Due date:

This file contains Lab 6. You must submit your answers to the D2L Dropbox "Lab-6"

1. Answer for question 1 should be written on this file, due at the **end of lab today.**

2. A working java program for question 2 by the **end of today (11:30 pm).**

Lab 6 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.**

Please do not zip or compress your submissions. D2L allows you to upload multiple files

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=2000)

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. [2 mark]

public int transform(int a[]){

int distance = 0, minDistance = 0;

sort(a);

distance = Math.abs(a[0] - a[1]);

for(int i = 2; i < a.length - 1; i++){

minDistance = Math.abs(a[i] - a[i-1]);

if(minDistance < distance){

distance = minDistance;

}

}

return distance;

}

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

public int bruteforce(int a[]){

int distance = Math.abs(a[0] - a[1]);

int minDistance;

for(int i = 0; i < a.length; i++){

for(int j = i; j < a.length; j++){

minDistance = Math.abs(a[i] - a[j]);

if(minDistance == 0){

} else if(minDistance < distance)

distance = minDistance;

}

}

return distance;

}

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

Transform is faster than brute force as it only needs to go through the array once.

O(nlog n) < O(𝑛2)

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.

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 such as offer(), peek(), poll(), and size().