CSCI 570: Homework 7

Due Date: Oct 19th

Graded Problems

1. You are given an integer array $a[1], \ldots, a[n]$, find the contiguous subarray (containing at least one number) which has the largest sum and only return its sum. The optimal subarray is not required to return or compute. Taking $a = [5, 4, -1, 7, 8]$ as an example: the subarray $[5]$ is considered as a valid subarray with sum 5, though it only has one single element; the subarray $[5, 4, -1, 7, 8]$ achieves the largest sum 23; on the other hand, $[5, 4, 7, 8]$ is not a valid subarray as the numbers 4 and 7 are not contiguous. (20 pts)

a. Define (in plain English) subproblems to be solved. (4 pts)
b. Write a recurrence relation for the subproblems (6 pts)
c. Using the recurrence formula in part b, write pseudocode to find the subarray (containing at least one number) which has the largest sum. (5 pts)
d. Make sure you specify
    i. base cases and their values (2 pts)
    ii. where the final answer can be found (1 pt)
e. What is the complexity of your solution? (2 pts)

2. You've started a hobby of retail investing into stocks using a mobile app, RogerGood. You magically gained the power to see N days into the future and you can see the prices of one particular stock. Given an array of prices of this particular stock, where prices[i] is the price of a given stock on the ith day, find the maximum profit you can achieve through various buy/sell actions. RogerGood also has a fixed fee per transaction. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction. (20 pt).

a. Define (in plain English) subproblems to be solved. (4 pts)
b. Write a recurrence relation for the subproblems (6 pts)
c. Using the recurrence formula in part b, write pseudocode to solve the problem. (5 pts)
d. Make sure you specify
    i. base cases and their values (2 pts)
    ii. where the final answer can be found (1 pt)
e. What is the complexity of your solution? (2 pts)

3. You are given an array of positive numbers $a[1], \ldots, a[n]$. For a sub-sequence $a[i_1], a[i_2], \ldots, a[i_t]$ of array a (that is, $i_1 < i_2 <, \ldots, i_t$): if it is an increasing sequence of numbers, that is, $a[i_1] < a[i_2] < \ldots < a[i_t]$, its happiness score is given by

$$\sum_{k=1}^{t} k \times a[i_k]$$

Otherwise, the happiness score of this array is zero.
For example, for the input a = [22, 44, 33, 66, 55], the increasing subse-quence [22, 44, 55] has happiness score (1) × (22) + (2) × (44) + (3) × (55) = 275; the increasing subsequence [22, 33, 55] has happiness score (1) × (22) + (2) × (33) + (3) × (55) = 253; the subsequence [33, 66, 55] has happiness score 0 as this sequence is not increasing. Please design an efficient algorithm to only return the highest happiness score over all the subsequences (20 pts)

a. Define (in plain English) subproblems to be solved. (4 pts)
b. Write a recurrence relation for the subproblems (6 pts)
c. Using the recurrence formula in part b, write pseudocode to find the highest happiness score over all the subsequences. (5 pts)
d. Make sure you specify
    i. base cases and their values (2 pts)
    ii. where the final answer can be found (1 pt)
e. What is the complexity of your solution? (2 pts)

4. You are given an $m \times n$ binary matrix $g \in \{0, 1\}^{m \times n}$. Each cell either contains a "0" or a "1". Give an efficient algorithm that takes the binay matrix g, and returns the largest side length of a square that only contains 1's. You are **not** required to give the optimal solution. (20 pts)

a. Define (in plain English) subproblems to be solved. (4 pts)
b. Write a recurrence relation for the subproblems (6 pts)
c. Using the recurrence formula in part b, write pseudocode using iteration to compute the largest side length of a square that only contains 1's to meet the objective. (5 pts)
d. Make sure you specify
    i. base cases and their values (2 pts)
    ii. where the final answer can be found (1 pt)
e. What is the complexity of your solution? (2 pts)

Practice Problems

1. Given n sand piles of weights $w[1], \ldots, w[n]$ on a line, we want to merge all the sand piles together. At each step we can only merge adjacent sand piles, with cost equal to the weight of the merged sand pile. In particular, merging sand piles i and $i + 1$ has cost $w_i + w_{i+1}$. Furthermore,

one gets a single sand pile of weight $w_i + w_{i+1}$ in its place which is now adjacent to sand piles $i - 1$ and $i + 2$. Note that different merging orders will result in different final costs, please find the minimum cost to merge all the sand piles. The optimal merging order is **not** required. (20 pts)

- Sample Input: $n = 4, w[1, \ldots, n] = [100, 1, 1, 100]$.
- Sample Output: the minimum cost is 1+1+2+100+102+100 = 306 by first merging (1, 1), then (100, 2) and (102, 100).

a. Define (in plain English) subproblems to be solved. (4 pts)
b. Write a recurrence relation for the subproblems (6 pts)
c. Using the recurrence formula in part b, write pseudocode using iteration to compute the the minimum cost to merge all the sand piles to meet the objective. (5 pts)
d. Make sure you specify
    i. base cases and their values (2 pts)
    ii. where the final answer can be found (1 pt)
e. What is the complexity of your solution? (2 pts)