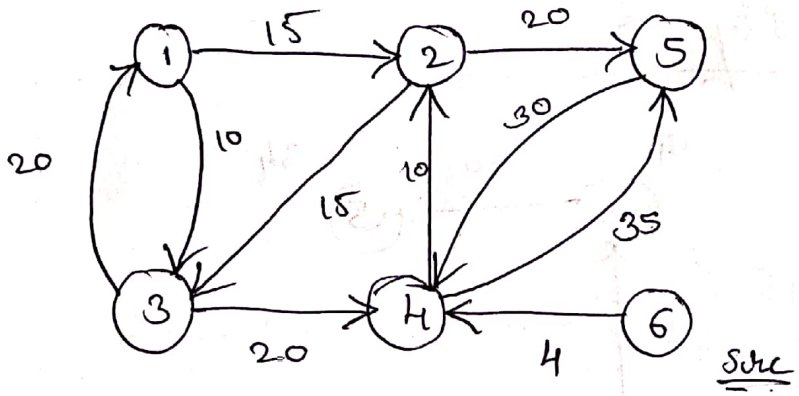
```
// path from vertex v to vertex j
in a diagraph G with n.
// vertices, dist[v] is set to zero.
// G is represented by its cost
matrix cost[1:n, 1:n]
{
  for i:= 1 to n do
  {
      // Initialize S
      s[i]:= false; dist[i]:= cost[v,i];
  }
  s[v]:= true; dist[v]:= 0.0;
                          // put v in S.
  for num:= 2 to n-1 do
  {
    // Determine n-1 paths from v.
    Choose u from among those vertices
not in S such that dist[u] is minimum;
    s[u]:= true; // put u in S.
    for (each w adjacent to u with
                          s[w] = false) do
    // Update distances.
    if (dist[w] > dist[u] + cost[u,w]))
                                    then
```

dist [w] := dist [u] + cost [u, w];

}

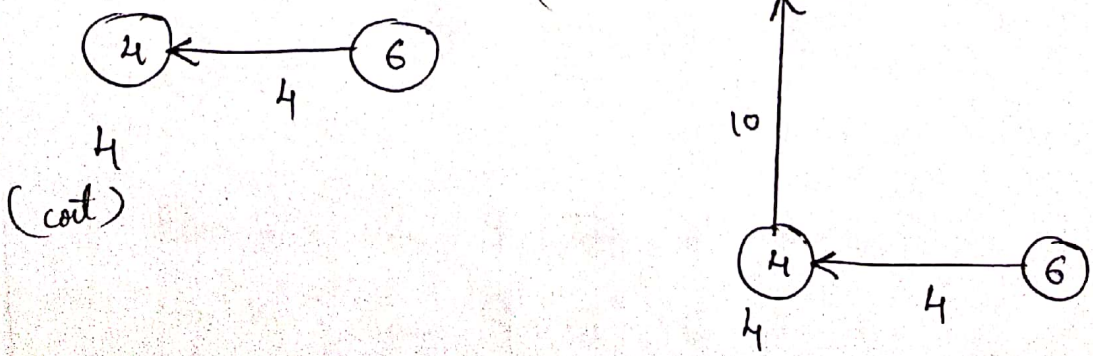
}

For example ;

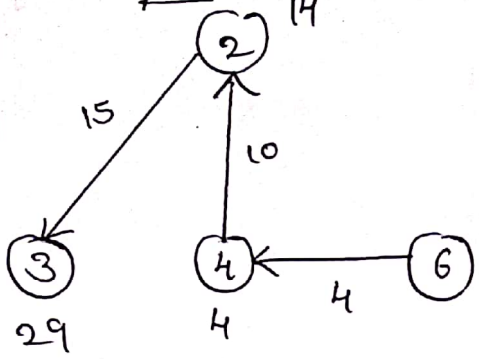


Let us solve the above example by considering 6 as source vertex.

