Q1. (10 points) Given a undirected, weighted graph G=(V,E). Assume that no two edges have the same weight. Prove that there is a unique minimum spanning tree (MST) for such a graph.

Suppose the graph has two different MSTs T_1 and T_2. Let e be the lightest edge which is present in exactly one of the trees (there must be some such edge since the trees must differ in at least one edge). Without loss of generality, say $e \in T_1$. Then adding e to T_2 gives a cycle. Moreover, this cycle must contain an edge e' which is (strictly) heavier that e, since all lighter edges are also present in T_1, where e does not induce a cycle. Then adding e to T_2 and removing e' gives a (strictly) better spanning tree than T_2 which is a contradiction.

Q2. (10 points) Consider the following algorithm for finding a MST. Given a graph G=(V,E), sort the edges in decreasing order of their weights. Pick each edge e in sorted order, and if e creates a cycle in G, remove it from G. Note that each time you remove an edge, you are modifying the graph, so that the next check for a cycle will be on the reduced graph. Prove that this method is correct, and analyze its running time.

(a) Consider an MST T which contains e. Removing e breaks the tree into two connected components say S and $V \backslash S$. Since all the vertices of the cycle cannot still be connected after removing e, at least one edge, say e' in the cycle must cross from S to $V \backslash S$. However, then replacing e by e' gives a tree T' such that $cost(T') \leq cost(T)$. Since T is an MST, T' is also an MST which does not contain e.

(b) If e is the heaviest edge in some cycle of G, then there is some MST T not containing e. However, then T is also an MST of $G - e$ and so we can simply search for an MST of $G - e$. At every step, the algorithm creates a new graph $(G - e)$ such that an MST of the new graph is also an MST of the old graph (G). Hence the output of the algorithm (when the new graph becomes a tree) is an MST of G.

(c) An undirected edge (u, v) is part of cycle iff u and v are in the same connected component of $G - e$. Since the components can be found by DFS (or BFS), this gives a linear time algorithm.

(d) The time for sorting is $O(|E| \log |E|)$ and checking for a cycle at every step takes $O(|E|)$ time. Finally, we remove $|E| - |V| + 1$ edges and hence the running time is $O(|E| \log |E| + (|E| - |V|) * |E|) = O(|E|^2)$.

Q3. (10 points) Consider the problem of a constrained MST. Given a weighted, undirected graph G=(V,E), and given a subset of vertices $C \subseteq V$, describe a method to find a constrained MST such that vertices in C only appear as leaves in the constrained MST. What is the running time of your algorithm? Note that for some choice of C, it may be the case that the constraint cannot be satisfied, i.e., it may be the case that there is at least one vertex in C that cannot be made a leaf node. In these cases, you can output that the constraint cannot be satisfied.

We first note that each $u \in U$ must have at least one neighbor in $V \backslash U$, else the problem has no solution. If T is the optimal tree, then $T \backslash U$ must be a spanning tree of $G \backslash U$. Moreover, it must be a minimum spanning tree since the nodes in U can be attached as leaves to *any* spanning tree. Hence, we first find an MST of $G \backslash U$ (in time $O(|E| \log |V|)$) and then for each $u \in U$, we add the lightest edge between u and $G \backslash U$ (in time $O(|E|)$).

Q4. (15 points) Given an alphabet with n characters. Let the characters be numbered from 1 to n, and let the frequency of character i be $f_i = 1/2^i$ for $i = 1,2,\ldots,n-1$, with the last character n having frequency $f_n = 1/2^{n-1}$. For example, if n=6, then the frequencies of the first five characters are 1/2, 1/4, 1/8, 1/16,

1/32, respectively, and the frequency of the last character is 1/32. Answer the following questions:
a) (5 points) Show that the frequencies of all characters sum to 1 (as they should) for any *n*.
b) (5 points) Show what the huffman encoding is for each character. In building the encoding tree, the larger frequency child should be on the left, and if there are two groups of characters with the same frequency, the one with the smaller index should be on the left.
c) (5 points) What is the expected number of bits per character? Let the encoding length of character *i* be denoted as l_i, then the expected or average number of bits for the encoding is computed as $\sum l_i f_i$. Use this formula to derive a closed form expression (for any n).

