CS:5350 Homework 4, Spring 2016

Due in class on Thu, Feb 25

Collaboration: You are welcome to form groups of size 2 and work on your homeworks in groups. Of course, you are not required to work in groups. Every group should make one submission and names of both group members should appear on the submission and both students in a group will receive the same score. Other than the TA and the professor, you can only discuss homework problems with your group partner. Collaboration can be positive because talking to someone else about these problems can help to clarify your ideas and you will also (hopefully) get to hear about different ways of thinking about the problem. On the other hand, collaboration can be negative if one member of the group rides on work being done by the other member – please avoid this situation. If your solutions are (even partly) based on material other than what has been posted on the course website, you should clearly acknowledge your outside sources.
Late submissions: No late submissions are permitted. You will receive no points for your submission if your submission is not turned in at the beginning of class on the due date.
Evaluation: Your submissions will be evaluated on correctness *and* clarity. Correctness is of course crucial, but how clearly you communicate your ideas is also quite important.

1. This problem is related to the *weighted interval scheduling (WIS)* problem discussed in class. For ease of exposition, let us assume that all start and finish times of intervals are distinct. Also recall that the weights of all intervals are non-negative.

   (a) Suppose that we label the intervals $I_1, I_2, \ldots, I_n$ in left-to-right order of *start* times. In other words, the labeling $I_1, I_2, \ldots, I_n$ is such that $s_1 < s_2 < \cdots < s_n$. Let $OPT(j)$ denote the weight of an optimal solution to WIS with input $\{I_1, I_2, \ldots, I_j\}$. **Prove** or **disprove** the following recurrence:

   $$OPT(j) = \max\{OPT(j-1), OPT(p(j)) + w_j\}$$

   for all j, $1 \leq j \leq n$, and $OPT(0) = 0$.
   Here $p(j)$ is the maximum index $k < j$ such that I_k does not intersect I_j. If no such k exists, then $p(j) = 0$.

   (b) Assume the same labeling as in part (a). But, now let $OPT(j)$ be the weight of an optimal solution to WIS with input $\{I_j, I_{j+1}, \ldots, I_n\}$. **Prove** or **disprove** the following recurrence:

   $$OPT(j) = \max\{OPT(j+1), OPT(q(j)) + w_j\}$$

   for all $1 \leq j \leq n$ and $OPT(n+1) = 0$.
   Here $q(j)$ is the minimum index $k > j$ such that I_k does not intersect I_j. If no such k exists, then $q(j) = n + 1$.

2. (This problem is from "Algorithms" by Dasgupta, Papadimitriou, and Vazirani.) A certain string-processing language offers a primitive operation which splits a string into two pieces. Since this operation involves copying the original string, it takes n units of time for a string of length n, regardless of the location of the cut. Suppose, now, that you want to break the string into many pieces. The order in which the breaks are made can affect the total running time. For example, if you want to cut a 20-character string at positions 3 and 10, then making the first cut at position 3 incurs a cost of 20 + 17 = 37, whereas cutting at position 10 first, incurs a cost of 20 + 10 = 30.

   Give a dynamic programming algorithm that, given the locations of m cuts in a string of length n, finds the minimum cost of breaking the string into $m + 1$ pieces.

   Notes: To be more concrete, suppose that the given string is $S = s_1 s_2 \ldots s_n$. A cut at position i, $1 \leq i \leq n-1$, partitions S into a prefix $s_1 s_2 \ldots s_i$ and a suffix $s_{i+1} s_{i+2} \ldots s_n$.

The input to the problem is a string $S = s_1 s_2 \ldots s_n$ and a set $\{j_1, j_2, \ldots, j_m\}$ of m cut positions. For each k, $1 \leq k \leq m$, j_k satisfies $1 \leq j_k \leq n - 1$.

How to write your solution: (i) First define your subproblems carefully. Your subproblems will be defined using some number of parameters; make sure you clearly state the range of values these parameters can take. (ii) Second, state the recurrence the relates the cost of an optimal solution of a subproblem to costs of optimal solutions to smaller subproblems. Make sure that the base cases of your recurrence are accurately specified. (iii) Third, describe how you plan to store solutions of your subproblems and the order in which your data structure will be filled. (iv) Finally, write pseudocode that implements the dynamic programming algorithm that computes the minimum cost of cutting the string via the given cuts.

3. (This problem is also from "Algorithms" by Dasgupta, Papadimitriou, and Vazirani.) Given integers n and k, along with probabilities $p_1, p_2, \ldots, p_n \in [0, 1]$, you want to determine the probability of obtaining *exactly* k heads, when n biased coins are tossed independently at random, where p_i is the probability that the ith coin comes up heads. Give an $O(nk)$ dynamic programming algorithm for this task. Assume that you can multiply and add numbers in the range $[0, 1]$ in constant time each.

   Notes: The guidelines on "How to write your solution" from Problem 2 apply to this problem as well.