CS6013: Advanced Data Structures and Algorithms

Programming Assignment I (out of 10 marks)
(Start Date: 03 September 2021)
(Submission Deadline: 11:59 pm, Sunday, 12 September 2021)

0.1 Problem Description

Closest-Points Problem: We know that the Euclidean distance between any two points on a plane, say (a_1, b_1) and (a_2, b_2) is $\sqrt{(a_2 - a_1)^2 + (b_2 - b_1)^2}$. Write a program that takes as input from keyboard the value of a positive integer N, and the coordinates of N points on the 2-dimensional plane, namely $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The program then finds a closest pair of points and outputs this pair of points along with the distance between them. It is possible that two points have the same position; in that case, that pair is the closest, with distance zero. The program should print the output on the standard output. You may round real numbers to two decimal places.
Input/Output format explained using an example:
How many points are there on the 2D plane? 3
Enter the coordinates of Point 1
x_1 : 3
y_1 : 2
Enter the coordinates of Point 2
x_2 : 2
y_2 : 0
Enter the coordinates of Point 3
x_3 : 3
y_3 : 3.5
The closest pair of points are $(3, 2)$ and $(3, 3.5)$. The distance between them is 1.5 units.

0.2 Outline of the Algorithm to be Used

Go to Section 10.2.2 (page 470) of the book "Data Structures and Algorithm Analysis in C++ (DSAAC)", by MARK ALLEN WEISS (a soft copy of the book is available in our Google Classroom at Classwork → Books,Resources). You are expected to implement the divide and conquer algorithm for the "Closest-Points Problem" outlined in this section. This algorithm, given N points, finds a closest pair of points in $O(N \log N)$ time.

0.3 Program Related Instructions

1. You can write your program in one of C, C++, Java, or Python.
2. The main divide and conquer algorithm described in Section 10.2.2 of DSAAC involves sorting N numbers in $\Theta(N \log N)$ time. You are expected to write the code for a MERGE SORT algorithm from scratch for sorting purposes.

0.4 Submission Guidelines

1. Your submission will be one zip file named <roll-number>.zip , where you replace roll-number by your roll number (e.g. cs20mtech11003.zip), all in small letters. The compressed file should contain the below mentioned files:

(a) Programming files (please do not submit python notebooks or IDE files).

(b) A description of your solutions in pdf format named report.pdf. This file should describe the algorithms you use, pseudo-code is the preferred way to write out algorithmic concepts. If it is not an algorithm described in class, then you have to both describe it and argue that it is correct. You may state and used results proven in class without proof.

(c) Upload your zip file in Google Classroom at Classwork→Week 3→Assignment 1. No delays permitted.

2. The preferred way to write your report is using LATEX(Latex typesetting). However it is okay to submit pdf files not created in latex. **No handwritten file please!**

3. Failure to comply with instructions (file-naming, upload, input/output specifications) will result in your submission not being evaluated (and you being awarded 0 for the assignment).

4. **Plagiarism policy**: If we find a case of plagiarism in your assignment (i.e. copying of code, either from the internet, or from each other, in part or whole), you will be awarded a zero and will lead to a FR grade for the course in line with the department Plagiarism Policy (https://cse.iith.ac.in/academics/plagiarism-policy.html). Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded a zero for the submission. Apart from the book DSAAC, you may refer to internet sources like Wikipedia for clarification (since the problem is well-known), but the written explanation in the report and the implementation should be your own.

0.5 Evaluation Scheme

Your assignment will be awarded marks based on the following aspects:

- The logic in the code of the main divide and conquer $O(n \log n)$ algorithm - 3 marks.
- The logic in the code of the divide and conquer merge sort algorithm - 2 marks
- Code clarity (includes comments, indentation, naming of variables and functions, etc.) - 1 mark
- Perfect output - 2 marks
- Report (should contain a description of your program, correctness proof, and running time analysis. You don't have to explain the correctness and running time of well-known algorithms like merge sort or the algorithms we taught in class.) - 2 marks