Step - 1 → Step - 2

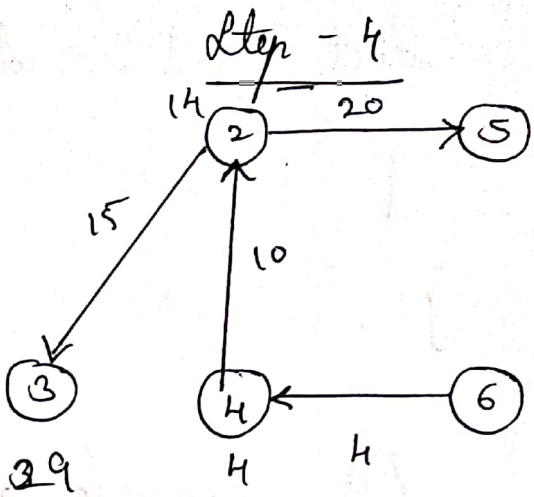


Step - 3

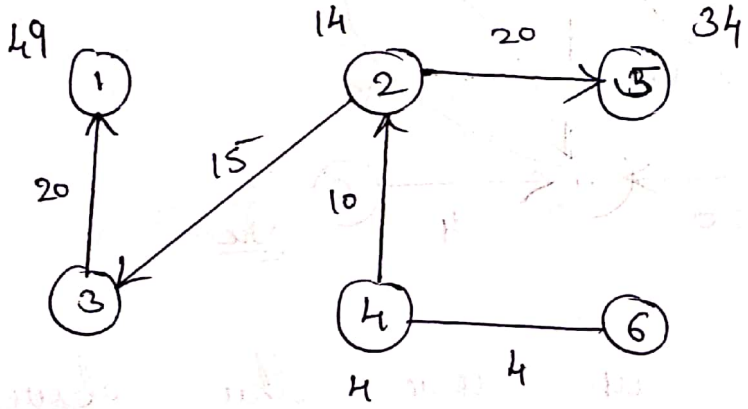


⟹

Step - 4



Step - 5



∴ This is the single shortest path by taking 6 as source vertex.

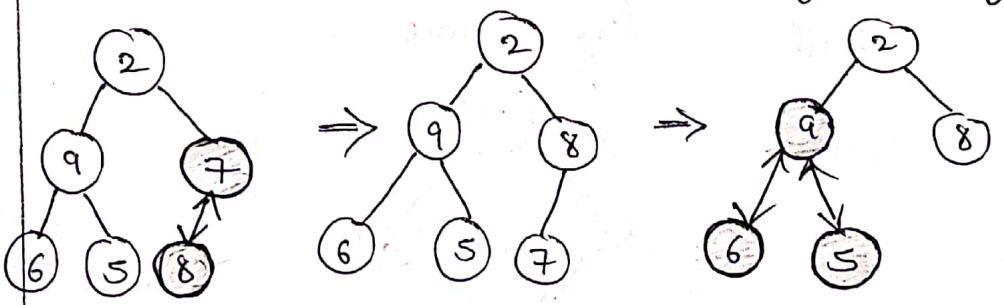
6 c) Sort the given list of numbers using heap sort :

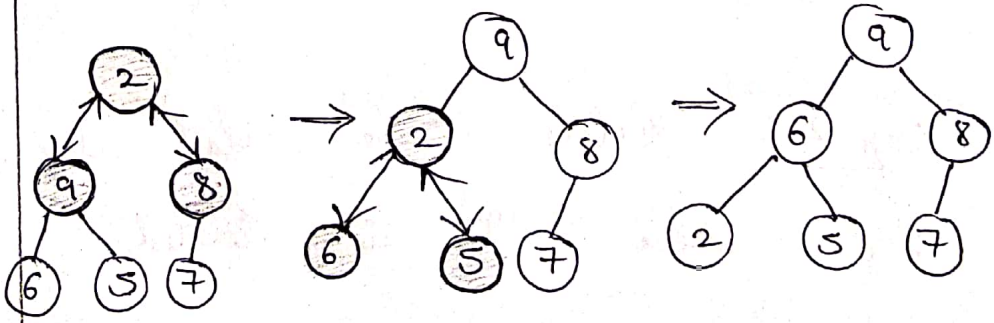
2, 9, 7, 6, 5, 8 [8 marks]

→ Heapsort - This is a two-stage algorithm that works as follows.

