CS 141, Fall 2019
Assignment 3
Posted: October 28th, 2019
Due: November 11, 2019, 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. (20 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.
(20 points) Given two strings A and B and the following operations that can performed on A. Find minimum number of operations required to make A and B equal.
1. Insert
2. Delete
3. Replace

Problem 3.
(20 points) Let A be a n x m matrix of 0’s and 1’s. Design a dynamic programming \( O(nm) \) time algorithm for finding the largest square block of A that contains 1’s only.

Hint: 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.
(20 points) Given n dice each with m faces, numbered from 1 to m, find the number of ways to get sum X, where X is the sum of values on each face when all the dice are thrown.
Problem 5. (20 points) Let $A = a_1, a_2, ..., a_n$ and 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.