CSC 505, Homework 3
Due date: Friday, October 15, 9:00 PM

Homework should be submitted via Gradescope in PDF. To avoid reduced marks, please submit **word/latex-formated PDF** file, **NOT scanned writing in pdf format**. All assignments are due on 9 PM of the due date. Late homework will be accepted only in circumstances that are grounds for excused absence under university policy. The university provides mechanisms for documenting such reasons (severe illness, death in the family, etc.). Arrangements for turning in late homework must be made by the day preceding the due date if possible.

All assignments for this course are intended to be individual work. Turning in an assignment which is not your own work is cheating. The Internet is not an allowed resource! Copying of text, code or other content from the Internet (or other sources) is plagiarism. Any tool/resource must be approved in advance by the instructor and identified and acknowledged clearly in any work turned in, anything else is plagiarism.

If an academic integrity violation occurs, the offending student(s) will be assessed a penalty that is at least as severe as getting a 0 for the whole homework for which the violation occurred, and the case will be reported to the Office of Student Conduct.

Instructions about how to "give/describe" an algorithm (taken from Erik Demaine): Try to be **concise, correct, and complete. To avoid deductions,** you should provide (1) a textual description of the algorithm, and, if helpful, flow charts and pseudocode; (2) at least one worked example or diagram to illustrate how your algorithm works; (3) a proof (or other indication) of the correctness of the algorithm; and (4) an analysis of the time complexity (and, if relevant, the space complexity) of the algorithm. **Remember that, above all else, your goal is to communicate.** If a grader cannot understand your solution, they cannot give you appropriate credit for it.

1. *Purpose: Apply various algorithm design strategies to solve a problem, practice formulating and analyzing algorithms, implement an algorithm.* In the US, coins are minted with denominations of 50, 25, 10, 5, and 1 cent. An algorithm for making change using the *smallest* possible number of coins repeatedly returns the biggest coin smaller than the amount to be changed until it is zero. For example, 17 cents will result in the series 10 cents, 5 cents, 1 cent, and 1 cent.
a) (4 points) Give a recursive algorithm that generates a similar series of coins for changing $n$ cents. Don't use dynamic programming for this problem.

b) (4 points) Write an O(1) (non-recursive!) algorithm to compute the number of returned coins.

c) (1 point) Show that the above greedy algorithm does not always give the minimum number of coins in a country whose denominations are 1, 6, and 10 cents.

d) (6 points) Given a set of arbitrary denominations $C = (c_1,...,c_d)$, describe an algorithm that

uses **dynamic programming** to compute the minimum number of coins required for making change. You may assume that *C* contains 1 cent, that all denominations are different, and that the denominations occur in in increasing order.

e) (6 points) Implement the algorithm described in d). The code framework are given in the zip file: framework.zip. To avoid loss of marks please make sure that all provided test cases pass on remote-linux server by using the test file. Instructions for setting up remote-linux server and testing are given in the document HW3_Programming_Assignment_Setup.pdf.

*Problem 2.* (10 points) In class we showed that multiplying two matrices

$$\underset{m\times p}{C} \leftarrow \underset{m\times n}{A} * \underset{n\times p}{B}$$

requires mnp scalar multiplications. You are given the following matrix chain:

$$\underset{d_0\times d_1}{A_1} * \underset{d_1\times d_2}{A_2} * \underset{d_2\times d_3}{A_3} * \underset{d_3\times d_4}{A_4}$$
$$\underset{2\times 3}{} \quad \underset{3\times 5}{} \quad \underset{5\times 2}{} \quad \underset{2\times 4}{}$$

Denote *m[i,j]* the *minimum number of scalar multiplications to compute* $A_i * \dots * A_j$. In class we showed the following recurrence:

$$m[i,j] := \begin{cases} 0 \text{ ; if } i=j \\ \min_{i-1<k<j} m(i,k)+m(k+1,j)+d_{i-1}d_k d_j \text{ ; otherwise} \end{cases}$$

a) (6 points) Fill the Table below with the missing values for *m[i,j]*. Also, for each *m[i,j]* put the corresponding value *k*, where the recurrence obtains its minimum value, next to it.

| i\j | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **1** | | | | |
| **2** | Undefined | | | |
| **3** | Undefined | Undefined | | |
| **4** | Undefined | Undefined | Undefined | |

Use the table to answer the following questions:

b) (1 point) What is the minimum number of scalar multiplications required to compute the matrix chain?

c) (2 points) Give the optimal order of computing the matrix chain by fully parenthesizing the matrix chain below.

$$A_1 \quad * \quad A_2 \quad * \quad A_3 \quad * \quad A_4$$

d) (1 points) How many scalar multiplications are used to compute $(A_1 * A_2) * (A_3 * A_4)$ ? Keep the order of matrix multiplications indicated by the brackets.

*Problem 3. Purpose: practice algorithm design using dynamic programming.* A subsequence is palindromic if it is the same whether read left to right or right to left. For instance, the sequence A,C,G,T,G,T,C,A,A,A,A,T,C,G has many palindromic subsequences, including A,C,G,C,A and A,A,A,A (on the other hand, the subsequence A,C,T is not palindromic). Assume you are given a sequence x[1...n] of characters. Denote L(i,j) the length of the longest palindrome in the substing x[i,...,j]. The goal of the Maximum Palindromic Subsequence Problem (MPSP) is to take a sequence x[1,...,n] and return the length of the longest palindromic subsequence L(1,n).

a) (4 points) Describe the optimal substructure of the MPSP and give a recurrence equation for L(i,j).
b) (6 points) Describe an algorithm that uses dynamic programming to solve the MPSP. The running time of your algorithm should be $O(n^2)$.

*Problem 4. Purpose: practice designing greedy algorithms.* (10 points) Suppose you have a long straight country road with houses scattered at various points far away from each other. The residents all want cell phone service to reach their homes and we want to accomplish this by building as few cell phone towers as possible.
More formally, think of points x1, ..., xn, representing the houses, on the real line, and let d be the maximum distance from a cell phone tower that will still allow reasonable reception. The goal is to find a minimum number of points y1,...,yk so that, for each i, there is at least one j with $| yj - xi | \leq d$.
Describe a greedy algorithm for this problem. If the points are assumed to be sorted in increasing order your algorithm should run in time O(n). Be sure to describe the greedy choice and how it reduces your problem to a smaller instance. Prove that your algorithm is correct.