CS302 - Problem Set 8

Due: Monday, Nov 13

1. Suppose you run a company with two offices, one in Washington and the other in San Francisco, California. You always spend the whole week in one location, but each weekend, you decide whether to fly to the other office. In week i, you can make W_i dollars if you are in Washington, and C_i dollars if you are in San Francisco. Each flight from one office to the other costs \$1000. Suppose you are give the lists $W_1, W_2, \ldots, W_n$ and $C_1, C_2, \ldots, C_n$. Also suppose you are in Washington initially, and need to be back in Washington after the nth week. What schedule will maximize your profits? (Note your solution should not just give the maximum profit, but should return a schedule.)

   (a) **[6 points]** (*) A greedy algorithm would always choose to work in the office with the larger profit that week. Give a counter example showing that this strategy is not always optimal.

   (b) **[6 points]** (**) Think about designing a dynamic programming algorithm. In your algorithm, you should first construct an array, and then work backwards through the array. Explain what values you will put in the array, provide a recurrence relation that you can use to fill out the array, and explain why this relation is correct. (You don't need to give a formal proof.)

   (c) **[9 points]** (**) Give psuedocode for a dynamic programming algorithm.

   (d) **[11 points]** (**) Use a loop invariant to prove that when you work backwards through the array to construct the optimal solution is correct.

   (e) **[3 points]** (*) What is the runtime of your algorithm?

2. Suppose you are given an array A of size n. Then the longest ascending subsequence of A is a the longest list $A[i_1], A[i_2], A[i_3], \ldots, A[i_k]$ such that $i_1 < i_2 < i_3 < \cdots < i_k$, and $A[i_1] \leq A[i_2] \leq \cdots \leq A[i_k]$.

   (a) (**) Describe a dynamic programming algorithm that runs in $O(n^2)$ time and outputs the longest ascending sequence.

   (b) (***) Describe a dynamic programming algorithm that runs in $O(n \log n)$ time and outputs the length of the longest ascending sequence. (It is also possible to output the longest ascending sequence itself in $O(n \log n)$ time, so feel free to think about that too.)

3. A scheduling problem. Suppose you have n events, each with a start time s_i and end time f_i, for $i \in \{1, n\}$. Unfortunately, you only have one auditorium, and you can't schedule conflicting events (events where a start time of one is between the start time and end time of another.) We would like to maximize the number of events that are held. For each of the following greedy algorithms, either provide an example where the algorithm does not perform optimally, or prove that it gives an optimal algorithm. (For the correctness proof, try using an exchange algorithm similar to the one for the scheduling problem we looked at in class

where not all values are distinct, but base the progression on selection sort rather than bubble sort.) (In each case, ties can be broken using any method.) [3 points (*) for each counter example, 11 points (**) for proof of correctness.]

(a) At each iteration, pick the remaining event with the earliest start time.

(b) At each iteration, pick the remaining event that is the shortest ($f_i - s_i$ is smallest.)

(c) At each iteration, pick the remaining event with the earliest finish time.

(d) At each iteration, pick the remaining events with the fewest conflicts with other remaining events.