Problem Set 5
Due: Friday, April 20. There is no class this day (Spring Symposium) so submit the hard-copy to my office by 10:00am.

[25] Problem 1. Activity Selection
This is problem #3 from Problem Set 4.
Consider a variation of the Activity Selection problem. As in the original problem, we have \(n\) activities to schedule and each activity has a start time \(s_i\) and a finish time \(f_i\), and we cannot schedule two activities if they overlap. However, now each activity has a weight \(w_i\) which is the profit you get for scheduling activity \(a_i\). Our goal is to find a set \(S\) of non-overlapping activities which have maximum total weight. You may assume all the weights are distinct.

(a) \([10\ \text{points}]\) Consider the following greedy strategy:

(1) sort the activities so \(w_1 > w_2 > \ldots > w_n\).
(2) for \(i = 1\) to \(n\)

Put activity \(a_i\) into \(S\) unless it overlaps an activity already in \(S\).

Give an example (by providing values for \(s_i\), \(f_i\), and \(w_i\)) where this strategy does not yield an optimal solution.

(b) \([15\ \text{points}]\) Now suppose each activity has length 1 (so \(f_i = s_i + 1\)) and the start (and therefore end times) are integers. Formally prove that the algorithm of part (a) finds an optimal solution.

(a) \([12\ \text{points}]\) Recall the recursive non-memoized version of Coin-Changing that we discussed on the first day of class. Although this algorithm always produces the correct answer (i.e. the minimum number of coins needed to make change for the given amount), for any set of coin denominations, it has an exponential running time. Formally prove that the running time of this algorithm is \(O(m^n)\), where \(m\) is the number of coin denominations and \(n\) is the amount for which to make change. Hint: Consider the recurrence tree that is generated by this algorithm.

(b) \([13\ \text{points}]\) Prove that the greedy approach of always choosing the coin with the largest denomination yields the optimal solution if the coins are in U.S. denominations. Use the standard proof technique we discussed in class – i.e. prove (1) the greedy choice property and (2) the optimal substructure property. For simplicity, you may assume that \(n \geq 25\).

(a) \([6\ \text{points}]\) Give a Huffman code for the input string:

ABAACAABAACACAABACABAACA

Show the tree and the actual binary encoding of the string. If two characters may be assigned the same code, break ties alphabetically, i.e. assign the lower valued code to the character that appears alphabetically first.

(a) \([10\ \text{points}]\) Let \(k\) denote the number of character types in a file to be encoded (e.g. for the string above, the character types are A, B, and C, so \(k = 3\)). Suppose the characters are sorted in increasing order of frequency. For this setting, describe how to modify Huffman’s algorithm to improve its running time. Provide pseudocode along with your description.

(a) \([4\ \text{points}]\) Assuming the sort occurs prior to your algorithm, briefly describe the runtime of your algorithm (1-2 sentences is sufficient) in terms of \(k\) (the number of character types).