Zeus Polanco Salgado
CS 203 - Assignment 2: Technical Report
June 2021

Technical Report

Summary

The purpose of this report is to examine the performance of two algorithms theoretically and empirically. Both algorithms are solutions to finding the closest pair of points in a 2D set. One algorithm practices Exhaustive Search design technique while the other practices Divide and Conquer. A controlled factor to the efficiency of the algorithms is the input size, which we will consider to be the number of points in the 2D set. Another factor which affects efficiency in the Divide and Conquer algorithm is the value and order of the points in the set; this factor does not affect the Exhaustive Search algorithm since it calculates all possible distances between points.

Theoretical Analysis

Exhaustive Search

Right away it is apparent that the efficiency of the Exhaustive Search implementation will be $\Theta(n^2)$. Just like most brute force methods of comparing multiple elements in an array, the implementation contains a one for loop inside of the other. The basic operation will be considered to be the operation to calculate the distance between the points being searched and a comparison to determine if the distance is shorter than the known minimum distance, both of these operations rest within the innermost loop. The count formula can be represented as:

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 2 \rightarrow 2 \sum_{i=0}^{n-2} (n - i - 2) \rightarrow 2\left[\sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} i - \sum_{i=0}^{n-2} 2\right] \rightarrow 2\left[(n^2 - n) - (\tfrac{n^2-4n-4}{2}) - (2n - 2)\right]$$

$$\rightarrow (2n^2 - 2n) - (n^2 - 4n - 4) - (4n - 4) \rightarrow n^2 - 2n + 8 \in \Theta(n^2)$$

This proves that the Exhaustive Search algorithm is $O(n^2)$. No best/worst case scenario analysis required.

Divide and Conquer

The Divide and Conquer implementation is much more sophisticated than the previous algorithm since this algorithm takes advantage of other algorithms such as Quicksort and Mergesort. This algorithm is also recursive; the efficiency must be represented as a piecewise function rather than a pure summations. The analysis is broken down into portions and for simplicity. Since this algorithm includes sorting algorithms, the efficiency of is also affected by the value and order of the input set.

Hoare Partitioning

Basic operations: one iteration of a while loop

$$HoarePartionC(n) = \sum_{i=0}^{n-2} 1 + \sum_{i=1}^{n/2} (1 + \sum_{i=1}^{n-1} 1) \rightarrow 2n - 4$$

Quicksort

Basic operations: *if/else* blocks and HoarePartitioningC(n)

$$QuicksortC(n) = \{if\ n < 2,\ 1;\ if\ n = 2,\ 3;\ ifn > 2, 4n - 1 + QuicksortC(n - 1)\}$$

$$Solve\ for\ closed\ formula: 2n^2 + 21 \in O(n^2)$$

Mergesort

Basic operations: *if/else* blocks in and outside of while loops.

Assume sets are split evenly.

$$MergesortC(n) = \left\{if\ n < \ 2,\ 1;\ if\ n > 1, 1 + \frac{5n}{2} + 2MergesortC(n/2)\right\}$$

$$Apply\ master\ theorem: a = b^d \rightarrow O(nlogn)$$

DivideAndConquer

Basic operations: *if/else* blocks in and out of for and while loops, and distance calculation.

Assume sets are split evenly.

$$DivideAndConquerC(n) = \{n = 2, 1; n = 3, 7; n > 3, 2n + n^2 + MergesortC(n) + 2DivideAndConquerC(n/2)$$

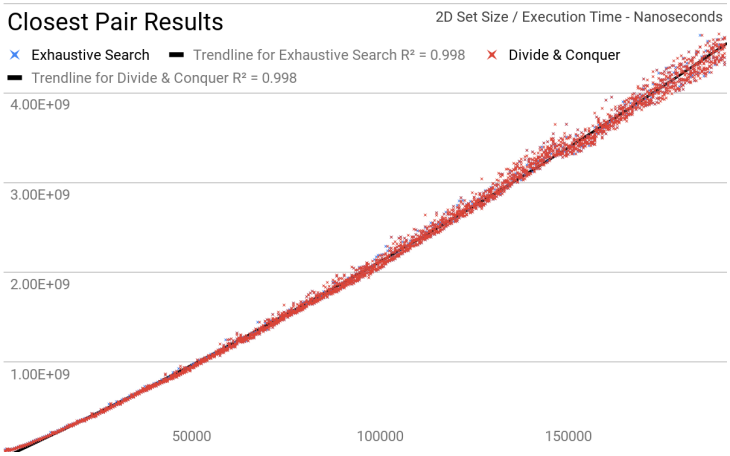
$$Apply\ master\ theorem: a < b^d \rightarrow O(n^2)$$

Full Implementation

$$T(n) = QuicksortC(n) + DivideAndConquerC(n) \in O(n^2)$$

Empirical Analysis

In order to fully evaluate the performance of the algorithms, evidence had to be gathered. The figure shown displays the results from tests given to both algorithms. The set of points generated for the tests were in the range of -50k to 50k with decimals. The most appropriate trend line for the results of both algorithms were polynomial estimations of the 2nd degree. Consider that both algorithms are implemented in Java and are tested on a 1.7 GHz Quad-Core Intel i7 processor.



Conclusion

In conclusion, the actual performance of the algorithms after being implemented were different as to the theoretical efficiency shown in the textbook, especially for the Divide and Conquer approach. It is difficult to observe how the Divide and Conquer algorithm outperforms the Exhaustive Search algorithm with the data recorded. Although the Exhaustive Search algorithm may not have much room for improvement, the Divide and Conquer implementation does. For the efficiency to change, factors that cause the algorithm to contain n^2efficinect must be eliminated such as the Quicksort implementation and finding a potential smaller distance near the median.