Eric Agocs ITEC360 Program1

* how to compile and run your program

Compile: Javac Closest.java

Run: (example) Java Closest Brute 20 10 10

(example) Java Closest Both sample.txt

(Program is already compiled)

* your algorithms and how they operate

Brute force checks each point against every point and returns the shortest distance.

Divide and conquer starts by taking an arraylist, splitting it into two arrays, one sorted by ascending x-coords and one sorted by ascending y-coords. The arrays are put into the algorithm and split further by left and right x-y coords, making four arrays total. The four arrays are put through the algorithm recursively and a shortest distance is found based on left-right arrays. A minimum (dmin) is found and a mid-line is established. Points for which |x-m| < minimum are copied into array S[]. S[] is then used to find a new minimum until which the hypotenuse of all Y coords are greater than (dmin)2 and less than or equal to S[] size - 1. The new shortest distance must be smaller than (Xi - Xj)2 + (Yi - Yj)2.

* describes how you measured performance

Measured using System.currentTimeMillis() before and after running each algorithm and subtracting the end-time from the start-time.

* gives data that shows the performance (ie a table of dataset sizes and run times)

BRUTE FORCE DIVIDE AND CONQUER

|  |  |  |  |
| --- | --- | --- | --- |
| Numpoints | Runtime (miliseconds) | Numpoints | Runtime (miliseconds) |
| 50 | 1 | 50 | 2 |
| 100 | 4 | 100 | 4 |
| 200 | 6 | 200 | 6 |
| 300 | 22 | 300 | 8 |
| 400 | 37 | 400 | 11 |
| 500 | 46 | 500 | 15 |
| 1000 | 104 | 1000 | 30 |
| 1500 | 199 | 1500 | 37 |
| 3000 | 714 | 3000 | 82 |
| 5000 | 1947 | 5000 | 94 |

* an analysis of your performance results

Brute force conforms to n2 growth and divide and conquer conforms to nlogn growth.