Name - Arighna Chakraborty Roll - 511 Date - 13.01.21 (1)

Group - A.

Question 1

(i) The running computing time of quick sort depends on whether the partitioning is balanced or unbalanced, which in turn depends on which elements are used for partitioning.

If the partitioning is balanced, the algorithm runs asymptotically as fast as merge sort $O(n \log n)$. If the partitioning is unbalanced, the same algorithm can run asymptotically as slowly as insertion sort $O(n^2)$.

- Worst Case Partitioning — The worst-case behaviour for quicksort occurs when the partitioning routine produces one subproblem with $n-1$ elements and one with zero elements,

$$T(n) = T(n-1) + T(0) + O(n)$$
$$= T(n-1) + O(n)$$

If we sum the costs incurred at each level of recursion, we get an arithmetic series which evaluates to $O(n^2)$.

- Best Case Partitioning – In the best case, the partitioning produces two subproblems, each of size no more than $n/2$.

$$T(n) = 2T(n/2) + O(n)$$

The above recurrence relation has a solution $O(n \log n)$. Hence quick sort is not always "quick" for all input instances.

(ii) Dynamic programming, like divide-and-conquer method, solves problems by combining the solutions to subproblems.

Divide-and-conquer algorithms partition the problem into disjoint subproblems, solve the subproblems recursively, and then combine their solutions to solve the original problem.

In contrast, dynamic programming applies when the subproblems overlap i.e. when subproblems share their subproblems. In this context, divide-and conquer algorithm does more work than necessary by repeatedly solving the common subproblems.

A dynamic programming algorithm, on the other hand, solves each subproblem just once and then saves it in a buffer memory, thereby avoiding the work of recomputing the answer everytime it solves each subproblem.

Question 2

Huffman invented a greedy algorithm that constructs an optimal prefix code called Huffman Code. Here given character set,

A - 0.15

B - 0.10

C - 0.15

D - 0.10

E - 0.40

F - 0.10

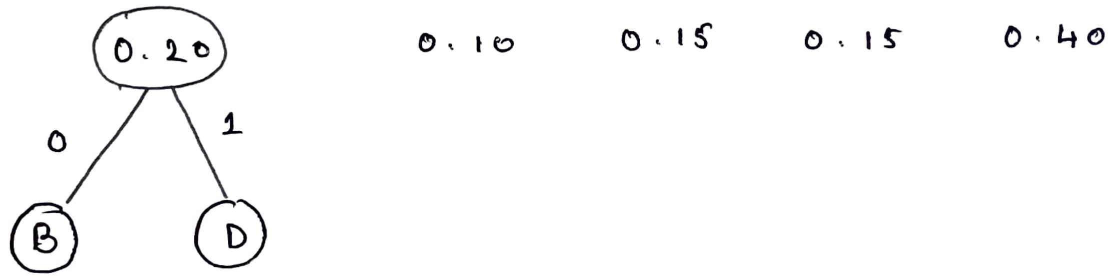
In order to generate a Huffman code from the above character set, we initialize a list of one node binary tree and we connect them, smallest weighted pair at a time in the following manner —

Name - Arighna Chakraborty Roll - 5M Date - 13.01.21

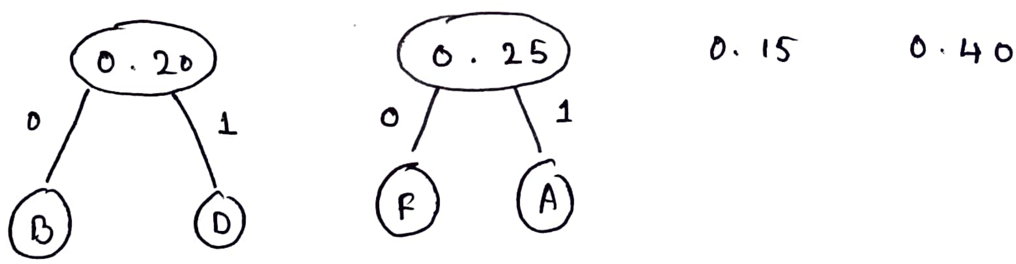
Step 1 - Take two nodes with minimal weights. at each step.

