CLO #4: Implement the algorithms, compare the implementations empirically, and apply fundamental algorithmic knowledge to solve practical problems related to the program.

Q2) It is observed in NUCES that our LAN is not properly designed and is not using the available resources to its fullest. As you people are familiar with the fact that the NUCES Lahore campus consists of various building blocks so the LAN needs to connect the classrooms, labs, faculty offices in these buildings efficiently.

We have modeled this problem to a graphs, where computers in classrooms, labs, offices are represented by nodes and the networking wires that connect them are represented by edges. The edges have weight that indicates the length of the wire needed to connect two nodes. The management wants to design a LAN network, FLAN that covers all the nodes with minimum wire length and do so efficiently.

a. **a. Which graph algorithm (that you have already studied in class you think is most suitable to compute the FLAN and what is time complexity in term of |V| and |E|?**

b. **b. The ideal FLAN may not be possible due to various issues. For example, the optical fiber might need underground wiring which means additional cost even if the length of the required wire is short. After providing the program that produces the best FLAN, you are now asked to add the functionality that will help the director's office to modify the networking by choosing to disallow a certain connection in the existing network. When a connection is disallowed, the FLAN cannot use that connection anymore. Your algorithm should update the best network accordingly. It should take linear time to do so i.e O(|V|+|E|).**

c. **c. A recent technology has been introduced and it has been estimated that a direct connection between any two nodes in the network can in fact be cabled in $5000, regardless of the distance between them. This means that the cost of each connection becomes a fixed amount of $1000. Additionally, k of these connections have been sponsored by a software house. Without even computing the ideal network, can you tell what could be the minimum possible cost of the network under the new scenario that FAST needs to pay?**

CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

Q3:

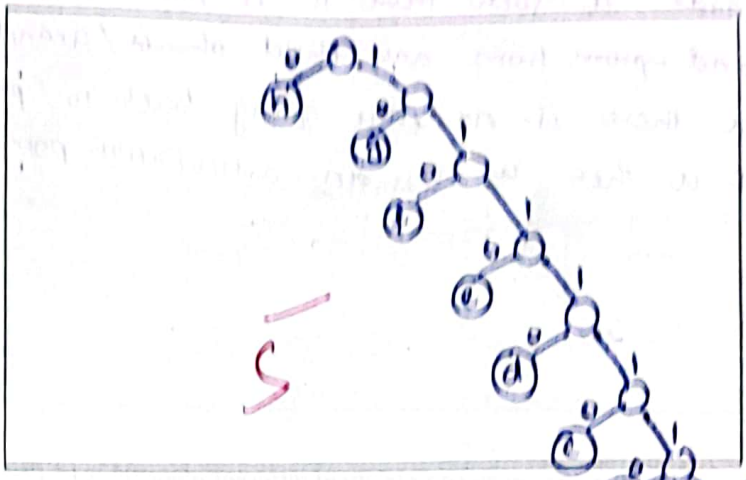
a. **Given a binary string 5 of type(1^m0^n), devise an algorithm that finds the number of zeroes in 0(log k) time. Here m+n=k. Write 3-4 lines to explain your idea.**

Binary String = {1111000000}
we use divide and conquer approach to divide the string. Here upon dividing we get a zero so we have atleast $\frac{k}{2}+1$ zeroes. we can have more so we divide the right string
if we get 0 again we repeat and add the half of the string we, in our answer. we stop when we get 1, then we divide the right substring and repeat the recursive process

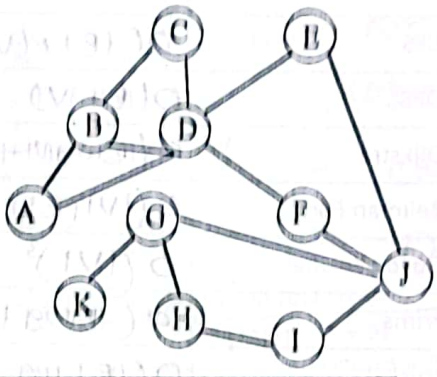
```
zeroes(s, l, r){
  m = l+r/2
  if s[m]='0'
    return 1+zeroes(s, l, m);
  else
    return zeroes(s, m, r);
}
```

b. Few days back I generated these codes using Huffman coding. Unfortunately, I lost the paper where I drew that tree. Please help me in re-generating the tree using given codes

