Documentation

Closest Points Application

Brief Summary

This a documentation for the Closest Points application. The application was made for a job interview test to solve the given problem. There are many multidimensional points in the space and this application finds that two points which have the least distance between each other.
The multidimensional points placed in a TSV file. The program read this file and create a new file for the result with the original line number of the two points in the input file and their coordinates in tab separated format.

1. Requirements

The application based on the java 8 version so the following libraries are important to build and run the application.

- JDK 1.8 (Oracle)
- JRE 8u77 environment

2. Usage

First the application has to be built. The entry point is the Main.class then run the jar file and give the input TSV file name.

```
Java -jar [jar file name] [input TSV file]
```

Then the program will generate automatically an output TXT file for the result.

3. Structure of application

The application contains 5 Java class:

- **Main** class defines the entry point of application and execute to solve the problem.
- **MultidimensionalPoint** class reperesents a multidimensional point.
- **ClosestPoints** class solve the problem and provides the two searched point.
- **Utils** class contains utility methods to read tsv file and export the result into a file.
- **DistanceMeasure** interface defines the Euclidean distance algorithms. Really this behave as a trait object. The interface is implemented by ClosestPoints class

4. Algorithm

Problem: There is a list of n-dimensional points in a TSV file. The two nearest points have to be found and give back by the algorithm.

Brute Force based solution

The implemented algorithms is brute force based. It iterate over the list or array and calculate the minimal distance with the other points during find the least distance. For the minimal distance calculation the euclidean algorithm used which uses coordinates of points.
Big-O-notation: O(n * n)

Divide and conquer based solution

This implementation will be done in the future. The current application does not contain this implementation. The solution is that the n-dimensional space should be reduced to smaller space with fewer n-dimensional points in order to minimize the list of points for the iteration.
Big-O-notation: O(n * log n)

5. Todos

This section contains the future implementation tasks:

- Support other file formats for input.
- Support to add the name of output file.
- GUI interface (JavaFX framework based)
- Junit tests
- Implementing a Divide and conquer based algorithm.
- Checking functionalities: empty input file, not numbers in file, etc.

References

https://en.wikipedia.org/wiki/Euclidean_distance