Homework 4

Course: Algorithm Design and Analysis — **Semester**: Spring 2024

Instructor: Shi Li — **Due Date**: **2024/5/5**

Student Name: _________ Student ID: _________

Problems	1	2	3	4	5	Total
Max. Score	20	20	20	20	20	100
Your Score						

Problem 1. Suppose we are given an undirected graph $G = (V, E)$, with non-negative edge weights $(w_e)_{e \in E}$. Let T be the minimum spanning tree of G, and let T' be any spanning tree of G. Suppose we sort the $n-1$ edge weights of T in ascending order to obtain $y_1 \leq y_2 \leq \cdots \leq y_{n-1}$, and we sort the $n-1$ edge weights of T' in ascending order to obtain $y'_1 \leq y'_2 \leq \cdots \leq y'_{n-1}$.

Prove the following statement: for every $i \in [n-1]$, we have $y_i \leq y'_i$.

Problem 2. We are given a connected undirected graph $G = (V, E)$ with non-negative edge weights $(w_e)_{e \in E}$. Design an $O(n \log n + m)$-time algorithm to decide if the minimum spanning tree of G is unique or not.

Problem 3. We are given a connected undirected graph $G = (V, E)$ with non-negative edge weights $(w_e)_{e \in E}$. We are also given two vertices $s, t \in V$. Design an $O(n \log n + m)$-time algorithm to decide if the shortest path from s to t in G is unique or not.

Problem 4. Consider the minimum cost arborescence problem on the directed graph $G = (V, E)$ with non-negative edge weights $(w_e)_{e \in E}$ and a specified root r. Let C be a 0-cost simple cycle in G that does not contain r. Prove the statement that we skipped in class: there exists a minimum cost arborescence T in G (rooted at r) that includes all but one edge of C.

Problem 5. This problem asks you to find the largest rectangle in a histogram given an array A of n non-negative integers. For instance, with the input array $(3, 5, 10, 11, 20, 4, 8, 10)$, the largest rectangle has an area of 30, achieved by a rectangle of height 10 spanning from column 3 to column 5 (see Figure 1).

Additionally, you are given a sorted array of indices in $[n]$ according to the heights of the bars in the histogram; that is, a permutation $(i_1, i_2, \ldots, i_n)$, of n with $A[i_1] \leq A[i_2] \leq \ldots \leq A[i_n]$. You need to use the union-find data structure to design an algorithm that solves this problem in $O(n\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackermann function.

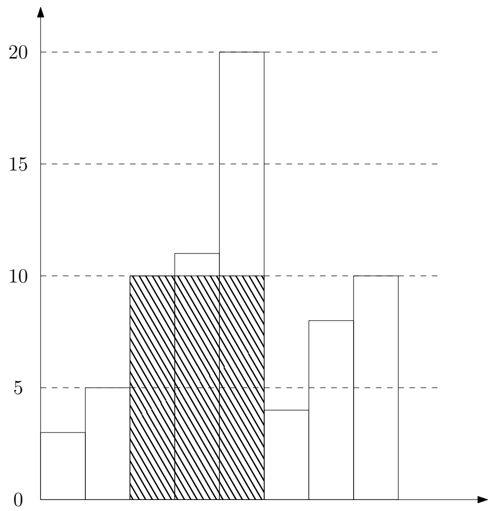

Figure 1: The largest rectangle of in the histogram is given by the shaded area.