CSE 321 - Homework 5

Due date: 08/01/2023, 23:59

1. ***15 pts.*** Given an array of strings, design a divide-and-conquer algorithm that finds the longest common substring that is present at the beginning of all these strings.

   Example:
   Input: ["programmable", "programming", "programmer", "programmatic", "programmability"]
   Output: "programm"

   Example:
   Input: ["compute", "compatible", "computer", "compare", "compactness"]
   Output: "comp"

2. Assume you are a merchant. Your aim is to maximize your profit by buying some goods preferably at a low price and then selling them preferably at a higher price afterwards. Assuming that you know the market prices for a given period of time, your goal is to decide when to buy and when to sell in order to maximize your profit. Do not forget that you cannot sell before you buy.

   (a) ***15 pts.*** Design a divide-and-conquer algorithm to solve this problem.

   (b) ***5 pts.*** Design an algorithm that is not based on the divide-and-conquer approach to solve this problem in linear time.

   (c) ***5 pts.*** Compare these two algorithms in terms of their worst-case time complexity.

   Example:
   Input: [10, 11, 10, 9, 8, 7, 9, 11]
   (It means that the market price is 10₺ on Day_0, 11₺ on Day_1, ..., 11₺ on Day_7)
   Output: Buy on Day_5 for 7₺ and sell on Day_{11} for 11₺.

   Example:
   Input: [100, 110, 80, 90, 110, 70, 80, 80, 90]
   (It means that the market price is 100₺ on Day_0, 110₺ on Day_1, ..., 90₺ on Day_8)
   Output: Buy on Day_2 for 80₺ and sell on Day_4 for 110₺.

3. ***20 pts.*** Given an integer array, design a dynamic programming algorithm to find the length of the longest increasing sub-array. The elements of the sub-array should be consecutive in the main array. Make sure your algorithm does not compute the sub-problems which had been computed before.

   Example:
   Input: [1, 4, 5, 2, 4, 3, 6, 7, 1, 2, 3, 4, 7]
   Output: 5 (the sub-array is [1, 2, 3, 4, 7])

   Example:
   Input: [1, 2, 3, 4, 1, 2, 3, 5, 2, 3, 4]
   Output: 4 (the sub-array is [1, 2, 3, 4] or [1, 2, 3, 5])

   PS: Printing the sub-array is not mandatory.

4. Remember the first problem of Homework 4.

   Consider a computer game with a 2D map with axes $A(A_1, A_2, ..., A_n)$ and $B(B_1, B_2, ..., B_m)$. The goal is to start from A_1B_1, move step by step to arrive at A_nB_m, and reach the highest possible score. At each coordinate the player arrives, they gain a (positive) number of game points. Additionally, there is a rule that restricts the movements. If the player is at A_iB_j, their next move should be either A_iB_{j+1} or $A_{i+1}B_j$, and no other movement is possible.

   Example:

   Input: $n = 4, m = 3$
   Game map:

	B_1	B_2	B_3
A_1	25	30	25
A_2	45	15	11
A_3	1	88	15
A_4	9	4	23

   Output:
   Route: $A_1B_1 \rightarrow A_2B_1 \rightarrow A_2B_2 \rightarrow A_3B_2 \rightarrow A_3B_3 \rightarrow A_4B_3$
   Points: $25 + 45 + 15 + 88 + 15 + 23 = 211$

   (a) **15 pts.** Design a dynamic programming algorithm to solve this problem.

   (b) **15 pts.** Design a greedy algorithm to solve this problem.

   (c) **10 pts.** Consider the brute-force solution given in the solution file of Homework 4. Compare these three solutions in terms of their correctness and worst-case time complexity.

Important Notes

- Whenever you are asked to design an algorithm, implement your solution in Python3. Write a driver function to test each of these algorithms. Inputs should be randomly generated (by using *random* library) or taken from the user (you may assume that the inputs are proper). Gather all of the python code in a single .py file. Do not use external libraries or functions to implement a part of the solution. Pay attention to clean coding.
- Write a report explaining the reasoning behind the algorithms you coded and analyze the worst-case time complexity of each of them. This report should also include your answers to Question 2-c and Question 4-c. Write your report by using a program like MS Office and then convert it to a single PDF file. Pictures of handwritten works are **not accepted**.
- Upload two files only, a .py file and a .pdf file, **not a .zip or a .rar file**.