Practice Assignment 3

Due: Tuesday, February 2 at 2PM to Canvas

To get credit, each student must submit their own solutions (which need not be typeset) to Canvas by the due date above – no late submissions are allowed. Note that while you must each submit solutions individually, you are free to work through the problem sets in groups. Solutions will be posted shortly after class and may be discussed in class. These will be not be graded on the basis of *completion* alone, not *correctness*.

In the following questions, you are given a graph G with n vertices and m edges and positive weights on the edges.

1. Give pseudocode for Prim's MST algorithm that makes it clear what the running time is using a priority queue data structure. What is the asymptotic running time in terms of n and m?
2. Give pseudocode for Borůvka's MST algorithm that makes it clear what the running time is using a *union find* data structure (as for Kruskal's algorithm) without contracting edges of G. What is the asymptotic running time in terms of n and m?
3. (a) Prove that if the weights of the edges of G are unique then there is a unique MST of G.
   (b) Note that the converse is not true. Give a counterexample.
4. The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V; E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.
   a. If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.
   b. If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.
   c. Let e be any edge of minimum weight in G. Then e must be part of some MST.
   d. If the lightest edge in a graph is unique, then it must be part of every MST.
5. Suppose we want to encode the following phrase "`bananas are tasty`" in binary. There are 9 characters that need to be represented (including the space); since k-bit code-words can represent 2^k letters, we need 4-bit code-words. Assigning one of these 4-bit code words to each letter $\{a, b, e, n, r, s, t, y, ' '\}$ allow us to encode the phrase in $17 \times 4 = 68$ bits. Having each code-word be the same length allows us to easily know the start and end positions of each letter.

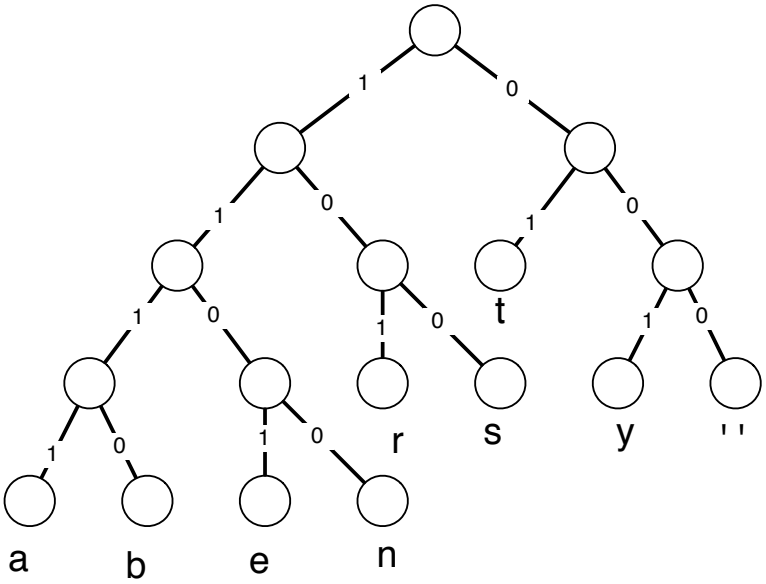
   Suppose we allowed for different length codewords such as:

a	1111
b	1110
e	1101
n	1100
r	101
s	100
t	01
y	001
' '	000

   The phrase encodes in 58 bits to: 11101111110011111100111110000011111011101000011111110001001

   Note that while it is not obvious where one code-word starts and another ends, we can still decode this phrase because *no code-word is the prefix of another code-word.* This is known as a prefix-free code.

It seems quite cumbersome to decode until we notice that we can represent the code with a tree:



Develop another code that encodes "`bananas are tasty`" in fewer than 58 bits.