CS2800 Design and Analysis of Algorithms
Problem Set #7 March 13, 2023

1. Let $\mathcal{S} = \{a, b, c\}$ be an alphabet with frequencies $f[a]$, $f[b]$ and $f[c]$. Which of the following codes are valid Huffman codes, and why? If they are valid, give values for $f[a]$, $f[b]$ and $f[c]$ that will lead to that particular Huffman code.
   (a) $C(a) = 0, C(b) = 10, C(c) = 11$.
   (b) $C(a) = 0, C(b) = 1, C(c) = 00$.
   (c) $C(a) = 10, C(b) = 01, C(c) = 00$.

2. Prove the following two properties of Huffman codes.
   (a) Show that if there is a letter in an alphabet with frequency more than $2/5$, then any Huffman code for that alphabet contains a letter that is encoded by a single bit.
   (b) Show that if every letter in an alphabet has frequency less that $1/3$, then for any Huffman code for that alphabet, there is no letter that is encoded by a single bit.

3. Let $G(V, E)$ be a weighted graph with a weight function $w : E \to \mathbb{R}$. Instead of the minimum spanning tree, suppose that we are interested in the maximum spanning tree which is a spanning tree with the maximum weight among all spanning trees. Does this problem have a greedy algorithm? If yes, design the algorithm and prove its correctness. If no, then explain why.

4. We will see an alternate algorithm to find an MST in a connected undirected graph $G(V, E)$ using the following observation.
   (a) Prove the following statement: Let C be any cycle in the graph $G(V, E)$, and let e be the heaviest edge in C. Then there is an MST that does not contain e.
   (b) How will you use the above observation to design an algorithm to construct an MST?
   (c) What is the running time of your algorithm?

5. Suppose that you are given a graph $G(V, E)$ with edge weights $w : E \to \mathbb{R}$ and an MST T of G. An update is given where an edge $e \in E$ has its weight modified from $w(e)$ to $w(e)'$. Describe an efficient algorithm to update the MST T to obtain a new MST. Clearly explain why your algorithm is correct.

6. Design an algorithm to find the second smallest (in weight) spanning tree in a graph $G(V, E, w)$.

   If you are given G and an MST T, then there is a linear-time algorithm to get the second smallest spanning tree. For this exercise, any polynomial-time algorithm will suffice.

   Hint: The second smallest spanning tree will differ from an MST in only one edge. Prove this and use it to design your algorithm.

7. In this question, we will try to show that for certain classes of graphs, like planar graphs, Boruvka's algorithm actually runs in time $O(n)$. To that end, let's look at a slightly modified version of the algorithm.

   First, for a weighted graph $G(V, E, w)$, a *contraction* of an edge $e = (u, v)$ (denoted by G/e) gives a new graph where the vertices u and v are replaced by a new vertex, all edges incident on u or v are now incident on the new vertex, and the parallel edges to the new vertex is replaced by the minimum-weight edge incident on it.

   Now, we can think of Boruvka's algorithm as first finding the minimum weight edges in each step, contracting all of them to create a new graph and finding an MST in the new graph G'.

   (a) Show that if T' is the MST of G', and $G' = G/e$, then $T' \cup \{e\}$ is the MST of G.

   Hint: This is very similar in nature to the recursive construction we saw for Huffman coding in class. Use similar ideas to prove the statement.

   (b) Show that given an edge e, we can construct the adjacency list representation of G/e in time $O(n + m)$ where n is the number of vertices in G and m is the number of edges.

   Call a class of graphs *nice*[1] if the following conditions hold:

   - For any edge e, G/e is nice.
   - A nice graph G on n vertices has $O(n)$ edges.

   (c) Use Parts (a) and (b) to show the MST of a nice graph G on n vertices can be constructed using Boruvka's algorithm in $O(n)$ time.

[1] For instance, planar graphs are nice.