CS 141, Winter 2018
Assignment 3
Posted: February 9th, 2018 Due: February 23th, 2018, 11:59pm

Notice

- Include your full name and student ID in your solution
- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on iLearn by the deadline. Late submission allowed for 20% penalty for a calendar day.

Problem 1. (10 points) Write an algorithm to construct the actual solution of the matrix chain multiplication problem (i.e., the parentheses order). Trace its output on the following examples:
(a) Three matrices (A, B, and C) with dimensions 10 x 50 x 5 x 100, respectively.
(b) Four matrices (A, B, C, and D) with dimensions 20 x 5 x 10 x 30 x 10, respectively.

Problem 2. (5 points) Discuss the main differences between Divide-and-Conquer method and Dynamic Programming method.

Problem 3. (25 points) Let \( A \) be a \( n \times m \) matrix of 0’s and 1’s. Design a dynamic programming \( O(nm) \) time algorithm for finding the largest square block of 1’s in \( A \). Define the dynamic programming table \( l(i, j) \) be the length of the side of the largest square block of 1’s whose bottom right corner is \( A[i, j] \).

Problem 4. (30 points) Let \( A = \{a_1, a_2, \ldots, a_n\} \) be a set of \( n \) positive integer and let \( T \) be another integer. Design a dynamic programming algorithm that determines whether there exists a subset of \( A \) whose total sum is exactly \( T \). Analyze the time- and space-complexity of your solution.
For instance, if \( A = \{4, 5, 17, 23, 11, 2\} \) and \( T = 35 \) the algorithm should return True because the subset \( \{5, 17, 11, 2\} \) sums to 35. For the same set of numbers if we choose \( T = 31 \) the problem has no solution, and the algorithm will return False.
Problem 5. (30 points) Given an array $A = \{a_1, a_2, \ldots, a_n\}$ of integers, we say that a subsequence $\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ is (monotonically) increasing if for every $i_s < i_t$, we have $a_{i_s} < a_{i_t}$. Given an array $A$ of size $n$, we want to compute the length of the longest increasing subsequence (LIS) in $A$. For instance, if $A = \{9, 5, 2, 8, 7, 3, 1, 6, 4\}$ the length of the LIS is 3, because $(2, 3, 4)$ (or $(2, 3, 6)$) are LIS of $A$. Give a $O(n^2)$ dynamic programming algorithm for this problem\(^1\). Analyze the time- and space-complexity of your solution.

\(^1\) $O(n \log n)$ is possible, but the easier $O(n^2)$ solution suffices