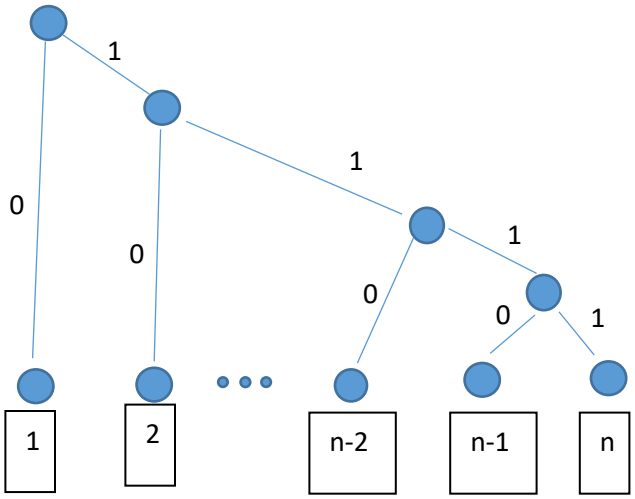
a. We use geometric series summation for first n-1 characters to obtain

$$1/2 + 1/4 + 1/8 + \ldots + 1/2^{n-1} = 1/2(1 + 1/2 + 1/4 + \ldots + 1/2^{n-2})$$
$$= 1/2 \times (1 - 1/2^{n-1})/(1-1/2) = 1-1/2^{n-1}$$

Now we add in the last character to get $(1-1/2^{n-1}) + (1/2^{n-1}) = 1$

b. The first two characters to be merged in the code are the last two with the smallest frequency, namely characters n-1 and n, which results in a combined frequency of $1/2^{n-1} + 1/2^{n-1} = 1/2^{n-2}$

Next this will merge with the 3rd smallest, which is character n-2, which has frequency $1/2^{n-2}$, so this will combine with the previously merged subtree to get a combined frequency of $1/2^{n-3}$.



This will continue, since the two lowest frequency nodes combine to produce a merged subtree with the same frequency as the next lowest node. The final Huffman coding tree thus looks like as shown above.

The code for the characters is as follows:

1: 0, len = 1

2: 10, len = 2

3: 110, len = 3

…

n-1: 1…10 (n-2 1's followed by 0), len = n-1

n: 1…11 (n-2 1's followed by 1), len = n-1

c. The expected bit length for a character is given as 1/2 x 1 + 1/4 x 2 + 1/8 x 3 + … + $1/2^{n-1}$ x n-1 + $1/2^{n-1}$ x n-1

= $1/2^{n-1}$ x n-1 + $\sum_{i=1}^{n-1} i \times 1/2^i \cong \sum_{i=0}^{n} i \times 1/2^i \leq 2$

The last step follows from the fact that $\sum_{i=0}^{\infty} i \times r^i = \frac{r}{(1-r)^2}$, and we have r=1/2

In other words, we expect only 2 bits for each character in the encoded string.

For example, if n=6, and let there be a string of length 32 that follows the frequency distribution given in the question, namely character 1 occurs with probability ½, i.e., 16 occurrences out of 32, character 2 occurs 1/4 x 32 = 8 times, char 3 occurs 1/8 x 32 = 4 times, char 4 occurs 1/16 x 32 = 2 times, and chars 5 and 6 occur 1 time each.

Then the encoding is 1: 0, 2: 10, 3: 110, 4: 1110, 5: 11110, 6: 11111

Therefore encoding length of the string is

16x1 + 8x 2 + 4 x 3 + 2 x 4 + 1 x 5 + 1x 5 = 16+16+12+8+10 = 62 <= 64 bits

This is, at most 2 bits per char in the string of length 32.