CS-203 – ASSIGNMENT 2 – 2022 SUMMER

GIUSEPPE TURINI – KETTERING UNIVERSITY

INTRODUCTION

2D Closest-Pair Problem

The closest-pair problem consists in finding the two closest points (*i.e.*, the closest pair) in a given set S of n distinct points. [1]

This problem can be specified for various different geometric domains: one-dimensional (1D), two-dimensional (2D), three-dimensional (3D), etc. For simplicity, here we consider only the 2D closest-pair problem (see Figure 1).

Additionally, we assume that the 2D points in the input set S are specified by their Cartesian coordinates (x and y) as integer values, and that the distance between two 2D points is measured using the standard Euclidean distance. [2]

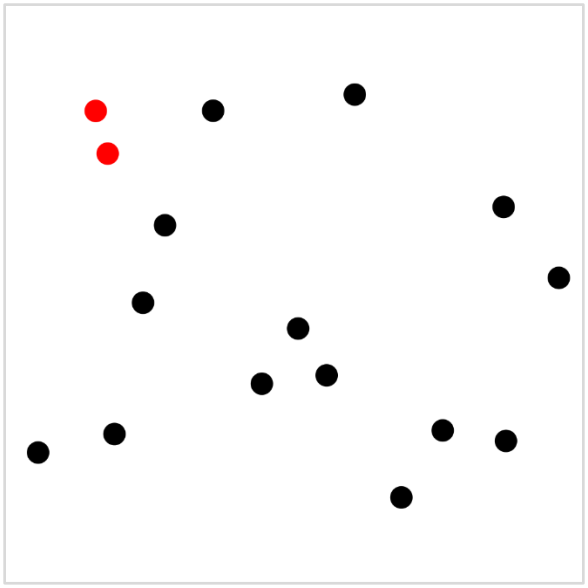

Figure 1. A 2D closest-pair problem, with 16 distinct points and solution in red.

[1] See: Wikipedia – Closest-Pair of Points Problem.

[2] See: Wikipedia – Euclidean Distance.

Solution by Exhaustive Search

The 2D closest-pair problem can be easily solved by an exhaustive search algorithm: arranging all possible pairs of points in the input set S, computing the associated distance for each of these pairs, and identifying the pair with the minimum distance.

Solution by Divide-and-Conquer

Another approach to solving the 2D closest-pair problem is by using a divide-by-2-and-conquer algorithm. This strategy begins by splitting the input set S into 2 halves, then solves these 2 smaller subproblems, and finally computes the final solution by analyzing the results of the subproblems already solved. [3] [4] [5]

ASSIGNMENT

Implement an Exhaustive Search Algorithm to Solve the 2D Closest-Pair Problem

Write a Java program which, given in input n distinct 2D points, finds a solution to the 2D closest-pair problem by exhaustive search, and prints the result (the points of the closest pair, with their coordinates and distance) to standard output.

Implement a Divide-and-Conquer Algorithm to Solve the 2D Closest-Pair Problem

Write a Java program which, given in input n distinct 2D points, finds a solution to the 2D closest-pair problem by divide-and-conquer (see attachment), and prints the result (points of the closest pair, with their coordinates and distance) to standard output.

Analyze the Time Efficiency of your Algorithms

Analyze your two algorithms performing both the theoretical analysis and the empirical analysis to evaluate the time efficiency. Each analysis should include: (1) a description of the analysis process (formulas, data, etc.), (2) a graph illustrating the results (units, approximation function, etc.), and (3) brief conclusions commenting the results. These analyses should be included in the technical report associated with this assignment, together with a final comparison/discussion of the efficiencies of your algorithms. [6]

[3] Levitin A. *"Introduction to the Design and Analysis of Algorithms (3rd Edition)."* Pearson, § 5.5. 2012.
[4] Johnsonbaugh R. and Schaefer M. *"Algorithms."* Pearson Education. 2004.
[5] Shamos M.I. and Hoey D. *"Closest-Point Problems."* In IEEE SFCS, pp. 151-162. 1975.
[6] Note: the analysis should include the initial sorting of input arrays required by the divide-and-conquer algorithm.

SUBMISSION

Deadline and Procedure

Before Monday of 11th week, send an email to the instructor (gturini@kettering.edu):

- Send the email using your Kettering email account (subject: "CS-203: Assignment 2").
- The email must include in attachment your assignment (2 files, see below).
- Late submissions will be assessed a 3% penalty (-3 points) for each day of delay. [7]

Content

The content of your submission must be:

- A Java file ("Solver2DClosestPair.java") with your code.
- A Technical Report ("CS-203-A2-Report.pdf") with your analyses (~3 pages in total).

EVALUATION

This is the form used to grade this assignment.

ASSIGNMENT 2 – EVALUATION FORM

Implementation, Performance, and Analysis (70/100)
Exhaustive Search Implementation/Performance (20/100)
Exhaustive Search Analyses (10/100)
Divide-and-Conquer Implementation/Performance (30/100)
Divide-and-Conquer Analyses (10/100)
Design, Style, and Submission (30/100)
OOP Design and Compilation Requirements (…/100)
Coding Style and Commenting (…/100)
Submission Procedure and Delay (…/100)

[7] Note: the maximum penalty for a late submission is -30 points.