CSCI 570 : Homework 7

1 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.

(2). Solve Kleinberg and Tardos, Chapter 6, Exercise 10.

(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 subsequence $[22, 44, 55]$ has happiness score $(1) \times (22) + (2) \times (44) + (3) \times (55) = 275$; the increasing subsequence $[22, 33, 55]$ has happiness score $(1) \times (22) + (2) \times (33) + (3) \times (55) = 253$; 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. The optimal subsequence is **not** required to return or compute.

(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.

2 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.

- 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 $(102, 100)$.