Mihai Dan
1/31/2016

Practice Problems #3

Problem 1.) Prim's MST Algorithm

```
initialize array of size n to contain MST
initialize Key[] and picked[] arrays
for i=0 to n
        key[i] is set to infinity
        picked[i] is set to false
set key[0] to zero because first location
set MST[0] to the first location
for i=0 to n
        set currentMin to the smallest value in key array
        set picked[currentMin] to true
        for v=0 to n
            if picked[v]==false and current value < key[v]
                add currentMin to MST[v]
                update the key value at that spot
        end
return MST[]
```

The runtime for this problem, in terms of n vertices and m edges is $O(n^2)$ because it traverses through the number of vertices twice with the nested for loops.

Problem 2.) Borůvka's MST Algorithm

```
initialize all vertices as individual sets
initialize an empty MST array
while there is more than one set
        for i=0 to m edges
```

find the smallest edge that connects two sets
for i=0 to n vertices
add the vertices from edges found above to MST[]
end
return MST

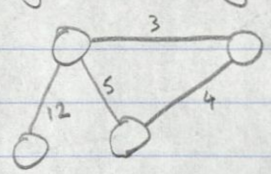
This algorith will run through every edge at least once, but may not need to run through every vertex because it has already been added. This then makes the runtime $O(m \log n)$.

Problem 3.) (a) Let $T=(V,S)$ and $T'=(V,S')$ be two distinct MST's for graph G. Assuming that all edge values are distinct, picking next minimal edge e cannot be different for both T and T', therefore they must be the same.

(b) Assuming T and T' described above, if the edge values are not distinct, picking the next smallest edge e may be different for T and T', allowing for the same weight, but with a different path.

Problem 4.) (a) This scenario makes it so G is not a complete graph. That means that the unique heaviest edge weight may be the only thing connecting a cut across G.

counterexample:



The MST must contain the heaviest edge.

(b) True. If T is a spanning tree that contains e, remove e. This will leave you with two components. Since e is within a cycle, there will have to be at least one more edge connecting the two components. Therefore, e is not used in a MST.

(c) True. Let T be a spanning tree that contains e, and another edge x that has the same weight as e. Since they are the same weight, T' that includes x and not e, will give the same value but with a different path. The edge with minimum weight will be used.

(d) True. Let there be a spanning tree T that includes e, and a spanning tree T' such that it does not include e, but e', which is another edge that makes the same cut as e. Since the weight of T' will be more than that of T, T' cannot be a minimal spanning tree. Therefore e must be included.

Problem 5.) In order to get another code that will lead to an encryption smaller than 58 bits for the phrase 'bananas are tasty', you can use the values shown in the example, but assigned differently. They will be assigned by length, shortest one going to the letter that appears most. These values are already the shortest that allow for no prefixes. The values are

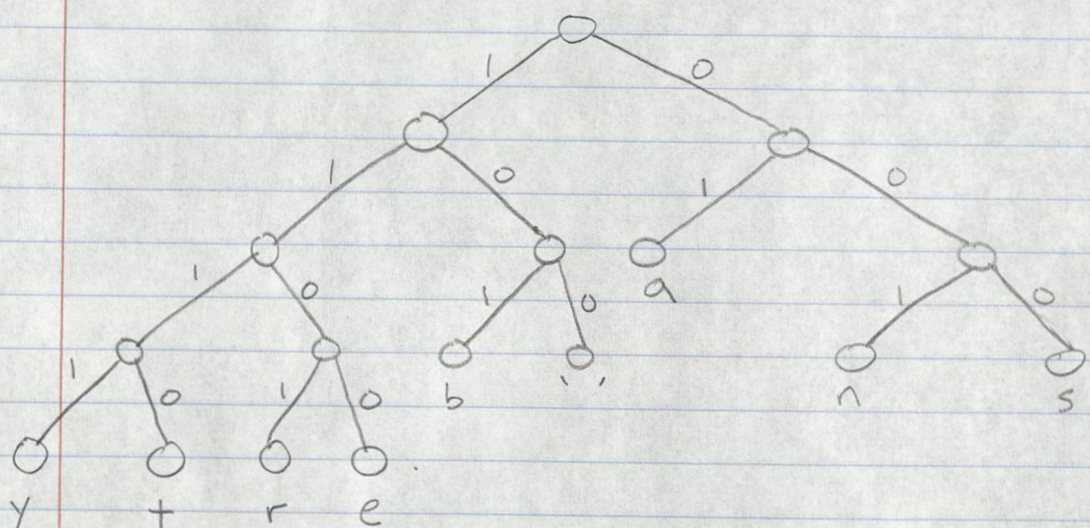
a: 01	' ': 100	r: 1101
n: 001	b: 101	t: 1110
s: 000	e: 1100	y: 1111

"bananas are tasty" encodes to

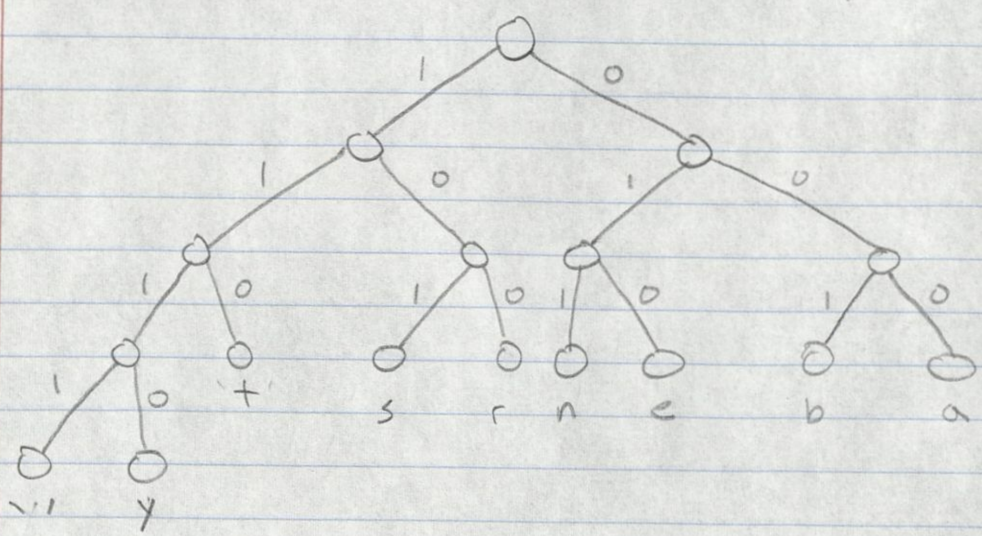
101010010100101000100011101110010011100100011101111

This set of codes for these letters yields an encryption that is 51 bits long for "bananas are tasty".

The code represented in a tree looks as such:



Assigning new values to the characters based on another tree yields to 54 bits used for the phrase.



Giving the values

a: 000	n: 011	t: 110
b: 001	r: 100	y: 1110
e: 010	s: 101	' ': 1111

This encodes "bananas are tasty in 54 bits.

001000011000 01100010111110001000101111110000101110 1110