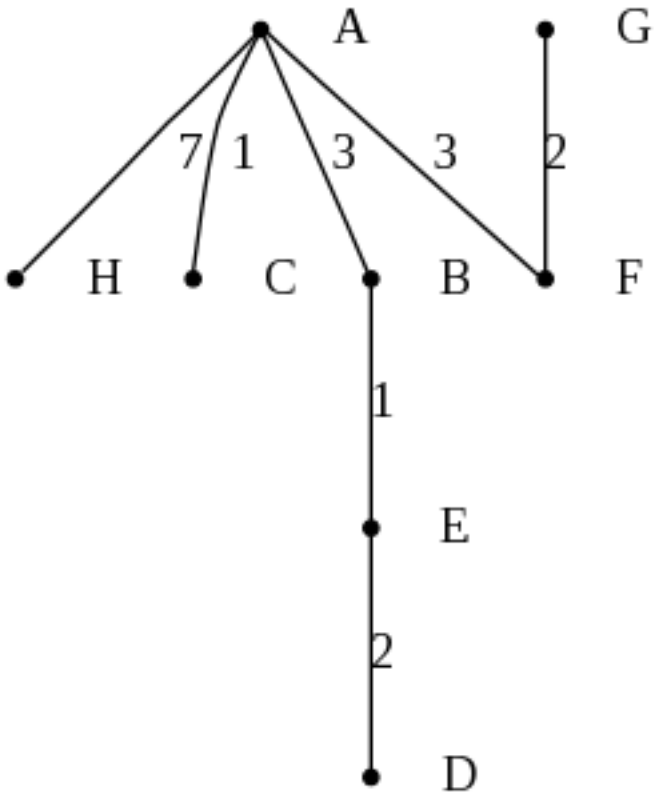
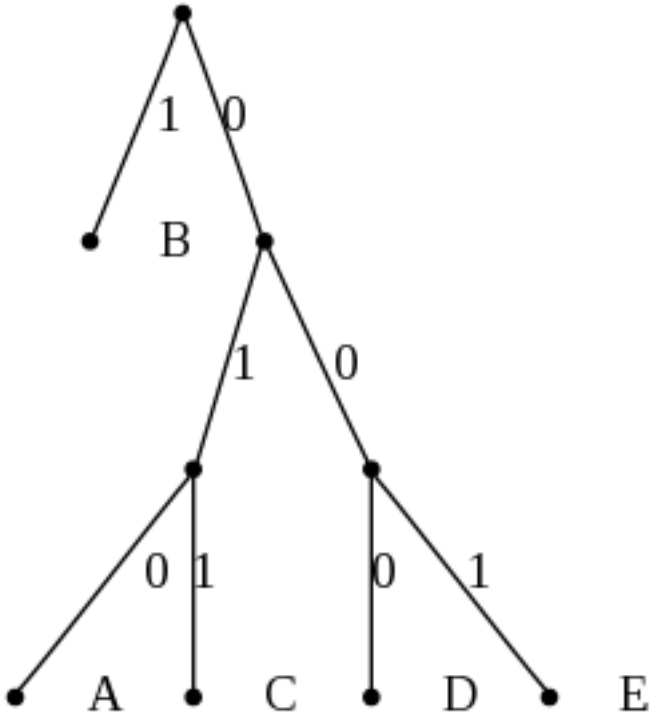
Quiz 3 Solutions
CS3510

1. **MST:**



weight: 19

2. **Huffman Codes:**



Character	Encoding
A	010
B	1
C	011
D	000
E	001

Your Huffman tree should look something like this, but it is possible that your edge labels are swapped, as well as the values of the final bit for two children of the same node. If your tree has a greater height than this tree, i.e., some of your encodings have more bits than necessary, then you probably built your tree by choosing the highest frequency character first. You were instead supposed to build from the lowest frequency values and work your way up to minimize the tree height.

The encoding of ABBCCED using this particular tree results in the binary string 10000101101111110.

3. **True / False**

(a) Suppose all the weights in a graph are distinct. Then the longest

edge cannot be in the minimum spanning tree.

This statement is false. If a graph contains no cycles, then the longest edge will be necessary so that all nodes are reachable in the minimum spanning tree.