B	D	F	A	C	E
0.10	0.10	0.10	0.15	0.15	0.40

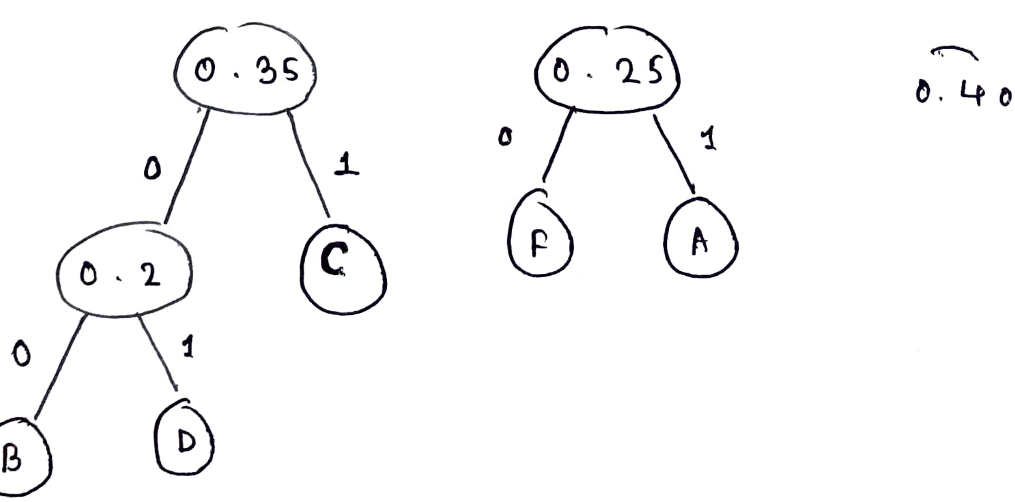
Step 2 - Add the weight and make it parent node.



