CSCI 3104 Summer 2017 - Assignment # 5

Due: July 7th 5:30pm

Be sure to justify all of your work.

1. Write a dynamic programming algorithm to solve the longest increasing subsequence (LIS) problem. You are given an array A of n integers. An *increasing subsequence* is a sequence of k indices $0 \leq i_1 < i_2 < \ldots < i_k \leq (n-1)$ such that $A[i_1] < A[i_2] < \ldots < A[i_k]$. Thus, given an array A, you are to compute an increasing subsequence $i_1, \ldots, i_k$ whose length k is the longest possible.
   Example: $A = [1, 5, 2, 3, 8]$. The longest increasing subsequence has length 4 with corresponding indices $i = [0, 2, 3, 4]$.
   Let's define the function `LIS(A, j, M)` to be length of the LIS for the first j elements of the array $(A[0], \ldots, A[j-1])$, where j ranges from 0 to n and all elements of this subsequence are less than M.
   a) For $A = [1, 5, 2, 3, 8]$, write down the values of `LIS(A, 4, 3)` and `LIS(A, 5,` ∞`)`.
   b) Write down the base case for `LIS(A, 0, M)`.
   c) Write down the recurrence for `LIS(A, j, M)` for any $1 \leq j \leq n$.
   d) Describe a scheme to memoize the recurrence `LIS` above.
   e) Implement your dynamic programming algorithm scheme in `Python`. Input to your function is the array A, and the output should be the LIS itself.

2. One of Gru's favorite games to watch is a game played by his minions. The game works as follows: The game is played on an n x n square grid by a minion. The minion starts by placing a token on any square of the grid. On each turn, the minion moves the token either one square down, or one square to the right. The game finishes when the token is moved off the edge of the board. Each square on the grid has some numerical value (could be positive, negative, or zero). The initial score for the minion is zero, and each time the token lands on a square, its value gets added to the minion's score. The goal of the game is to score as many points as possible.
   a) Provide an algorithm (pseudocode) to compute the maximum possible score for this game, given an n x n array of values as input.
   b) Prove the correctness of your algorithm.
   c) Analyze your algorithm's time and space complexities.

3. Gru claims he can determine whether a directed graph G, when represented by an adjacency matrix, contains a black hole in $\mathcal{O}(V)$ time. A black hole is defined as a vertex with in-degree $|V|-1$ (every other vertex points to this one) and out-degree 0. Prove that Gru is correct, and provide pseudocode for this algorithm (the algorithm to determine whether a graph G contains a black hole, given that G is represented as an adjacency matrix).

4. a) Gru, always curious, gives you an array A of numbers, with length n, and he aims to find an algorithm to find out if any two elements of the array A sum to T. For example, suppose $A = [14, -36, 21, 4, 9, 41]$. If $T = 25$, the result is `TRUE` because $A[2] + A[3] = 25$. However, if $T = -1$, the result would be `FALSE`, since no two elements in A sum to -1.
   Of course, Gru assigns *you* the task of writing a `Python` function `sumTwo(A, T)` that determines if, for the given array A, two elements sum to T. Since Gru is always interested in efficiency, he informs you that your algorithm must run in $\Theta(n)$ in the average case (**Hint:** Use a hashtable).

   b) Now Gru asks you to write a `Python` function `sumThree(A, T)` which checks if three elements in A sum to T (**Hint:** Use your function from part (a) as a subroutine).