Stage 1 (Heap construction) :

Construct a heap for a given array.





Step 1 : 2 9 7 6 5 8

Step 2 : 2 9 8 6 5 7

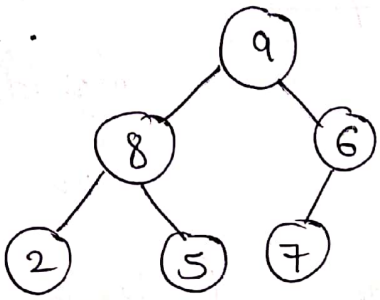
Step 3 : 2 9 8 6 5 7

Step 4 : 9 2 8 6 5 7

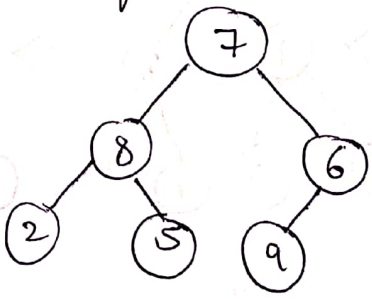
Step 5 : 9 6 8 2 5 7

Stage 2 (maximum deletions) :

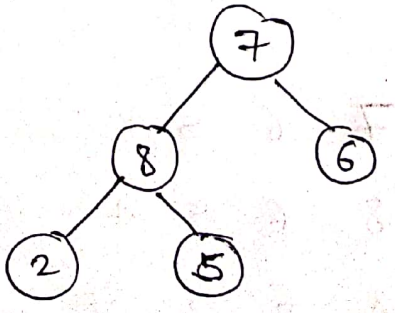
Apply the root - deletion operation n - 1 times to the remaining heap.



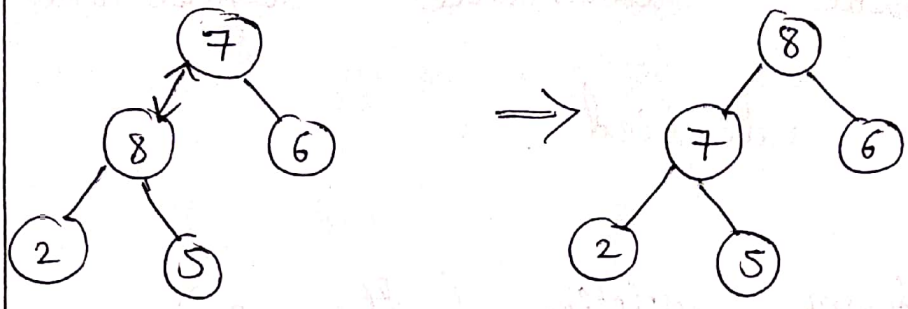
Step 1 : Replace the root node with last leaf node.



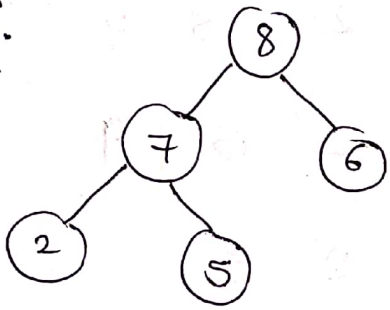
Step 2 : Delete the last leaf node, i.e node 9 is deleted.



Step 3: Compare the nodes and heapify the elements.



Step 4:



Therefore we have deleted root's key from a heap. The key to be deleted is swapped with the last key. After which the smaller tree is "heapified" by exchanging the new key in its root with larger key

in its children until the parental dominance requirement is satisfied.

Maximum deletion (Stage - 2)

[9] 6 8 2 5 7

7 6 8 2 5 | 9

[8] 6 7 2 5

5 6 7 2 | 8

[7] 6 5 2

2 6 5 | 7

[6] 2 5

5 2 | 6

[5] 2

2 | 5

[2]