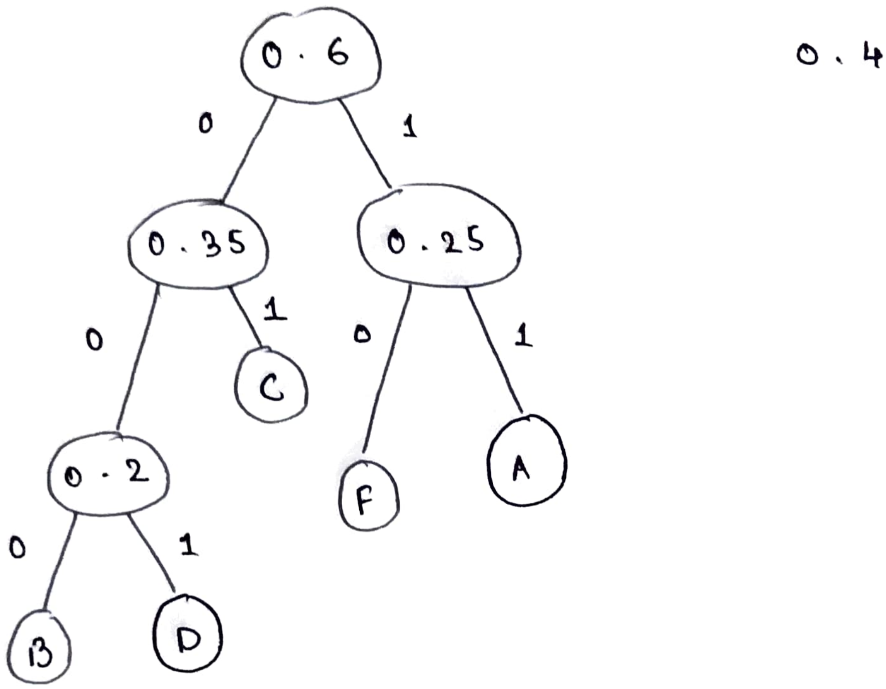
Step 3 -



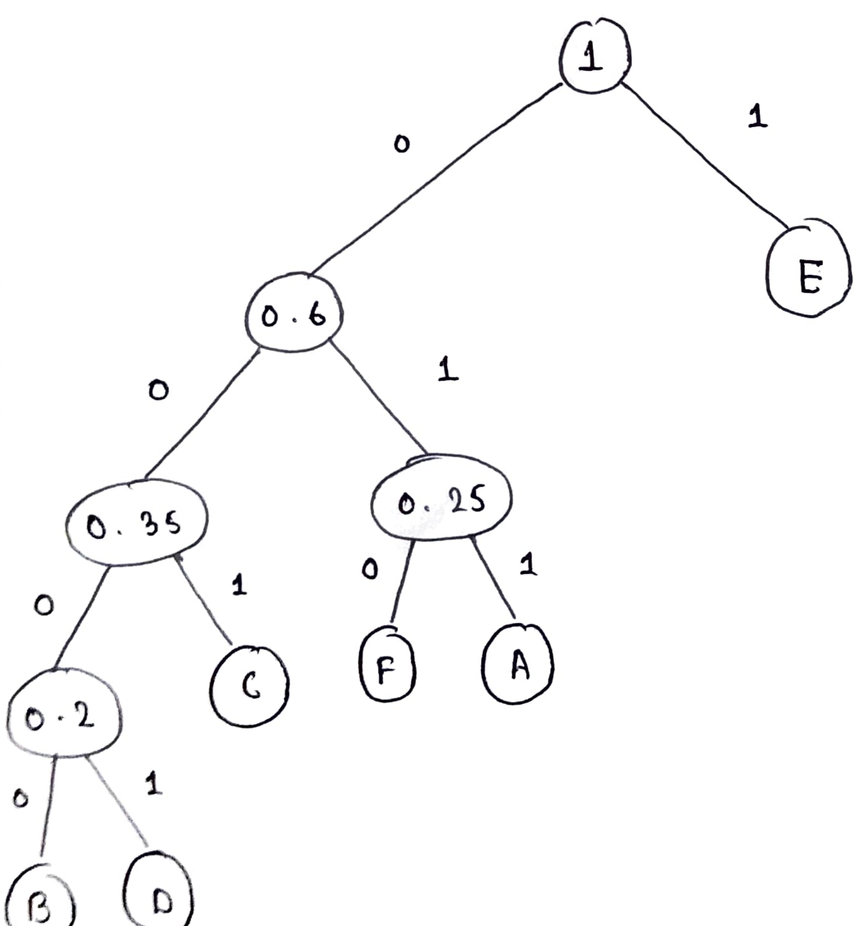
Step 4 -



Step 5 -


0.6
0
1
0.35
0.25
0
1
0
1
C
0.2
F
A
0
1
B
D
0.4

Step 6 -


1
0
1
E
0.6
0
1
0.35
0.25
0
1
0
1
C
F
A
0.2
0
1
B
D

The immediate decodability property of Huffman codes is clear. Each character is associated with a leaf node in the Huffman tree and there is a unique path from the root of a tree to each leaf.

Thus we get,

A - 011

B - 0000

C - 001

D - 0001

E - 1

F - 010

As you can see above, none of the character code bits occur in any other character code. Each character code is unique and hence is immediately decodable.

For example, given a message string,

0 1 0 1 0 1 1 0 1 0

0 1 0	1	0 1 1	0 10
F	E	A	F

Hence, the code produced is immediately decodable.

Question 3

According to the problem,

$$n = 3, \quad W = 20$$

Further given,

w	18	15	10
v	25	24	15
v/w	1.4	1.6	1.5

We obtain the feasible solutions by selecting the objects in,

i) Order of decreasing value
ii) Order of increasing weight
iii) Order of decreasing v/w

We assume that the objects can be broken down into smaller pieces, so we can carry fraction x_i of object i where $0 \leq x_i \leq 1$.