Character	Code
a	1111110
b	1111111
c	111110
d	11110
e	1110
f	110
g	10
h	0



c. Suppose you are a medical practitioner and it is your job to vaccinate people in each village of Pakistan to limit the spread of adenovirus in the country. The adenovirus can be transmitted between two people if they handshake. For each village, you are given a handshaking graph, G, whose vertices are the people in that village and whose edges are pairs of people who regularly handshake. You have limited supply of the vaccine, and you must have to vaccinate the central hand shakers only, where in the graph a central hand shaker is a person D, such that there are no two people C and E, who are hand shake by D such that there is a sequence of handshaking pairs of people that starts with C and leads to E while avoiding D.



I. Which vertices are the central hand shakers in this graph?

II. Describe an efficient algorithm for identifying all the central hand shakers in G, and provide its running time. Explain your idea in 3-4 lines.

(i) J, G, D are central hand shakers

(ii) We need to find articulation points. We can use modified DFS to compute the low value for every vertex low[v] = min(d[v], lowest low[w] from tree edges, lowest d[w] in back edges)
if there exists a node u such that for a child w low[w] ≥ d[u] then u is articulation point / central hand shaker.

Running time = O(|E| + |V|)

In DFS tree, for root if root has two or more children it is central handshaker

Q) How can we say in DFS tree if there exist a node u such that it is not root of the tree and you can't any child of u / grandchild of u there is no edge going back to proper ancestor of u then u is an articulation point / cut vertex.

TO #2: Analyze the time and space complexity of different algorithms by using standard asymptotic notations for recursive and non-recursive algorithms.

[Marks: 5 + 5]

a. Fill the Table. Also mention the choice of data structure for graph algorithms.

Algorithm Name	Worst Case Time Complexity	
BFS	O(|E| + |V|) ✓	using Queue DS ✓
DFS	O(|E|+|V|) ✓	adjacency list for graph
Dijkstra	O(|E|log|V|+|V|log|V|) ✓	Priority Queue
Bellman Ford	O(|V||E|) ✓	adjacency list for Graph
Floyd Warshal	O(|V|)³ ✓	adjacency list for Graph
Prims	O(|E|log|V|) ✓	Priority Queue
Kruskal	O(|E| log |V|) ✓	Priority Queue ✓
Activity Selection Problem	O(n²) ✗	✓
Huffman Coding	O(|V| log|V|) nlogn ✓	using MinHeap
0-1 knapsack Problem	O(n²) / O(W*I) ✓	W = total weight, I = no. of items
Fractional Knapsack Problem	if sorted already then O(n) otherwise, n log n ✓	
Edit Distance	O(n²) / O(n1×n2)	n1 = length of string1, n2 = length of string 2
Rod Cutting	O(n²) / O(L * pieces)	L = length of rod, pieces = possible pieces we can cut in

National University of Computer and Emerging Sciences
Lahore Campus

b. *Derive the recurrence of the given recursive function and solve the recurrence.*

```
tribonacci ( n )
{
  If n == 0
     return 0
  else if ( n == 1  ||  n == 2 )
     return 1
  else
     return  tribonacci ( n/2 )  +  tribonacci ( n/4 ) + tribonacci ( n/4 )
}
```

$T(n) = T(\frac{n}{2}) + 2T(\frac{n}{4}) + c$

height of tree will be decided by left most subtree $(\frac{n}{2})$.

level	p.b size	no. of nodes	cost. 1 node	cost level
0	n	1	c	$1c$
1	$n/2$	3	c	$3c$
2	$n/4$	9	c	$9c$
i	$n/2^i$	3^i	c	$3^i.c$
k	$\frac{n}{2^k} = 1$	3^k	c	$3^k.c$

