Umar Khan

**Brute Force Algorithm (findClosestPointsBruteForce):**

The Brute Force algorithm compares every pair of points in the input array.

For n points, it performs n \* (n - 1) / 2 distance calculations.

Therefore, the time complexity of the Brute Force algorithm is O(n^2).

**Merge Sort Algorithm (mergeSort):**

Merge Sort is a divide-and-conquer algorithm that divides the input array into halves, sorts the halves recursively, and then merges them

The time complexity of Merge Sort is given by the recurrence relation T(n) = 2T(n/2) + O(n), where n is the size of the input array.

According to the Master Theorem, if a recurrence relation is of the form T(n) = aT(n/b) + f(n), where a >= 1, b > 1, and f(n) is a function, then:

If f(n) = O(n^c) where c < log\_b(a), then T(n) = Θ(n^log\_b(a)).

If f(n) = Θ(n^c \* log^k(n)), where c = log\_b(a), then T(n) = Θ(n^c \* log^(k+1)(n)).

If f(n) = Ω(n^c), where c > log\_b(a), and if af(n/b) <= kf(n) for some constant k < 1 and sufficiently large n, then T(n) = Θ(f(n)).

In Merge Sort, a = 2, b = 2, and f(n) = O(n).

Here, c = log\_b(a) = log\_2(2) = 1.

Since f(n) = O(n) = Θ(n^1), it falls under the second case of the Master Theorem.

Therefore, the time complexity of Merge Sort is O(n log n).

**Divide and Conquer Algorithm (findClosestPoints):**

The algorithm recursively divides the input array into halves until it reaches a base case.

At each level of recursion, it performs O(n) operations to find the closest pair of points in each half and then merges them together.

The recursion tree has a height of log n.

Each level of the recursion tree has a time complexity of O(n).

Therefore, the overall time complexity of the Divide and Conquer algorithm is O(n log n).

**Overall time complexity of the provided code is O(n log n)**