Name . Arighna Chakraborty Roll - 511

Selecting Method	Amounts of Object Selected (x_i)			Total Value
	(x_1)	(x_2)	(x_3)	
Max v	1	0.1	0	25 + 3.2 = 28.2
Min w	0	0.7	1	15 + 16 = 31
Max v/w	0	1	0.5	24 + 7.5 = 31.5

From the above table it is clear tha the most optimal solution to a knapsack problem is achieved when we consider the maximum value per unit weight of the objects.

Question 4

The optimal way to schedule n jobs on a single machine (processor | server) is,

(i) To minimise the average time that a job spends in the system.

(ii) Every job should have a deadline, and a job brings in a certain profit only if it is completed in the deadline.

We will apply the minimising time algorithm to find an optimal solution for a schedule. Suppose there are n jobs. Each job i takes computing time t_i, $1 \leq i \leq n$. We want to minimize the average time that a job spends in the system.

For example, suppose $n = 3$ and $t_1 = 2$, $t_2 = 1$ and $t_3 = 6$. It is observed that the average time for a job is minimized if the jobs are executed in order of their increasing computing time.

1	2	3	$2 + (1+2) + (1+2+6) = 14$
3	2	1	$6 + (6+1) + (6+1+2) = 22$
2	1	3	$1 + (1+2) + (1+2+6) = \underline{\underline{13}}$

Hence, the algorithm is given as

Name - Arighna Chakraborty Roll - 511

function MinScheduling ()

{ Minimizes the average time a job spends.

S ← { Set of Jobs }

T ← { Set of computing time of the jobs }

n ← number of jobs

While (i <= n) do

x ← Select the job with the smallest computing time. from T

Execute (S(x))

end While.

}

The only necessary part of the algorithm is to sort the jobs in order of increasing computing time. Which takes $O(n \log n)$.

Hence the worst case complexity is $O(n \log n)$.

PTO

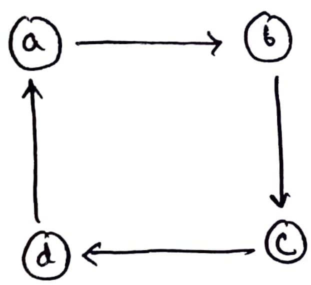
Name - Arighna Chakraborty Roll - 511

Group - B

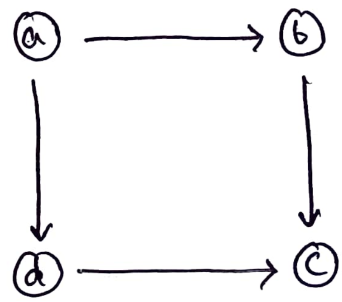
Question 6

a) A directed graph is strongly connected if there is a path from Vertex A to B and from B to A whenever A and B are vertices of the graph.

Wheras a directed graph is a weakly connected graph if there is a path between every two vertices in the underlying undirected graph.



Strongly Connected

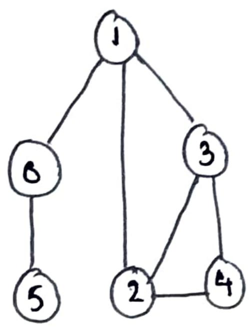


Weakly Connected.

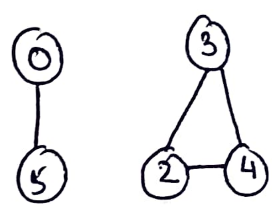
A bipartite graph or bigraph is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V.

(C) In a graph, a vertex is called an Articulation Point if removing it and all the edges associated with it results in the increase of the number of connected components in the graph.

