Fall 2019 B503 Homework 2

Due date: September 29, 11:59PM Lecturer: Qin Zhang

Your Name:_____ Your University ID:_____

Instruction: Please submit your homework solution **before due date, via Canvas**. Homework solution must be typesetted in PDF format via LaTeX (preferred) or Word. Please add references to ALL the resources you have used for completing the assignment. You are allowed to discuss the assignment with other students, and if you do so, please list their names in the submission.

Due: September 29, Sunday, 11:59PM

Total points: 80

Late Policy: No extensions or late homeworks will be granted, unless a request is made to the course instructor before due date and written documents are provided to support the reason for late submission.

Problem 1 (10 points).

1. In Interval Scheduling problem, if each request not only has a starting time and a finishing time but also has weight, and we want to maximize the total weights of intervals we selected. Can we still use the greedy algorithm that we learned during the lecture (process according to the finishing time)? If yes give a proof; if no give an counterexample.

2. Give a counter example showing that Dijkstra's algorithm does not work for graphs with negative edges. Please simulate the running of Dijkstra's algorithm on your counter example and explain why Dijkstra's algorithm fails in this case.

Problem 2 (10 points). During the lecture, we learned a greedy algorithm to solve the Scheduling All Intervals problem. We also learned that a naive implementation of the greedy algorithm would take $O(n^2)$ time. Please describe a more efficient implementation of the greedy algorithm with $O(n \log n)$ running time?

Problem 3 (10 points). On Thanksgiving day, you arrive on an island with n turkeys. You've already had thanksgiving dinner, so you don't want to eat the turkeys, but you do want to wish them all a Happy Thanksgiving.

However, the turkeys each have very different sleep schedules. Turkey i is awake only in a single closed interval $[a_i, b_i]$. Your plan is to stand in the center of the island and say loudly "Happy Thanksgiving!" at certain times $t_1, \ldots, t_m$. Any turkey who is awake at one of the times t_j will hear the message. It's okay if a turkey hears the message more than once, but you want to be sure that every turkey hears the message at least once.

Design a greedy algorithm which takes as input the list of intervals $[a_i, b_i]$ and outputs a list of times $t_1, \ldots, t_m$ so that m is as small as possible and so that every turkey hears the message at least once. Your algorithm should run in time $O(n \log n)$. Prove that your algorithm is correct.

Problem 4 (10 points). Consider the following two statements:

1. The first k edges chosen by Kruskal's algorithm have the following property: there is no cheaper acyclic subgraph of G with k edges. (Assume edge costs are distinct)

2. The first k edges chosen by Prim's algorithm (starting from some "root node" r) have the following property: there is no cheaper connected subgraph of G containing the node r along with k other nodes. (Assume edge costs are distinct)

Give a proof if the statement is true or a counterexample if the statement is false.

Problem 5 (10 points). Let G be a connected weighted undirected graph. In class, we defined a minimum spanning tree of G as a spanning tree T of G which minimizes the quantity

$$X = \sum_{e \in T} w_e,$$

where the sum is over all the edges in T, and w_e is the weight of edge e.

Define a "minimum-maximum spanning tree" to be a spanning tree that minimizes the quantity

$$Y = \max_{e \in T} w_e.$$

That is, a minimum-maximum spanning tree has the smallest maximum edge weight out of all possible spanning trees.

(a) Prove that a minimum spanning tree in a connected weighted undirected graph G is always a minimum-maximum spanning tree for G.

(b) Show that the converse to part (a) is not true. That is, a minimum-maximum spanning tree is not necessarily a minimum spanning tree.

Problem 6 (10 points). One of the first things you learn in calculus is how to minimize a differentiable function such as $y = ax^2 + bx + c$, where $a > 0$. The Minimum Spanning Tree Problem, on the other hand, is a minimization problem of a very different flavor. One can ask what happens when these two minimization issues are brought together, and the following question is an example of this.

Suppose we have a connected graph $G = (V, E)$. Each edge e now has a time varying edge cost given by a function $f_e : R \to R$. Thus, at time t, it has cost $f_e(t)$. We will assume that all these functions are positive over their entire range. Observe that the set of edges constituting the minimum spanning tree of G may change over time. Also, of course, the cost of the minimum spanning tree of G becomes a function of the time t; we will denote this function $c_{G(t)}$. A natural problem then becomes: find a value of t at which $c_{G(t)}$ minimized.

Suppose each function f_e is a polynomial of degree 2: $f_{e(t)} = a_e t^2 + b_e t + c_e$, where $a_e > 0$. Give an algorithm that takes the graph G and the values $(a_e, b_e, c_e) : e \in E$ and returns a value of the time t at which the minimum spanning tree has minimum cost. Your algorithm should run in time polynomial in the number of nodes and edges of the graph G. You may assume that arithmetic operations on the numbers (a_e, b_e, c_e) can be done in constant time per operation.

Problem 7 (10 points). We are given two arrays of n points on the number line: red points $r_1, r_2, \ldots, r_n \in R$ and blue points $b_1, b_2, \ldots, b_n \in R$. You may assume that all red points are distinct and all blue points are distinct.

We want to pair up red and blue points in the following way: find a bijection $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ so that r_i is matched with $b_{\pi(i)}$. Note that each red point is matched with a unique blue point and vice versa.

We would like to minimize the following objective function:

$$\sum_{i=1}^{n} |r_i - b_{\pi(i)}|$$

Design an $O(n \log n)$ time algorithm that finds the matching (bijection) that minimizes the sum of distances between the matched points. Prove the correctness of the algorithm and analyze its running time.

Problem 8 (10 points). Jane loves tomatoes! She eats one tomato every day, because she is obsessed with the health benefits of the potent antioxidant lycopene and because she just happens to like them very much.

The price of tomatoes rises and falls during the year, and when the price of tomatoes is low, Jane would naturally like to buy as many tomatoes as she can. Because tomatoes have a shelf-life of only d days, however, she must eat a tomato bought on day i on some day j in the range $i \leq j < i + d$, or else the tomato will spoil and be wasted. Thus, although Jane can buy as many tomatoes as she wants on any given day, because she consumes only one tomato per day, she must be circumspect about purchasing too many, even if the price is low.

Jane's obsession has led her to worry about whether she is spending too much money on tomatoes. She has obtained historical pricing data for n days, and she knows how much

she actually spent on those days. The historical data consists of an array $C[1, \ldots, n]$, where $C[i]$ is the price of a tomato on day i. She would like to analyze the historical data to determine what is the minimum amount she could possibly have spent in order to satisfy her tomato-a-day habit, and then she will compare that value to what she actually spent.

Give an $O(n)$ time algorithm to determine the optimal offline purchasing strategy on the historical data. Given d, n, and $C[1, \ldots, n]$, your algorithm should output $B[1, \ldots, n]$, where $B[i]$ is the number of tomatoes to buy on day i. Please also prove the correctness of your algorithm. You may get partial credits with slower algorithm.