CS 4349.003 Homework 5

Due Wednesday October 9th on eLearning

October 2, 2019

Please answer the following **2** questions.

To give everybody a bit a breather, **there will be no late penalty for submissions made before 1:00pm on Friday, October 11th**. As usual, submissions after 1:00pm on Friday may not be accepted so we can release solutions. The normal penalties will be applied for other homework assignments after this one.

1. (a) The greedy class scheduling algorithm we discussed in class is not the only greedy strategy we could have tried. Using an exchange argument, prove there exists a conflict-free schedule of maximum size which contains the class that *starts last*. You must give the full proof, but feel free to use Erickson Lemma 4.3 for reference.

   For parts (b) and (c), consider a weighted version of the class scheduling problem, where different classes offer a different number of credit hours (totally unrelated to the duration of the class lectures). Your goal is now to choose a set of non-conflicting classes that give you the largest possible number of credit hours. You may assume you're given three arrays $S[1..n]$, $F[1..n]$, and $C[1..n]$ so that each class i starts at time $S[i]$, ends at time $F[i]$, and is worth $C[i]$ credit hours.

   (b) Prove that the greedy algorithm suggested above—choose the class that starts last and recurse—does *not* always return an optimal solution for the weighted version of the problem. Your proof should describe a very small set of classes for which the optimal schedule is not given by this greedy algorithm.

   (c) Describe and analyze a dynamic programming algorithm that always computes an optimal schedule.

   [Hint: Sort the classes by starting time. Now decide: should I take that first class or not? If so, which other classes are still available to take?]

2. Let X be a set of n intervals on the real line (i.e., each element of X is a contiguous subset of the real numbers). A *proper coloring* of X assigns a color to each interval, so that any two overlapping intervals are assigned different colors. Describe and analyze an efficient algorithm to compute the minimum number of colors needed to properly color X. Assume that your input consists of two arrays $L[1 .. n]$ and $R[1 .. n]$, representing the left and right endpoints of the intervals in X. In other words, interval i starts at position $L[i]$ and ends at position $R[i]$. If you use a greedy algorithm (which you probably should), extra emphasis will be placed on the proof of correctness.

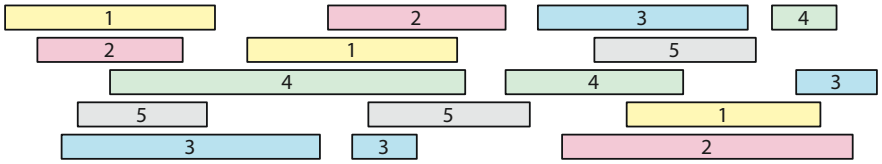

Figure 1. A proper coloring of a set of intervals using five colors.

[Hint: Sort the intervals by their left endpoints and pick colors for them in that order.]