$\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

$T.C = 1c + 3c + 9c + \cdots 3^k c$

$= c[1 + 3 + 9 \cdots 3^k]$

Number of levels $= \log_2 n + 1$

cost at one level $= c$

Time complexity $= (\log_2 n + 1)^* c$

$= c\log_2 n + c$

$= O(\log_2 n)$

CLO #3: Evaluate the correctness of algorithms by using theorem proving or executing test cases

[Marks: 5 + 5]

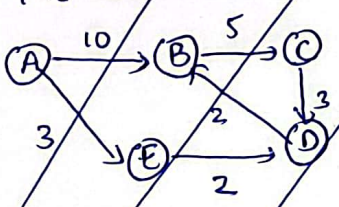
Q5:

a. Following algorithm takes a connected graph and a weight function as input and returns a set of edges T. Prove that T is a minimum spanning tree or prove that T is not a minimum spanning tree.

```
MAYBE-MST (G,w)
1    T ← Nil
2    for each edge e, taken in arbitrary order
3    T ← T U { e }
4    if T has a cycle c
5          let e' be a maximum-weight edge on c
6          T ← T − { e' }
7  return T
```

Answer on Answer Sheet.

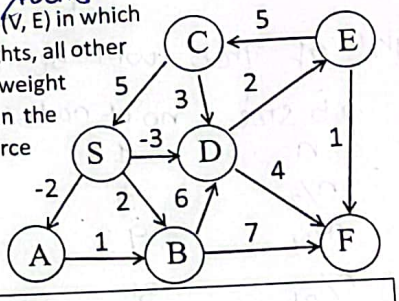
T is not MST. Consider counter example



we will randomly select edge A,B then B,C, C,D and D,B cycle exist so we remove B,C then we take E,D and A,E no cycle exist but the M.ST we get is wrong.

Also by removing the edge B,C which was the max weight edge in cycle we lose a path to node C, so this algorithm fails..

b. Suppose that you are given a weighted, directed graph G = (V, E) in which edges that leave the source vertex may have negative weights, all other edge weights are nonnegative, and there are no negative-weight cycles an example of such graph is provided below. Can the Dijkstra's algorithm correctly find shortest paths from source S? Provide the valid justification.



Yes, such algorithm will provide shortest path.

Problem occured when we took Djikstra and a node could not be relaxed because we already relaxed it and the negative edge did not contribute to shortest path in final answer. Here Djikstra will first account for negative weight edge and relax the nodes connected to source since they get relaxed then there is no chance we will miss out on any relaxation.

FINAL EXAM ANSWER SHEET

National University
of Computer & Emerging Sciences

FAST

16/5/24

SEMESTER : ☑ SPRING ☐ SUMMER ☐ FALL Year 2024

Please Tick Appropriate Boxes

Course Design & Analysis of Algorithms Serial No./ 05352

Roll No. 221-0504 Section BBS-4A Date

Serial No. of continuation sheet(s) attached

INSTRUCTIONS FOR CANDIDATES

- Write Question No. In the middle of the line using thick tipped pen.
- Use only blue or black pen to write your answers.
- Answers written using pencil will not be checked.
- Pencil is only allowed to draw diagrams or write program code.

(THIS ANSWER BOOK CONTAINS NOS. 1-18)

Q./Part No.	Marks
Q. - 1	
Q. - 2	
Q. - 3	
Q. - 4	
Q. - 5	
Q. - 6	
Q. - 7	
Q. - 8	
Q. - 9	
Q. - 10	

Q./Part No.	Marks
Q. - 11	
Q. - 12	
Q. - 13	
Q. - 14	
Q. - 15	
Q. - 16	
Q. - 17	
Q. - 18	
Q. - 19	
Q. - 20	

Marks Obtained

Total Marks

Examiner's Signature

Date

art No.

