Homework 7

Out: 11.16.23
Due: 11.29.23

1. [Graphs 15 Points]
   For which integer n > 0 values do the following graphs have the given chromatic number? Explain.
   a. C_{2n}, chromatic number 2.
   b. W_{2n}, chromatic number 3.
   c. S_{2n} chromatic number 2.

2. [Graph representation, 15 Points]
   Consider directed graphs with V vertices and E edges.
   For each of the following graph representations, provide:
   - How long would it take (asymptotically, in terms of the number of vertices V and edges E) to compute both the in-degree and out-degree of every vertex in the graph. Explain.
   - How much additional space the above would require. Explain.
   a. Edge-list implemented as a linked list.
   b. Adjacency-list implemented as a linked list of linked lists.
   Include, for each vertex v, a list of vertices connected by an outgoing edge from v.
   c. Incidence matrix implemented as a two-dimensional array. The incidence matrix of a directed graph is a $n \times m$ matrix B where n and m are the number of vertices and edges respectively, such that $B_{i,j} = -1$ if the edge e_j leaves vertex v_i, 1 if it enters vertex v_i and 0 otherwise .

3. [Graph algorithms, 10 Points]
   Give a linear-time algorithm for determining whether a given graph is bipartite.

4. [Graph algorithms, 10 Points]
   Design an efficient algorithm that computes a path in a given connected, undirected graph G that traverses every edge in G exactly once in each direction. Analyze its runtime.

5. [Huffman Code, 50 points]
   You are provided with a map representing characters and their frequencies, in *main.cpp*, 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 Character and Huffman code columns), and then prints out the decoded (i.e. uncompressed) text version of the provided file.

   The *Huffman* class must include a min heap priority queue member, which will be utilized to generate the Huffman tree, as well as the following methods:

a. buildHuffmanTree – accepts an array of characters, an array of integer frequencies, and the integer size of these arrays, and constructs a Huffman Tree from the input.
b. printCodes – prints the Huffman tree in a table format.
c. decodeText – accepts a pointer to a file name containing a Huffman code encoded text, and prints out the plain text version of the file.

Your code should be able to compile and run with the provided *main.cpp* file to the to generate the following output, consisting of the Huffman code (character separated by one space followed by the Huffman code, in tree order), and the decoded text:

```
g 0
o 10
e 110
l 111

googgle
```

Note that for consistency, you should follow the ordering as described in class:

- When building a tree, select the two trees with smallest frequencies from the heap, T_1 and T_2. The new tree, T, will have left subtree T_1 and right subtree T_2, and be assigned a frequency $f(T_1)+f(T_2)$.
- When inserting the new tree T to the priority queue, it should be inserted behind existing nodes with the same frequency.

The provided *makefile* assumes that your code is implemented in *Huffman.cpp* (and *Huffman.h*) files. If you use additional files, you will need to modify the *makefile* accordingly. To compile on the lab computers, type the following two commands:

```
> scl enable devtoolset-10 bash
> make
```

Your code will be tested with other *main.cpp* files and others text files. You may assume that the provided frequency and character arrays are already sorted in increasing frequency order.

Submit your code as a single *Problem5.zip* file consisting of your code (without main.cpp), along with a modified *makefile* that was used to compile it on the lab computers.