Foundations of Algorithms, Spring 2018: Homework 4

Due: Monday, March 26, 11:59pm

Just to make sure nobody goes down the wrong path - the final question is not a dynamic programming question. Our next topic is graph algorithms and the final question falls into that category.

Problem 1 (12 points: 7 for implementation / 5 for writeup)

Consider the following problem related to ordering playing cards in your hand. You hold n cards in your hand (you have a big hand; n can be an arbitrarily large number). The n cards have distinct integer values in the range $1, 2, \ldots, n$. The cards have been shuffled, so you initially hold them in a random ordering.

Design an $O(n^2)$ algorithm that determines the minimum number of cards that need to be moved in order to get the whole set of n cards in sorted, ascending order. One move corresponds to extracting one card from its current location, and moving it to a different location. In the process, the remaining cards shift as necessary in your hand to make room (this shifting does not count as any movement).

For example, if you hold the cards $2, 3, 1, 5, 4$ in your hand, it will take 2 moves to arrange the cards in sorted, ascending order. The 1 card can be moved to the leftmost position. The 4 card can be moved left of the 5.

Problem 2 (20 points: 14 for implementation / 6 for writeup)

a) Given is an algebraic expression involving only positive integers and the operators $+$ and $*$. Design a greedy $O(n)$ algorithm that determines the maximum possible value that can be obtained from the expression by fully parenthesizing it.

   For example, given the expression $3 * 2 + 4 * 5$, the maximum possible value of the expression is 90. One way of achieving this value is by parenthesizing as follows: $((3 * (2 + 4)) * 5)$.

b) Given is an algebraic expression involving only positive integers and the operators $+$ and $-$. Design an $O(n^3)$ dynamic programming algorithm that determines the maximum possible value that can be obtained from the expression by fully parenthesizing it.

   For example, given the expression $3 + 2 - 4 - 5$, the maximum possible value of the expression is 6. One way of achieving this value is by parenthesizing as follows: $(3 + (2 - (4 - 5)))$.

Problem 3 (18 points: 12 for implementation / 6 for writeup)

You are working at an amusement park on a day when a large company reserves the entire park for its employees. You are working at a ride fed by two lines. You already feel a headache coming on, and your goal is to get everybody in the two lines onto the ride with minimum aggravation of your headache. Here are the characteristics of your problem:

1. There are three types of employees: (E)xecutives, (V)eterans, and (N)ew Hires.
2. The two lines are filled with a mix of the three different types of employees. Line 1 has m people in it. Line 2 has n people in it.
3. The ride fits two people at a time. Standard procedure is to take the person at the front of each line and put them together on the ride. However...
4. E's and N's don't want to sit together. Matching an E with an N causes grumbling that incurs 5 units of headache for you. (Any other pairing causes no problem.)
5. You can choose to fill a ride by taking two people from the front of the same line. However, this causes grumbling from the other line that incurs 3 units of headache to you, unless the other line is empty. If the two people you take from the front of the same line happen to be an E and an N, you would also incur an additional 5 units of headache (see above).
6. You can also choose to fill a ride with only one person. But then everybody remaining complains about the inefficiency, incurring 4 units of headache, unless there is nobody left at that point in either line (this is the last person to ride).

For example, suppose Line 1 $= \{E, E, N\}$, and Line 2 $= \{N, N, V, E\}$, where the lines are listed from the back to the front. You could start by pairing V, E from line 2 at a cost of 3. Then pair N from the front of each line at no cost. Then match E, E from line 1 at a cost of 3. Finally, N from line 2 rides alone at no cost (because there is nobody left). Thus the total cost is 6. However, this is not optimal. You could do better by letting N from line 1 ride alone to start, at a cost of 4. Then pair the front of each line twice (E, E and E, V) at no cost. Finally, only N, N remains in line 2, and can ride at no cost (since the other line is empty). Thus the total cost is 4. No other solution yields a total less than 4.

Give an $O(mn)$ dynamic programming solution to this problem, that computes the minimum units of headache you will incur getting all of the company employees on the ride.

Problem 4 (12 points: 8 for implementation / 4 for writeup)

Given is an $m \times n$ array of zeros and ones, plus exactly one number 2 and exactly one number 3. The array represents a maze, where zeros correspond to empty positions and ones correspond to walls. Number 2 corresponds to a person and number 3 corresponds to the person's house. The person can move from one empty position to another adjacent empty position. Adjacent positions refer to the 4-neighbors (north, south, east and west) of the current position. Give an $O(mn)$ algorithm that outputs the length of the shortest path from position 2 to position 3.