CS:3330 Greedy Algorithms Practice Problems, Spring 2018

1. You are given a set $X = \{x_1, x_2, \ldots, x_n\}$ of points on the real line. Your task is to design a *greedy* algorithm that finds a smallest set of intervals, each of length 2, that contains all the given points.

   Example: Suppose that $X = \{1.5, 2.0, 2.1, 5.7, 8.8, 9.1, 10.2\}$. Then the three intervals $[1.5, 3.5]$, $[4, 6]$, and $[8.7, 10.7]$ are length-2 intervals such that every $x \in X$ is contained in one of the intervals. Note that 3 is the minimum possible number of intervals because points 1.5, 5.7, and 8.8 are far enough from each other that they have to be covered by 3 distinct intervals. Also, note that my solution is not unique – for example, I can shift the middle interval $[4, 6]$ to the right, say to $[5.7, 7.7]$, without disturbing the other intervals, and we would still have an optimal solution.

   (a) Suppose that elements of X are presented in increasing order. Describe (using pseudocode) a greedy algorithm, running in $O(n)$ time, for this problem.

   (b) Using the approach that we used for the proof of correctness of the *Interval Scheduling* greedy algorithm prove that your algorithm indeed produces an optimal solution. Your proof needs to be clear and precise, in addition to being correct.

2. A variant of the *Interval Scheduling* problem is one in which each interval has an associated non-negative weight. In this problem (called the *Weighted Interval Scheduling* problem), we want to find a set of mutually non-overlapping intervals that have the maximum total weight. For example, consider intervals $I_1 = [1, 3]$, $I_2 = [2, 4]$, and $I_3 = [3.5, 4.5]$ and suppose that $w(I_1) = w(I_3) = 1$ and $w(I_2) = 10$. Then, the optimal solution to this problem would be $\{I_2\}$ and not $\{I_1, I_3\}$ because the weight of I_2 is 10 whereas the weight of $\{I_1, I_2\}$ is $1 + 1 = 2$.

   (a) The greedy algorithm that we used to solve the Interval Scheduling problem repeatedly picked an interval with earliest finish time and deleted other intervals that overlapped the selected interval. Show that this algorithm does not produce an optimal solution to the Weighted Interval Scheduling problem.

   (b) What about an algorithm that repeatedly picks a heaviest interval from all that are available and then deletes other intervals that overlap with the chosen interval? Does this algorithm always produce an optimal solution? If yes, then prove correctness of the algorithm and if no, then construct a counter-example.