St. Francis Xavier University
Department of Computer Science

CSCI 355: Algorithm Design and Analysis
Assignment 2
Due October 3, 2024 at 11:30am

Assignment Regulations.

- This assignment must be completed individually.
- Please include your full name and email address on your submission.
- You may either handwrite or typeset your submission. If your submission is handwritten, please ensure that the handwriting is neat and legible.

[4 marks] 1. Suppose we are given a connected graph $G = (V, E)$ and a specific vertex $u \in V$. Using our knowledge of graph search algorithms, we construct a depth-first search tree rooted at u that contains all vertices of G, and we call this tree T. We then construct a breadth-first search tree again rooted at u, and we happen to obtain the same tree T.

Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not also belong to T.)

[8 marks] 2. Consider the following set of intervals:

Interval	Start	End
I_1	0	2
I_2	1	4
I_3	2	7
I_4	3	6
I_5	5	9
I_6	6	9
I_7	8	10
I_8	10	12
I_9	11	12
I_{10}	12	13

(a) Using the earliest-finish-time-first algorithm, determine a schedule that maximizes the subset of mutually compatible intervals from the given set of intervals.

(If you like, you may implement the earliest-finish-time-first algorithm in the programming language of your choice, but your answer must include a copy of your source code and a verbose listing of each step of your implementation's output.)

(b) Consider the following optimal interval scheduling obtained from the given set of intervals:

$$I_1, I_3, I_7, I_9, I_{10}.$$

Using an exchange argument, transform this solution to the one you obtained using the greedy earliest-finish-time-first algorithm in part (a). Show each step of the transformation.

[4 marks] 3. Consider the following variant of the interval scheduling problem: we are given a set of intervals I where each interval I_j has an associated start time $s_j \geq 0$, finish time $f_j > s_j$, and value $v_j \geq 0$. Our goal is to determine a schedule that maximizes the value of the chosen subset of mutually compatible intervals.

Suppose we apply the following strategy to this variant problem:

> Sort the intervals from greatest to least value-to-total-time ratio, then choose intervals greedily as long as they are mutually compatible with the current schedule.

Unfortunately, this strategy is suboptimal. Give a counterexample where this strategy produces a worse schedule than the optimal solution.

[9 marks] 4. The Canadian Atlantic Railway company is building a new railroad across Nova Scotia, but they have not settled on locations for stations yet. Suppose that the railroad is a single line with markers lying along this line at each kilometre. Some towns are located at certain kilometre markers, while all other kilometre markers lie between towns. (You may assume that all kilometre markers are natural numbers.)

The company approaches you to help them figure out where to place stations in such a way that the total number of stations constructed is minimized.

Every station must be placed at a kilometre marker. Moreover, not all stations need to be placed in towns, but each town must be within R kilometres of some station. Say that a train station S *covers* a town T if (i) the town T is within R kilometres of the station S, and (ii) the town T is not already covered by some other station.

(a) The company's former consultant, who was educated at the Dalhousie Railway School, previously suggested the following station layout approach:

> Until all towns are covered, repeatedly construct a station where you can maximize the number of towns covered.

However, this approach is suboptimal (hence, why the consultant is a *former* consultant).

To see why, determine how many stations would be constructed following this approach if towns were located at kilometres $\{0, 4, 6, 8, 10\}$ and $R = 2$. Then, determine the optimal number of stations for this instance.

(b) Design a greedy algorithm that solves this problem. Your greedy algorithm must run in polynomial time, and you should provide brief justifications for your algorithm's correctness, optimality, and runtime.

You may describe your algorithm in either plain English or pseudocode.