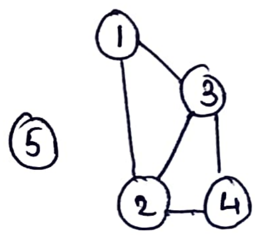
For example, consider the graph,



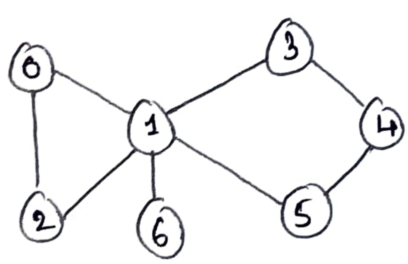
Here vertex 1 and 0 are the articulation points because removing them results in these two graphs,



and



An edge in a graph between two vertices u and v is called a bridge, if after removing it, there will be no path left between vertices u and v. For example,



If we remove the edge connecting vertices 1 and 6, there will be no path connecting 6. Hence the edge is a bridge

Name - Arighna Chakraborty Roll - 511

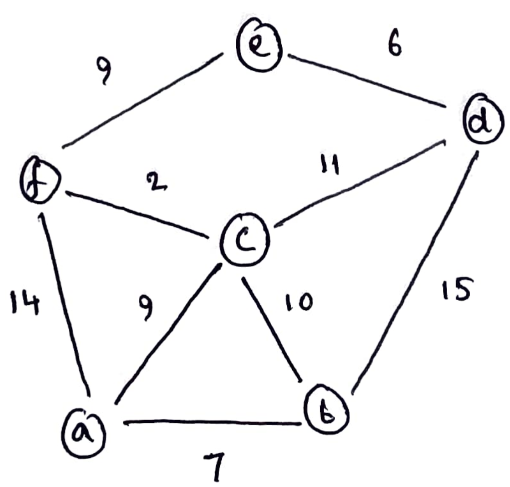
Using the concept of degree of a vertex, a pendant vertex is a vertex with degree as one.

For example,

Here both vertices a and b have a degree of 1. Hence both these vertices are pendant vertices.

Question 8

a) Given the following graph,



BFS starting from e,

e → f → d → c → a → b.

Name - Arighna Chakraborty Roll - 511

DFS starting from a,

a → b → c → f → e → d.

(b) In Krushkal's algorithm, we initially have a forest of n distinct trees for all n vertices of the graph. This algorithm is used to construct a minimum spanning tree, of an undirected edge-weighted graph. Here's how it works —

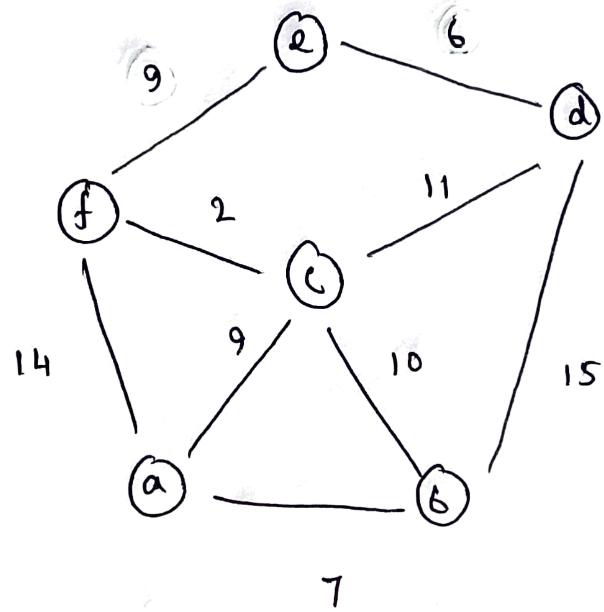
- Create a forest F, where each vertex in the graph is a separate tree.
- Create a set S containing all the edges in ascending order.
- When S is non-empty and F does not have any cycles —

  a) Remove the minimum edge from S

  b) If removed edge connects two different trees then connect it to the forest.

Name - Arighna Chakraborty Roll - 511

Consider the following graph,

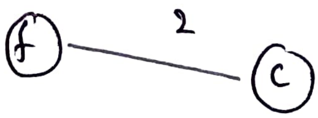


Step 1 : Start with the minimum edge which is 2, connecting vertices f and c.

Root of f = f
Root of c = c

} different.

Since the roots are different, so the vertices belong to different trees. Hence the edge is added.