Rough Work

Question 01

Recursive

MaxProfit [t, b, n] = max [maxProfit(t, b, n-1), } if Tn ≤ t
maxProfit(t-tn, b, n-2) + bn]

= maxProfit(t, b, n-2) if Tn > t

We make two calls for the MaxProfit
for one call we pass
MaxProfit(t, b, T, N)
for other we pass
MaxProfit(t, b, T, N-1)

The max of these two results will be our final answer.

Bottom-up solution

```
MaxProfit(t, b, T, N) {
    init dp[0:T][0:N]
    for (i=0 to T)
        dp[i][0] = 0
    for (i=0 to N)
        dp[0][i] = 0

    for (i=0 1 to T)
        for (j=1 to N) {
            if (T < t[j]) {
                dp[i][j] = dp[i-1][j];
            } else {
                dp[i][j] = max( dp[i-t[i]][j-1] + b
                                dp[i-1][j] );
```

Q / Part No.

```
if (i==1){
    if(T<t[1])
        dp[i][j]=0;
    else dp[i][j]= b[i];
}

MaxProfit (b,t,T,N){

int dp[0:N][0:T];

for (i=0 to N)
    dp[i][0]=0
for (i=0 to T)
    dp[0][i]=0

for (i=1 to N)  <- increment i by 2
    for (j=1 to T){

else{   if(T<t[i])
        if (T< t[i])
            dp[i][j]= dp[i-1][j];
        else{

            dp[i][j]= max( dp[i-2][j],
dp[i-2][j-t[i]]+ b[i] )
        }
}
    }

for (i=2 to N) (increment i by 2)
    for (j=2 to N){

        if (T< t[i])
            dp[i][j]= dp[i-2][j];
        else{
            dp[i][j]= max( dp[i-2][j],
dp[i-2][j-t[i]]+ b[i] )
        }
}
```

12

Rough Work

return max(dp[N][T], dp[N-1][T/2]);

}

Time complexity = ~~O(T*N)~~ 2 O(T*N)

first nested loops fills half of dp array the other nested loop fills other half.

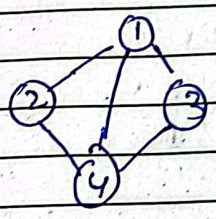
Question. 02

(a) ~~Prims Algorithm for MST~~.

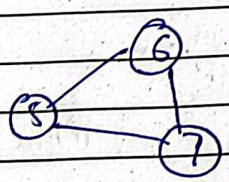
Kruskals Algorithm for MST

Time complexity = O(|E| log |V|) 2

(b) By removing one node we get two separate connected components say A and B.



A



B

we can run DFS on A and color all the nodes black.

Now we run DFS on any node of B. in G

In DFS-visit, if we encounter any node with color black we store that edge in minEdge var we will update minEdge variable if we encounter any other edge such that one node is white and other is black is its weight < minEdge.

Finally we get the minimum Edge after DFS is complete. this minimum edge ≠ the one rem

Q / Part No.

and joins two components back together. This takes O(|V|+|E|) time. The new MST = oldMST ∪ minEdge 10

(c)

If there are |V| nodes then FAST will need to pay

(|V|-1-K) * 1000 $ for the connection 3

Question. 05

(a)

The tree will be a Minimum Spanning Tree.

This method is similar to Kruskal's Algorithm where we pick edges in increasing order of their weight and dont allow any cycle to occur. Here instead of picking edges in increasing order we arbitrarily chose edges and if cycle occur we remove the edge in cycle with max weight. 5

This approach works because we are always removing the edge of cycle with maximum weight

Proof by counter example

Say that T we get is not M.S.T if T is not M.S.T then there exists an M.S.T T' such that T' contains an edge with less weight than one in T. This means T contains an edge with

Rough Work

Part No.

weight greater but we already removed all cycle forming edges in T with max weight. we made optimal choice at occurence of every cycle.
Hence T is a MST.

DFS_visit
time++
d[v] = time
π[v] = 0