EC330 Applied Algorithms and Data Structures for Engineers
Fall 2018

Homework 8

Out: December 1, 2018
Due: December 12, 2018

This homework has a written part and a programming part. Both are due at 8:59 pm on December 12. You should submit both parts on Blackboard. For the written part, you should submit a single PDF file containing either typeset answers or scanned copies of hand-written answers. Make sure you write your answers clearly. For the programming part, your code should be easy to read and understand, and demonstrate good code design and style. Your program must compile and run on the lab computers.

1. **Bipartite Graph [20 pt]**
   Describe a linear-time algorithm for determining whether a given undirected graph is bipartite. You can write your algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code. Justify your answer (make sure you describe the data structures that you use).

2. **Minimum Spanning Tree [20 pt]**
   Consider the following procedure for finding a minimum spanning tree for an undirected and connected graph G.

```
MAYBE-MST(G):
    while there is a cycle c in G:
        remove the maximum weight edge in c
```

   a. Does this algorithm always result in a minimum spanning tree of G? If not, provide a *small* graph as a counterexample. **[5 pt]**
   b. Provide an efficient algorithm for *implementing* `MAYBE-MST`. You can write your algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code. Analyze the running time of the algorithm (if the worst-case running time is different from the best-case, analyze both). **[15 pt]**

3. **Single-Source Shortest Path [20 pt]**
   Explain how to modify Bellman-Ford's algorithm to efficiently count the *number* of shortest paths (of equal weight) from a source vertex to every other vertex in the graph. You can write the modified algorithm in pseudo code similar to the ones I gave in recent lectures, or write actual C++ code.

4. **Programming [40 pt]**
For the programming part, make sure to write your name and BU ID in a comment at the top of the program, along with your collaborator's name and BU ID, if any. *While you are allowed to consult textbooks and web sources, you are supposed to write the answer on your own (with the exception that you may share code with your collaborator) and you should acknowledge all sources.* To use the compiler on the lab computers, run "*module load gcc*" first.

You are provided with a map representing characters and their frequencies, as well as a text file, `encoded.txt`, containing *Huffman encoded text*.

Implement *a Huffman class* that
- generates a Huffman tree from the provided map,
- prints the generated tree in table format (containing *Huffman Code* and corresponding *Character* columns),
- and then prints out the decoded (i.e. uncompressed) text version of the provided file.

To generate the Huffman tree, you should use a min-heap priority queue (from the Standard Library).

Your code should compile and run with the provided, unmodified, `main.cpp` file, to generate the following output:

```
Huffman Code    Character
--------------------------------
0               g
10              o
110             e
111             l

The decoded text is: googgle
```

Submit your code as a single `Problem3.zip` file consisting of your code (without `main.cpp`), along with a modified `makefile` that was used to compile it on the lab computers.