CS 577: Introduction to Algorithms — Homework 5

Out: 03/01/18 — **Due: 03/08/18**

Ground Rules

- The homework is worth 5 points (out of a total of 100 points you can accumulate in this course).
- The homework is to be done and submitted in pairs. You can partner with someone from either section.
- The homework is due at the beginning of either lecture on the due date.
- No extensions to the due date will be given under any circumstances.
- Turn in your solution to each problem on a **separate sheet** of paper (or sheets stapled together), with your names clearly written on top.

Problems

1. **(Worth: 2 points. Page limit: 1 sheet; 2 sides)** An airline company wants to schedule staff to attend their ticketing desk at an airport. The desk must be attended for half an hour any time they have an outgoing flight, starting fifteen minutes before the boarding time. Due to union rules, each employee can cover the desk for a two-hour block only each week. Design and analyze an algorithm that takes as input the boarding times of each outgoing flight during a week, and gives a schedule involving the fewest number of employees possible. Your algorithm should have a runtime of $O(n \log n)$, where n is the number of flights implicit in the input. (For simplicity, ignore issues such as flight delays, and assume no boarding time is between 11:30pm–12:30am.)

   Your submission should clearly describe the algorithm, provide a formal proof of correctness, and an argument for the running time. We recommend using a greedy approach for the algorithm.

2. **(Worth: 3 points. Page limit: 1 sheets; 2 sides)** Consider the following card game between two players. The dealer deals n cards face up in a row. Each card has a value visible to both players. The players move in sequence. Each player, at her turn, can pick either the rightmost or the leftmost of the remaining cards on the table. The game ends when no cards are left. The goal of each player is to accumulate as much value as possible.

   For example, suppose that $n = 5$ and the cards have values $3, 2, 1, 7, 4$ in order from left to right. Then suppose the players pick the cards in the order $\mathbf{4}, 7, \mathbf{3}, 2, \mathbf{1}$, where boldface indicates the first player's picks, and the remainder the second player's picks. Then the first player gets total value $4 + 3 + 1 = 8$ and the second gets value $7+2 = 9$. On the other hand, if the players pick the cards in the order $\mathbf{3}, 4, \mathbf{7}, 2, \mathbf{1}$, then the first player gets total value $3+7+1 = 11$ and the second gets value $4+2 = 6$. One can check that the first player can guarantee no better than 8 when picking 4 as the first card. So, the first player should start by picking the leftmost card with value 3.

   Design and analyze $O(n^2)$ time algorithm for computing the optimal strategy for the first player. In particular, given a list of values, your algorithm should output whether the first player should pick the leftmost or the rightmost card in order to maximize her total value. Assume that the second player also plays optimally.

   We recommend using dynamic programming for this problem. Your submission should contain the following parts. (1) Clearly define a recursive function and provide the recursive equations for the function. (2) Give a brief informal argument justifying your recursive equation. We do not need a formal proof. (3) Provide an algorithm based on the recursive equation. (4) Argue the running time of your algorithm.