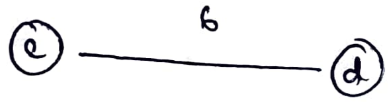
New root of c = f

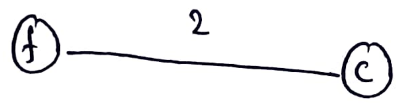
Name · Arighna Chakraborty Roll - 511

Step 2 : Next Smallest edge is 6, connecting e and d.

Root of e = e
Root of d = d
} different

Hence edge is added.





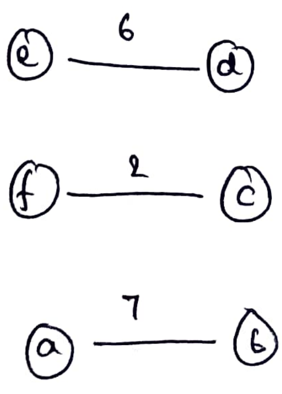
New root of d = e

Step 3 : Next smallest edge is 7 connecting a and b.

Root of a = a
Root of b = b
} different.

Hence edge added.

New root of b = a

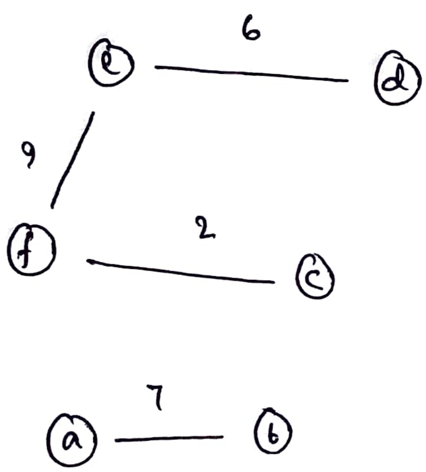


Step 4 : Next smallest edge is 9 connecting f and e

Root of f = f
Root of e = e } different hence edge added.

New root of f = e

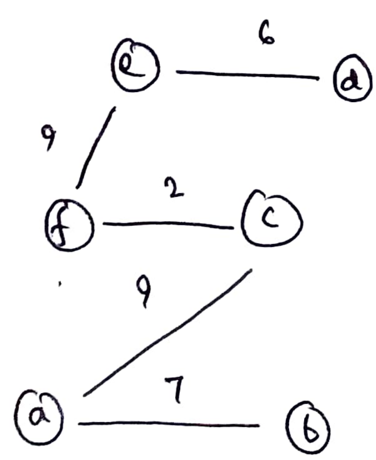
New root of c = e



Step 5 : Next edge is 9 connecting c and a

Root of c = e
Root of a = a } different

New root of a and b = e



Step 6 : Next edge is 10 connecting c and b

Root of c = e
Root of b = e
} Same root hence edge ignored.

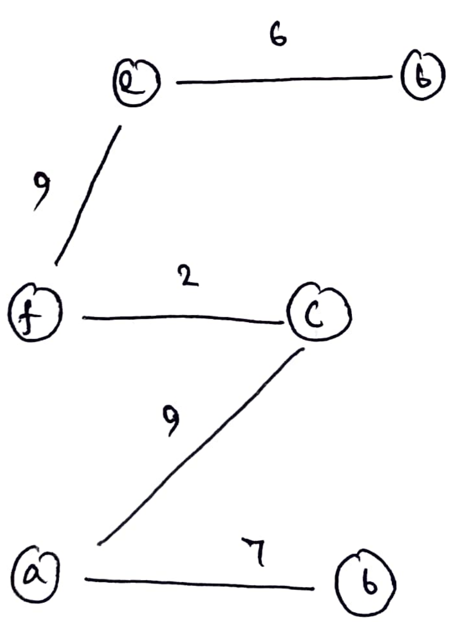
Step 7 : Next edge is 11 connecting c and d

Root of c = e
Root of d = e
} same root hence edge ignored.

Step 8 : Next edge is 14 connecting f and a

Root of f = e
Root of a = e
} Same hence edge is ignored.

Hence the resultant minimum spanning tree,



Root of the tree = e

Cost of MST = 33.