CSE 431/531: Algorithm Analysis and Design — Spring 2019

Homework 2

Instructor: Shi Li — **Deadline: 3/18/2019**

Your Name: ____________________ Your Student ID: _______________

Problems	1	2	3	4	Total
Max. Score	10	15	15	40	80
Your Score					

Problem 1 (10 points) Consider the minimum spanning tree problem, where the input is a graph $G = (V, E)$ and a weight vector $w : E \to \mathbb{R}_{\geq 0}$. For simplicity, we assume all the weights are different. Decide if each of the following strategy is safe or not.

(1a) Let C be a cycle in G and e^* be the heaviest edge on C. Then we do not include e^* in the spanning tree. (So, in the residual problem, we remove e^* from G.)

(1b) Let C be a cycle in G and e^* be the lightest edge on C. Then we include e^* in the spanning tree. (So, in the residual problem, we contract e^* in G.)

(1c) Let $U \subsetneq V, U \neq \emptyset$ be a strict non-empty subset of V and e^* be the lightest edge in E that is between U and $V \setminus U$. Then we include e^* in the spanning tree.

(1d) Let $U \subsetneq V, U \neq \emptyset$ be a strict non-empty subset of V and e^* be the heaviest edge in E that is between U and $V \setminus U$. Then we do not include e^* in the spanning tree.

If your answer is "not safe" for a strategy, you need to give a counter example. (For safe strategies, saying they are safe is sufficient.)

Problem 2 (15 points) Given a set of n jobs $\{1, 2, 3, \cdots, n\}$, each job j with a processing time $t_j > 0$ and a weight $w_j > 0$, we need to schedule the n jobs on a machine in some order. Let C_j be the completion time of j on in the schedule. Then the goal of the problem is to find a schedule to minimize the weighted sum of the completion times, i.e, $\sum_{j=1}^{n} w_j C_j$.

Example. Suppose there are two jobs: the first takes time $t_1 = 1$ and has weight $w_1 = 10$, while the second job takes time $t_2 = 3$ and has weight $w_2 = 2$. Then doing job 1 first would yield a weighted completion time of $10 \cdot 1 + 2 \cdot 4 = 18$, while doing the second job first would yield the larger weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$.

Design an efficient greedy algorithm to solve the problem.

Problem 3(15 points) In the interval covering problem, we are given n intervals $[s_1, t_1), [s_2, t_2), \cdots, [s_n, t_n)$ such that $\bigcup_{i \in \{1,2,3,\cdots,n\}} [s_i, t_i) = [0, T)$. The goal of the problem is to return a smallest-size set $S \subseteq \{1, 2, \cdots, n\}$ such that $\bigcup_{i \in S} [s_i, t_i) = [0, T)$. Design an efficient greedy algorithm for this problem. You do not need to optimize the running time. So you can simply use the two-step proof:

(1) Give a simple greedy strategy, and prove it is safe.

(2) Show that after you made a decision using the strategy, the residual task can be formulated again as an instance of the interval covering problem.

Problem 4(40 points) You need to implement the union-and-find data structure. In this problem, there are n elements numbered from 1 to n. Initially, each element is in a separate partition. You will receive a sequence of operations, each being one of the following:

- query-size(i). This operation asks for the size of the partition containing i, i.e, the number of elements in the partition containing i. It does not change the partitioning.
- merge(i, j). This operation will merge the two partitions containing i and j. If i and j were already in the same partition, the operation does nothing.

You need to implement the union-and-find data structure that supports the two operations using C++, Java or Python.

Implementation of the algorithm You need to read from the standard input (i.e, the terminal) and output to the standard output (i.e, the screen).

- **Input format**: In the first line of the input, there are two positive integers n and m. n is the number of elements and m is the number of operations given to you. The elements are numbered from 1 to n. You can assume that $1 \leq n \leq 10000$ and $1 \leq m \leq 100000$. In the next m lines, each line is either of the form $Q\ i$, where i is a number between 1 and n, or of the form $M\ i\ j$, where i and j are two different numbers between 1 and n. $Q\ i$ corresponds to the operation query-size(i), and $M\ i\ j$ corresponds to the operation merge(i, j).
- **Output format**: For every query(i) operation, output a single number in a line that answers the query: i.e, output the number of elements in the partition that contains i.

Input:	**Output**:	**Explanation**:
5 7	2	Initial : $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
M 1 5	3	M 1 5 : $\{1, 5\}, \{2\}, \{3\}, \{4\}$
Q 1	5	Q 1 : returns 2
M 5 4		M 5 4 : $\{1, 4, 5\}, \{2\}, \{3\}$
M 2 3		M 2 3 : $\{1, 4, 5\}, \{2, 3\}$
Q 4		Q 4 : returns 3
M 1 2		M 1 2 : $\{1, 2, 3, 4, 5\}$
Q 5		Q 5 : returns 5