**Monika Szucs**

Due date: Feb 16, 2025

This file contains Lab 5. You must submit your answers to the D2L Dropbox "Lab-5"

1. Answers to question 1 below, written on this file**.**

2. A working java program for question 2.

Lab 5 requires Java programming. You can work in pairs (but you must still submit your own work to D2L).

**Note that late assignments will not be graded.**

1- The Trans-Canada Highway (TCH) follows a completely straight line as it crosses the Canadian prairies. Gas stations occur at various intervals. Assuming that the first station is a mile 0, and the last station is at mile n, find the distance between two closest stations. (The distance between two stations x and y is computed as |x − y|.)

Input is the mile maker locations of the gas stations, for example:

480 231 0 477 121 ... 1176 501 2000 (for n=Number of mile-markers/gas-stations)

Output for this example (based on the input we can see) would be: |480 – 477| = 3

A). Design and write pseudocode for a presorting-based transform and-conquer algorithm that solve this problem. (You can use the word “sort” and do not need to implement that method)

[2 mark]

ALGORITHM ClosestGasStation(A[0..n-1])

// Solve the closest gas station

// Input: an array A[0..n-1]

sort the array A // smallest number is the first in the array A[0] // sorting means nlogn

shortestDistance <- A[1]-A[0]

for i <- 1 to n-2 do

if A[i+1] - A[i] < shortestDistance

shortestDistance <- A[i+1] - A[I]

return shortestDistance

B). Design and write pseudocode for a brute-force algorithm that solve this problem. [2 mark]

ALGORITHM ClosestGasStation(A[0..n-1])

// Solve the closest gas station

// Input: an array A[0..n-1]

shortestDistance <- A[1]-A[0]

for i <- 0 to n-2 do

for j <- i+1 to n-1 do

currentDistance <- | A[i] - A[j] |

if currentDistance < shortestDistance then

shortestDistance <- currentDistance

return shortestDistance

C). Compare the efficiency of your algorithm for part A with your algorithm for part B. [1 mark]

Sorting means we are doing nlogn

Since we are using log in question part A that means A is more efficient. Sorting is nlogn. The loop takes O(n). So the overall efficiency is nlogn.

Whereas part B is a nested loop, its efficiency is O(n^2). Which means it will do significantly more operations, especially for larger inputs.

This means Part A is a lot more efficient.

2- Design and implement an algorithm that finds the smallest k numbers (in value) out of n numbers. For example, if given an array with eight numbers {4, 5, 1, 6, 2, 7, 3, 8}, return the least four numbers 1, 2, 3, and 4.

The algorithm that sort the n input numbers increasingly and returns the first K number is not acceptable for the answer. Since it needs to sort, its time complexity is. You should design more efficient algorithm using Max-heap. [5 mark]

Hint.

You should create a max-heap with capacity k that will contain the least k numbers out of n input numbers. A correct answer will be .

In Java, the PriorityQueue class is implemented as a priority heap. You can find a sample code in D2l that shows the basic operations of PriorityQueue. Note, for this question, I would suggest to pay extra attention to poll(). No need to implement a heap, feel free to use PriorityQueue.

Some students think a min-heap is the best solution as you would insert the entire array into the PriorityQueue then pop off ‘k’ elements. However, that would not be O(nlogk), it would be O(nlogn).