**README.txt**

CPSC 3620

Dec 5, 2024

Hasan Raza, Connor Pittman. Chamod Jayathilake

**Introduction and Time Complexity**

This program implements two solutions to the closest pair problem, a brute force method, and a divide and conquer method. These algorithms return the closest pair, the distance between them, and the time it took them to discover them.

Generating the text files has a time complexity of O(N).

Performing the brute force algorithm on the coordinates has a time complexity of O(N2).

Performing the quick sort on the X coordinate, maintaining coordinate integrity has a time complexity of O(NlogN).

Performing the divide and conquer algorithm on the sorted coordinates has a time complexity of O(NlogN).

**Steps:**

Step 0 (optional):

Included in this program is a makefile, this readme assumes that the user is aware and can use makefiles. This makefile has been configured so that the user can generate a new set of points for all the available values of N. If the user chooses to do this, they must first delete the supplied outputN.txt files by using ‘make clean’. After all the files are deleted, the user can use ‘make’ to generate a new set of files.

Step 1:

When the user is ready to start the program, they should enter the command ‘make run’. This command compiles the project and prompts the user with a choice between two algorithms, “Divide and Conquer”, and “Brute Force”. If the user inputs unacceptable responses too many times, the program will end, this is to prevent infinite loops. Once the user has input an acceptable response, the program proceeds with either “Divide and Conquer” or “Brute Force”.

Step 2:

The user will now be prompted to enter N. N is the number of points that the user wishes to use in the closest point problem with their chosen algorithm. There are twenty-one acceptable values for N, the user is shown them, and is asked to input one. Additionally, the user is asked to input ‘0’ if they would like to run their chosen algorithm on all values of N. This step has error catchers to prevent unacceptable inputs from being passed into the system. If the user inputs too many unacceptable inputs in a row, the program will end. If the user input ‘0’, step 3 will repeat 21 times, each with a different N.

Step 3 (brute force):

Once the user has input ‘0’ or an acceptable value of N, the brute force algorithm will attempt to solve the closest pair problem with N points. It does this using two for loops to compare each point with every other, calculating the distance between them, and saving it if it is the new minimum distance. Once the brute force algorithm has finished, the program outputs the minimum distance, the two closest points, and the time it took to run the algorithm.

Step 3 (divide and conquer):

Once the user has input ‘0’ or an acceptable value of N, the divide and conquer algorithm will attempt to solve the closest pair problem with N points. Before it does that however, it needs to presort the points by their X coordinate while maintaining the point integrity. This is because attempting to use divide and conquer on an unsorted array of points is not reliable and. Once the points are sorted. The divide and conquer algorithm begins, points are compared, and the resulting minimum distance is returned, along with the two points that resulted in it.

Step 4:

Once the program has output the results of the algorithm, the program ends.

**Issues:**

This code is very robust and has a lot of error checking to ensure that the code does not encounter any unexpected input from the user. At the time of submission, there are no bugs in my code that I am aware of. One potential issue is that generating the text files uses the following code to store them.

std::filesystem::path base\_path = std::filesystem::current\_path() / "textfiles";

std::filesystem::path filename = base\_path / ("output" +

std::to\_string(acceptedNs.at(y)) + ".txt");

This has provided me with no issues so far, and as far as I can tell it works perfectly, however, should the filesystem::current\_path() be equal to an unexpected path, it could cause some issues with where the files are stored. However, since the files are also obtained by the algorithms using the same path, this may not cause any issues and could remain unnoticed by the user.