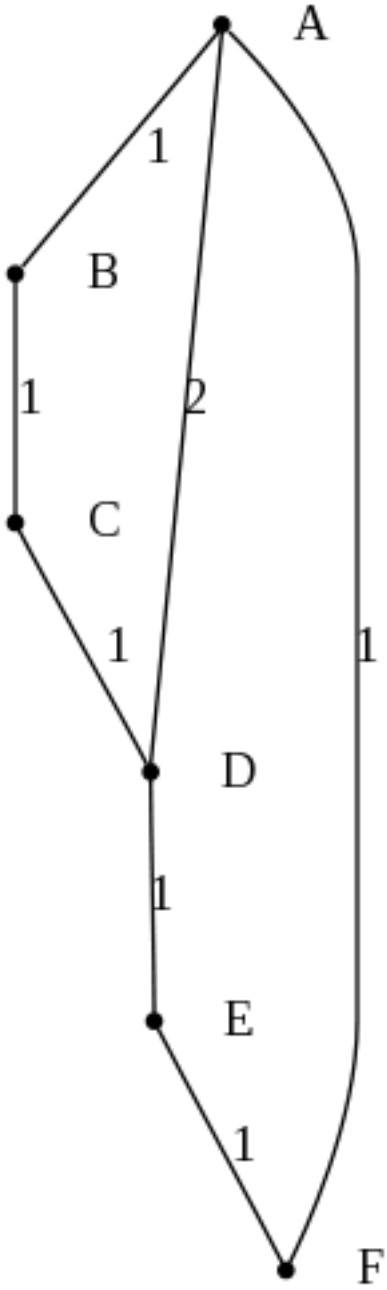
(b) Suppose all weights in a graph are distinct. Then the minimum spanning tree is unique.

True. If there are edges with equal weights in the graph, then the minimum spanning tree algorithm will have to choose arbitrarily between these edges, alphabetically for instance. Which means that another MST could be found if a different tie-breaking method were used. However, if every weight is distinct then the outcome is deterministic and there is a single unique minimum spanning tree for a particular graph.

(c) The Union by Rank data structure always creates balanced trees where all the leaves are on the kth or $k+1$st level of the tree, for some k.

False. If a rank 0 tree is joined with a rank 3 tree, then the shorter tree, a leaf, will be attached directly to the root putting it at depth 1. The leaves at depth 3, that gave the root its rank, will have a distance of two from the previous node thus disproving the the original claim

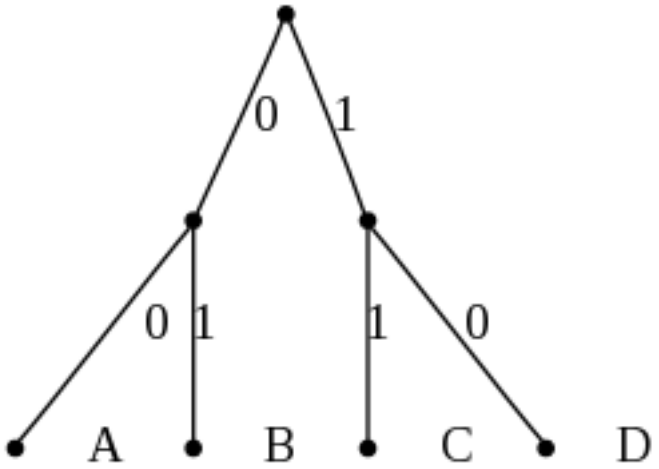
(d) In an undirected graph, the shortest path between two nodes always lies on some minimum spanning tree.



In the process of determining the minimum spanning tree of this graph, the edge with weight 2 will always be examined last, at which point it will introduce a cycle and thus be discarded. This means that the MST will only a contain a path from A to D with weight 3,

which is not the shortest path.

(e) In a Huffman coding, the item with the greatest probability is always a child of the root.



Character	Frequency
A	5
B	6
C	7
D	8

There are no direct children of the root node in this case, including the highest frequency element, D, thus countering the original claim.

4. **Dynamic Programming**

(a) Formally define the set of sub-problems you will solve.

Let $K(i)$ be the maximum value of the committee while considering the first i people.

(b) Give the recurrence for the solution of a given sub-problem in terms of other sub-problems.

$$K(i) = max \begin{cases} v_i + K(i-3) \\ K(i-1) \end{cases}$$

This handles the two possibilities for the value of $K(i)$ at a given position. The ith person is either included or not. If included, then

we must exclude possible value contributions from neighbors, which is why get the value of the rest of the maximum committee from position $i-3$. If the current person is not included, then we can take the value of the maximum committee considering all people up to and including the $i-1$th person.

(c) What are the base cases?

The only positions where the above recurrence cannot be used are the first three people considered. So we initialize values of $K(1)$, $K(2)$, and $K(3)$. At position one, only one person is considered so the maximum committee value is just v_1. The committee after considering the first two positions will end up being the maximum of v_1 and v_2. This is because a choice of of either position will exclude the other. The same logic follows for the best committee after considering the first three people.

$$K(1) = V_1$$

$$K(2) = max(v_1, v_2)$$

$$K(3) = max(v_1, v_2, v_3)$$

(d) Give a non-recursive pseudo-code specification of the algorithm and state its complexity in terms of n.

```
1   K(1) = v_1
2   K(2) = max(v_1, v_2)
3   K(3) = max(v_1, v_2)
4   for i = 3 to n
5     if v_i + K(i − 3) > K(i − 1)
6       K(i) = v_i + K(i − 3)
7       position[i] = i
8     else
9       K(i) = K(i − 1)
10      position[i] = position[i − 1]
```

We can find the chosen committee members by a single pass of the position array. Since this and the original algorithm only do a constant amount of work over a single pass of the array, the overall running time is $O(n)$.