Assignment 9

csci2200, Algorithms

Instructions:

- Honor code: Work on this assignment with at most one partner. Between different teams, Collaboration is at level 1 [verbal collaboration only]
- Write each problem on a separate page; If a problem has multiple parts, you can write all parts on the same page, as long as you leave space in between them.

---

1. **String shuffling:** A *shuffle* of two strings A and B is formed by interspersing the characters into a new string, keeping the characters from A and B in the same order.

   For example, the string bananaananas is a shuffle of the string banana and ananas (in several different ways, actually: bananaANANAS, banANAanaNAS and also bANANAANANAS).

   Similarly, the strings aNEVEGARIN and aNEGAVERIN are both shuffles of never and again.

   The problem: Given three strings $A[1..m]$, $B[1..n]$ and $C[1..m+n]$, come up with an efficient algorithm to determine whether C is a shuffle of A and B.

   (a) Define your subproblem. Clearly state what function you will compute, what value it should return, what the arguments represent.

   (b) Argue optimal substructure and give a recursive definition of the subproblem.

   (c) Imagine you write a function to compute the subproblem using the formula above (without dynamic programming). Briefly argue what the running time would be.

   (d) Give pseudocode for a recursive, dynamic programming approach and analyze its running time.

(e) **Extra credit:** Translate your efficient dynamic programming algorithm above into either Java or Python code. Your code should be able to take 3 arguments (if Java: preferably on the command line), which represent the three strings, and report whether the third string is a shuffle of the first two.

Test your code on: (a) BANANA, ANANAS, BANANAANANAS (True); (b) AA, BA, AABA (True); (c)BA, AA, AABA (True); (d) A, BA, AAB (False).

Take a screenshot of your code and show that it passes these 4 tests, and add it to the rest of the homework. Note: do not submit the source code.

2. **Art gallery guarding:** In the *art gallery guarding* problem we are given a line L that represents a long hallway in an art gallery. We are also given a set $X = \{x_0, x_1, x_2, ..., x_{n-1}\}$ of real numbers that specify the positions pf paintings in this hallway; assume that each painting is a point. Suppose that a single guard can protect all the paintings within a distance at most 1 of his or her position, on both sides.

   Design an algorithm for finding a placement of guards that uses the minimum number of guards to guard all the paintings in X. Briefly argue why your algorithm is correct and analyze its running time.

   We expect: The algorithm as pseudocode, a brief explanation, analysis and justification of correctness.