Homework 3

Course: Algorithm Design and Analysis **Semester**: Spring 2025

Instructor: Shi Li **Due Date**: **2025/5/11**

Student Name: _____ Student ID: _____

Problems	1	2	3	4	5	6	Total
Max. Score	20	15	15	15	15	20	100
Your Score							

- Remark: In an algorithm design problem, you need to prove the correctness of the algorithm and show that it achieves the required running time. However, if the running time and/or correctness are easy to see, brief explanations are sufficient. You can use give the answers in either Chinese or English.

Problem 1. Given an array A of n numbers, we say that a 10-tuple $(i_1, i_2, \cdots, i_{10})$ of integers is inverted if $1 \leq i_1 < i_2 < i_3 < \cdots < i_{10} \leq n$ and $A[i_1] > A[i_2] > A[i_3] > \cdots > A[i_{10}]$.

(1a) Give an $O(n^2)$-time algorithm to count the number of inverted 10-tuples w.r.t A.

(1b) Give an $O(n \lg n)$-time algorithm to count the number of inverted 10-tuples w.r.t A.

Problem 2. There are n balloons in a row. Each balloon is painted with a positive integer number on it. You are asked to burst all the balloons.

If you burst the i-th remaining balloon in the row, you will get $nums[i-1] \times nums[i] \times nums[i+1]$ coins, where $nums[t]$ for any t is the number on the t-th remaining balloon in the row. If $i - 1 = 0$ or $i + 1$ is more than the number of balloons, then treat it as if there is a balloon with a 1 painted on it.

Return the maximum coins you can collect by bursting the balloons wisely. For example, if there are initially 4 balloons in the row with numbers 3,1,5,8 on them. Then the maximum coins you can get is 167. This is how the array of numbers change when you burst the balloons: $(3, 1, 5, 8) \to (3, 5, 8) \to (3, 8) \to (8) \to ()$. The coins you get is $3 \times 1 \times 5 + 3 \times 5 \times 8 + 1 \times 3 \times 8 + 1 \times 8 \times 1 = 167$.

Design an $O(n^3)$-time algorithm to solve the problem. For convenience, you only need to output the maximum number of coins you can get.

Problem 3. This problem asks you to find the largest rectangle in a histogram, given to you as an array of n numbers. For example, if the input is $(3, 5, 10, 11, 20, 4, 8, 10)$, then the largest rectangle has size 30 (with height 10 and width 3, covering column 3 to column 5.)

In Homework 2, you designed a divide-and-conquer algorithm with running time $O(n \log n)$. In this homework, you need to use the union-and-find data structure to design an $O(n\alpha(n))$ time algorithm, where $\alpha(\cdot)$ is the inverse Arckerman function.

Problem 4. Consider the minimum spanning tree problem over the graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}_{\geq 0}$. Assume all the weights are distinct. Let T be the minimum spanning tree of $G = (V, E)$ w.r.t the weight function w. Prove the following statements:

(2a) An edge $e \in E$ is in T if and only if there is a proper cut $(U \subsetneq V, V \setminus U)$ of G such that e is the lightest edge in E between U and $V \setminus U$.

(2b) An edge $e \in E$ is not in T if and only there is a simple cycle C in G on which e is the heaviest edge.

Problem 5. We are given a directed graph $G = (V, E)$ with positive weight function: $w : E \to \mathbb{R}_{>0}$, and two vertices $s, t \in V$. Suppose we have already computed the d and π arrays using the Dijkastra's algorithm: $d[v]$ is the length of the shortest path from s to v, and $\pi[v]$ is the vertex before v on the path.

Show how to use the d and π array to count the number of shortest paths from s to t in $O(n \log n + m)$ time. You do not need to worry about the integer overflow issue. That means, you assume a word can hold a very big integer, and basic operations over these big integers take $O(1)$ time.

Problem 6. Let $G = (V, E)$ be a directed graph and let $w : E \to \mathbb{R}$ be a weight function on the edges. There can be negative weights on G, but it is promised that there are no negative cycles in G.

(6a) Show that there is a function $a : V \to \mathbb{R}$ such that for every edge $(u, v) \in E$ we have $a(u) - a(v) + w(u, v) \geq 0$.

(6b) Give an $O(mn)$-time algorithm to find the function a.

(6c) Show how to use the algorithm in (6b), to design an $O(n(n \log n + m))$-time algorithm to compute the lengths of shortest paths between all pairs. Recall that the Floyd-Warshall algorithm takes $O(n^3)$ time.