Design and Analysis of Algorithms: Homework #3

Due in class on March 1, 2018

Professor Kasturi Varadarajan

Alic Szecsei

Problem 1

You and your eight-year-old nephew Elmo decide to play a simple card game. At the beginning of the game, the cards are dealt face up in a long row. Each card is worth a different number of points. After all the cards are dealt, you and Elmo take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, you can decide which of the two cards to take. The winner of the game is the player that has collected the most points when the game ends.

Having never taken an algorithms class, Elmo follows the obvious greedy strategy—when it's his turn, Elmo always takes the card with the higher point value. Your task is to find a strategy that will beat Elmo whenever possible. (It might seem mean to beat up on a little kid like this, but Elmo absolutely hates it when grown-ups let him win.

(a) Prove that you should not also use the greedy strategy. That is, show that there is a game that you can win, but only if you do *not* follow the same greedy strategy as Elmo.

(b) Describe and analyze an algorithm to determine, given the initial sequence of cards, the maximum number of points that you can collect playing against Elmo.

Solution

Part A

Suppose there are a total of 8 cards, with point values ranging from 1 to 8. When dealt, they are arranged like so:

3	8	7	6	5	4	2	1

Table 1: an example game

If you were to play using the greedy strategy, you would first take card 3, while Elmo would take 8, and so on. At the end of the game, you would have $3 + 7 + 5 + 2 = 17$ points, while Elmo would have $8 + 6 + 4 + 1 = 19$ points.

However, if you first took card 1, Elmo would take card 3, leaving you to take card 8, and so on, now playing greedily. At the end of the game, you would have $1+8+6+4 = 19$ points, while Elmo would have $3+7+5+2 = 17$ points.

Thus, you can only win this game by making your first move take the smaller of the two available cards, as opposed to Elmo's greedy strategy.

Problem 1 continued on next page...

Part B

Algorithm 1 Elmo's Card Game

```
$memCards \leftarrow$ empty hash map
function ELMO($cards[1..n]$)
    if $n = 0$ then
        return $0$
    else if $n = 1$ then
        return $cards[1]$
    else if $memCards$ contains $cards$ then
        return $memCards[cards]$
    else
        $L \leftarrow cards[2..n]$                                  ▷ Try taking the first card
        if $cards[2] > cards[n]$ then                               ▷ Remove Elmo's card
            remove the first element from $L$
        else
            remove the last element from $L$
        end if
        $l \leftarrow cards[1] +$ ELMO($L$)
        $R \leftarrow cards[1..n-1]$                                ▷ Try taking the last card
        if $cards[1] > cards[n-1]$ then                             ▷ Remove Elmo's card
            remove the first element from $R$
        else
            remove the last element from $R$
        end if
        $r \leftarrow cards[n] +$ ELMO($R$)
        $x \leftarrow$ the maximum of $l$ and $r$
        Store $x$ in $memCards[cards]$
        return $x$
    end if
end function
```

This algorithm tests both options the player has: removing the first card, and removing the last card. The total points for whichever strategy leads to the player having the most points at the end of the game is memoized in a hash map.

To determine the run-time, we can examine the total number of sub-problems. A base case contains n different "final card"s to be picked up. Adding on to that, there are $n - 1$ possible prior states - one for each pair of neighboring cards. We can continue in this manner, until we reach a single possible state - the initial set of n cards - so that we have a total of $\sum_{i=1}^{n} i$ possible sub-problems, which is $O(n^2)$.

Problem 2

A palindrome is any string that is exactly the same as its reversal, like `I`, or `DEED`, or `RACECAR`, or `AMANAPLANACAT-ACANALPANAMA`.

(a) Describe and analyze an algorithm to find the length of the *longest subsequence* of a given string that is also a palindrome. For example, the longest palindrome subsequence of `MAHDYNAMICPROGRAMZLETMESHOWYOUTHEM` is `MHYMRORYHM`, so given that string as input, your algorithm should output the number 11.

(c) Any string can be decomposed into a sequence of palindromes. For example, the string `BUBBASEESABANANA` ("Bubba sees a banana.") can be broken into palindromes in the following ways (and many others):

```
          BUB • BASEESAB • ANANA
       B • U • BB • A • SEES • ABA • NAN • A
       B • U • BB • A • SEES • A • B • ANANA
B • U • B • B • A • S • E • E • S • A • B • A • N • ANA
```

Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string. For example, given the input string `BUBBASEESABANANA`, your algorithm would return the integer 3.

Solution

Part A

Part C

Problem 3

Suppose we are given a set L of n line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which no pair of segments intersects.

Solution

Problem 4

Oh, no! You have been appointed as the organizer of Giggle, Inc.'s annual mandatory holiday party! The employees at Giggle are organized into a strict hierarchy, that is, a tree with the company president at the root. The all-knowing oracles in Human Resources have assigned a real number to each employee measuring how "fun" the employee is. In order to keep things social, there is one restriction on the guest list: an employee cannot attend the party if their immediate supervisor is also present. On the other hand, the president of the company *must* attend the party, even though she has a negative fun rating; it's her company, after all. Give an algorithm that makes a guest list for the party that maximizes the sum of the "fun" ratings of the guests.

Solution