Assignment 3 (due Feb 17th Friday noon, submit pdf on LEARN)

Please first read the instructions at

https://www.student.cs.uwaterloo.ca/~cs341/#assignments.

1. [15 marks] Stable Marriage Problem

We are given an instance of the Stable Marriage problem where we are matching three interns, denoted $d_1, d_2, d_3$ to three hospitals, denoted $h_1, h_2, h_3$. The instance $I$ is specified by the following preference lists, where “$>$” means “prefers”:

- preference list for $d_1 : h_1 > h_2 > h_3$
- preference list for $d_2 : h_1 > h_3 > h_2$
- preference list for $d_3 : h_3 > h_2 > h_1$

- preference list for $h_1 : d_2 > d_1 > d_3$
- preference list for $h_2 : d_2 > d_3 > d_1$
- preference list for $h_3 : d_1 > d_2 > d_3$

(a) [5 marks] There are six possible matchings of the three interns with the three hospitals. For each matching, determine all the instabilities that exist for the instance $I$. How many of the six matchings are stable?

(b) [5 marks] Show how the Gale-Shapley algorithm would execute on the instance $I$. Assume that interns “propose” to hospitals. Show all the steps in the algorithm.

(c) [5 marks] Show how the Gale-Shapley algorithm would execute on the instance $I$, if hospitals “proposed” to interns. Show all the steps in the algorithm.

2. [15 marks] Building Fast Food Restaurants

MacWendy’s wishes to build fast food restaurants on a very long highway. There are $n$ possible locations for restaurants, denoted by $L_1, \ldots, L_n$. These values are positive integers which represent distances from the beginning of the highway. Associated with each possible location $L_i$ is an anticipated profit, denoted $p_i$. We are also given a distance $D$, which is a positive integer, and a positive integer $k \leq n$. Thus a problem instance is defined by the list of values

$$(L_1, \ldots, L_n, p_1, \ldots, p_n, D, k).$$

A feasible solution is a set of exactly $k$ restaurant locations, such that each pair of locations is distance at least $D$ apart (MacWendy’s does not want to build restaurants too close together). The optimal solution is the feasible solution that maximizes the total profit of the selected locations.
(a) [10 marks] Present a dynamic programming algorithm for this problem. The running time should be $O(n^2)$ (but you will still receive partial marks for less efficient dynamic programming algorithms).

Remember to define your subproblems precisely, derive the base cases and recursive formulas, write and analyze pseudocode, and show how to retrieve the optimal subset.

You only need to find one optimal subset, in case there is more than one (i.e., ties can be broken arbitrarily).

*Hint:* First sort the locations $L_1, \ldots, L_n$ in increasing order. For each $i$ and $j$, consider the subproblem of maximizing the total profit of a subset of exactly $j$ locations chosen from the points $\{L_1, \ldots, L_i\}$, subject to the condition that every pair of locations chosen is distance at least $D$ apart.

(b) [5 marks] Show the result of your algorithm on the following problem instance. Present all the tables that are filled in to solve the problem instance, and show the computation of the optimal solution.

\[
\begin{align*}
&n = 11 \\
&k = 4 \\
&D = 3 \\
&\text{locations:} & \quad & 1 & 3 & 6 & 8 & 9 & 11 & 14 & 16 & 17 & 18 & 20 \\
&\text{profits:} & \quad & 4 & 10 & 9 & 6 & 12 & 5 & 8 & 10 & 7 & 11 & 2
\end{align*}
\]

3. [22 marks] **Min-Max Pairwise Distance**

Given numbers $a_1 < a_2 < \cdots < a_n$ and $b_1 < b_2 < \cdots < b_m$ with $n \leq m$, form $n$ pairs $(a_1, b_{k_1}), (a_2, b_{k_2}), \ldots, (a_n, b_{k_n})$ where the indices $k_1, \ldots, k_n$ are distinct, such that the following maximum distance is minimized:

\[
\max_{i=1}^{n} |a_i - b_{k_i}|
\]

(There may be more than one optimal solutions and you need only find any one of them.)

(a) [12 marks] Assume $m = n$, in which case the indices $k_1, \ldots, k_n$ represent a permutation of the $b_i$’s. Consider three greedy strategies:

- **Strategy X:** Pick the pair $(a_i, b_j)$ with the smallest difference $|a_i - b_j|$. Then remove $a_i$ and $b_j$, and repeat.
- **Strategy Y:** For each $i$, pair $a_i$ with $b_i$.
- **Strategy Z:** For each $i$, pair $a_i$ with $b_{n-i+1}$.

Argue the correctness of each strategy: if it is correct, give a proof; if it is incorrect, give a counterexample.

(b) [10 marks] Now consider the general case $m \geq n$. Give a dynamic programming algorithm that solves the problem. [Hint: inspired by part (a) what can we say about $b_{k_1}, \ldots, b_{k_n}$? Try defining $C[i,j]$ to be the optimal cost for the prefixes $a_1, \ldots, a_i$ and $b_1, \ldots, b_j$. Remember to define subproblems precisely, derive the recursive formula with base cases, write and analyze the pseudocode. Show how to retrieve the indices.]
A palindrome is a string that is the same backwards as forwards, e.g. kayak or abbaabba. The Longest Palindromic Subsequence problem asks for the longest palindromic subsequence (or LPS) of a given string. For example, the longest LPS of the string character is carac. Note that a subsequence does not necessarily consist of contiguous symbols.

(a) [10 marks] Use dynamic programming to solve the Longest Palindromic Subsequence problem. Define subproblems precisely and derive the recursive formula with base cases.

(b) [10 marks] Programming question: Implement your algorithm, including a “trace-back” to actually compute the LPS (not just its length). For consistency, whenever you need to break-ties, choose the segment that has smaller indices. The complexity of your algorithm should be $O(n^2)$ and your trace-back should be $O(n)$.

Hand in a printed copy of your program or electrically submit the source file as a3.c or a3.java.

The input to your algorithm will be a string $a$ consisting of upper case alphabetic characters, i.e., A, B, . . . , Z.

The output of your program should consist of

- a positive integer $k$, which is the length of the LPS of $a$, and
- a string of length $k$ that is a palindromic subsequence of $a$.

As an example, if the input string is ABCABC, then the output should be

3

ABA