Summary on methods, algorithms, and data structures: For assignment 6

ASSIGNMENT NUMBER : 06
NAME : Dinesh Raj Pampati
ULID : dpampat
NAME OF SECRET DIRECTORY : namo

SUMMARY:

The Java program provided is an implementation of the closest pair of points problem, which aims to find the pair of points that are closest to each other from a given set of points in a two-dimensional space. The program defines a method of solving this problem efficiently using divide and conquer strategy and includes optimizations for handling larger datasets.

METHODS:

1. **compareTo(Coordinate o)**: Compares two coordinates based on their X and Y values.
2. **getDistance(Coordinate c1, Coordinate c2)**: Calculates the Euclidean distance between two points.
3. **bruteForce(ArrayList<Coordinate> coordinates)**: A brute force approach to find the closest pair when the number of points is small (3 or less).
4. **findClosestPair(ArrayList<Coordinate> xCoordinates, ArrayList<Coordinate> yCoordinates)**: The main recursive method that divides the problem into subproblems.
5. **stripClosest(ArrayList<Coordinate> strip, double delta, int resume)**: Finds the closest pair within a strip defined by a distance delta from the midpoint.
6. **main(String[] args)**: The entry point of the program, which processes the input file and orchestrates the closest pair finding process.

ALGORITHMS:

- **Divide and Conquer Approach:**
    1. Sort the set of points based on their X and Y coordinates.
    2. Split the set into two halves and recursively find the closest pair in each half.
    3. Find the closest pair across the divide (strip closest).
    4. Return the overall closest pair.
- **Brute Force (Base Case):**
    1. Calculate the distance between every pair of points.
    2. Keep track of the minimum distance and corresponding points.
- **Strip Closest:**
    1. Identify points within delta distance from the midpoint line.
    2. Compare the distances between these points in the vertical strip.
    3. Update the closest pair if a shorter distance is found.

DATA STRUCTURES:

- **Coordinate**: A nested class used to store the coordinates of the points with X and Y values.
- **ArrayList<Coordinate>**: Used to store the list of points and sort them according to their X or Y values.
- **static Coordinate closestPairPoint1, closestPairPoint2**: Static variables to keep track of the closest pair of points.

COMPLEXITY ANALYSIS:

- **Time Complexity:** The algorithm achieves **O(n log n)** complexity through the divide and conquer approach by recursively dividing the problem and efficiently finding the closest pair across the dividing line.
- **Space Complexity:** The space complexity of the algorithm is **O(n)** due to the storage of coordinate points and the temporary arrays (strips) used in the divide and conquer steps.

TESTING AND OUTPUT:

The code processes an input file containing a list of points, ignoring lines starting with "*" and terminates processing upon encountering a line starting with "-". For each set of points, it calculates the closest pair and prints out the points, their distance, and the time taken to compute.

CONCLUSION:

The Java implementation of the closest pair of points problem provided by me is effectively uses divide and conquer strategy to achieve an efficient solution. The report covers the methodologies, algorithms, data structures, complexity, and testing outcomes associated with the code.