Algorithms Laboratory (CS29203)
Assignment 1: Running time of algorithms
Department of CSE, IIT Kharagpur

11th August 2022

Question-1

In the census department of India, there is some analysis going on with the population of the cities and their distances from each other. Let there are n cities $(C_1, C_2, \cdots, C_n)$ under consideration where the population of the respective cities are stored in an array City$[W_1 \cdots W_n]$, where W_1 is the population of city C_1, W_2 is the population of city C_2, and so on. The population entries in the array are done in such a way that the consecutive cities are connected by direct highway road. That is cities C_1, C_2 are connected by direct road, cities C_2, C_3 are connected by direct road, and so on. But cities C_1, C_3 are not connected by direct road, and to travel from C_1 to C_3, one needs to hop through C_2. Among all cities, we are interested in determining the maximum number of hops required to travel from one city to the other where the destination city has more population than the source city. We will consider only one way journey from city C_i to C_j where $i < j$ and $W_i < W_j$ (i.e. from a low population to a high population city). We wish to find the maximum value of the difference $(j - i)$ for which $W_j > W_i$ among all possibilities.

For example, let us consider 7 cities whose populations (in integer) in Lakh are given as, City $= [90, 20, 10, 60, 70, 30, 80]$. In this case, the maximum difference is for indices $i = 1, j = 6$ (the respective entries are 20 and 80), and hence the number of hops required is $6 - 1 = 5$. If there is no answer possible, then we will print 'None'. For example if City $= [80, 70, 50, 40, 20, 10]$, then there is no such pair for which $W_i < W_j$ for $i < j$, so it returns 'None'. You will have to solve this problem in 2 ways:

(a) Find a brute force solution for the above problem. You will need 2 nested loops for this, which will yield a time complexity of $O(n^2)$.

(b) Next, improve the brute force solution so that the complexity of the problem reduces to $O(n)$. *Hint: Think of using an auxiliary array to store the intermediate maximum element of a subarray. Do not worry about the extra space required.*

Example:

```
Enter the number of cities: 10
Enter the populations: 120, 40, 70, 20, 90, 110, 20, 80, 10, 100
Maximum number of hops required = 8
```

Question-2

Fruit sellers are facing some problems of storing fruits in a traditional storage system where each worm-affected fruit results in spreading the worms and affecting the other fruits in the storage. The storage system is typically built as a row and column based positioning of each fruit (like a matrix). After enough investigation it is observed that worms affect mostly the other fruits in the same row and same column of the storage. Unfortunately the worms can't be detected until the fruit turns bad. In order to take preventive measures, the storage owner comes up with an idea to remove all the fruits in a row and column, if there is a single worm-affected fruit gets detected in that row or column. This will minimize the spread to other fruits to some extent. As a visualization of this idea, consider the following binary matrix where each element represents each fruit. If the entry for an element is '1' it means that the fruit is not yet detected as worm-affected, whereas an entry of '0' means that the fruit is affected:

$$\begin{pmatrix} 1 & 1 & \color{red}{0} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \color{red}{0} & 1 \\ 1 & 1 & 1 & 1 & 1 \\ \color{red}{0} & 1 & 1 & 1 & 1 \end{pmatrix}$$

Notice that there are 3 worm-affected fruits (marked in red). Now we will remove the other fruits in the same row and same column of affected fruits, and will mark '0' in those positions as well. Then the resulting matrix will be:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Now we wish to automatize this process efficiently by writing a program to do these replacements *without using any extra space for another matrix.* You will have to solve this problem in 2 ways:

(a) Find a naive solution to the above problem by traversing the matrix, checking for 0 value, marking the elements in the same row and column to some arbitrary value other than 0 or 1, and finally traverse the matrix once again and replace all elements with assigned value to 0. This solution will have a time complexity of $O(m \times n \times (m + n))$, where the matrix has the dimension of $m \times n$.

(b) Improve the time complexity of the solution by writing a program having complexity of $O(m \times n)$. *Hint: Check the first row and first column, and use these results in subsequent calculations.*