CSE 101 Spring 2024 Homework 5 Due: Thursday 5/16 at 11:59pm

1. Consider the following divide and conquer algorithm that claims to find an MST when the input is a complete graph G with positive edge weights:

   Algorithm Description: Given an undirected complete graph $G = (V, E)$ with positive edge weights where $V = [v_1, \ldots, v_n]$,

   - If $n = 1$ then return the empty set of edges.
   - Otherwise, split the set of vertices into two sets: $V' = [v_1, \ldots, v_{\lfloor n/2 \rfloor}]$ and $V'' = [v_{\lfloor n/2 \rfloor}+1, \ldots, v_n]$.
   - Create two new graphs $G' = (V', E')$ and $G'' = (V'', E'')$ where $E' \subseteq E$ is the set of edges with both endpoints in V' and $E'' \subseteq E$ is the set of edges with both endpoints in V''.
   - Recursively run the algorithm on G' and G'' to get T' and T'', respectively. Find the lightest edge that connects a vertex in T' to a vertex in T'' and call that edge e.
   - Return $T' \cup T'' \cup \{e\}$.

   Disprove the correctness of this algorithm by giving a counterexample. (8 points)

2. (14 points) You are given an increasing sequence of integers: $(A[1], A[2], \ldots, A[n])$. Design an algorithm that determines (returns TRUE or FALSE) if there exists an index i such that $A[i] = i$.

   Your algorithm should run in $O(\log n)$ time.

   (6 points Algorithm Description. A high level description is necessary. You may include an implementation level or pseudocode if you think it will help with understanding your algorithm.) (4 points Justification.)

   (4 points for runtime analysis.)

3. (12 points)

   You are given a list of n ordered pairs $[(x_1, f_1), \ldots, (x_n, f_n)]$. This list describes a list of length $\sum f_i$ that contains f_1 copies of the value x_1, f_2 copies of the value x_2 and so on.

   You wish to find the median value of this list in expected runtime of $O(n)$. (You can assume that $\sum f_i$ is odd.)

4. (12 points)

   (a) Let $T(n)$ be the runtime of a divide and conquer algorithm on an input of size n. The algorithm has 6 recursive calls each of size $n/4$ and the non-recursive part takes $O(n^{1.5})$ time. Use the Master theorem to find the best Big-Oh runtime.

   (b) Let $R(n)$ be the runtime of a divide and conquer algorithm on an input of size n. The algorithm has 1 recursive call of size $n/2$ and the non-recursive part takes $O(\log n)$ time. Find the best Big-Oh runtime.

   (c) Let $S(n)$ be the runtime of a divide and conquer algorithm on an input of size n. The algorithm has 2 recursive calls each of size $2n/3$ and the non-recursive part takes $O(n)$ time. Find the best Big-Oh runtime.