Report Closest Pair Lab

Valeria Andrea Jiménez Silvera 200135722

I. Abstract

The objective of this laboratory is to show the implementation of the Closest Pair Algorithm in a recursive way in order to find the closest pair in a set of coordinates in a plane of an array of N coordinates. For this test, an array is filled randomly with positive integers.

II. Introduction

To carry out this experiment, the **Closest Pair** algorithm will be executed recursively and non-recursively with different sizes **n** that allow us to see the complexity of the comparison time to find the closest pair in each iteration of size **n**.

III. Methods and Materials

For the development of this test, the ClosestPair algorithm is used to find the minimum distance between points of an array with coordinates **x** and **y**. The java method **Arrays.sort()** is used to organize each vector. After creating and organizing the vectors , an array with size **N** $\times$ **N** is created where the **x**'s coordinates are stored in the first row and **y**'s coordinates in the second row of the matrix. Finally, the matrix is divided into 2 new matrices, in which the coordinates resulting from the division of the original matrix are stored.

For this test the following materials were used:
- Github
- Netbeans IDE(Algorithm)
- Python(Graph)
- Latex (Documentation)

IV. Code

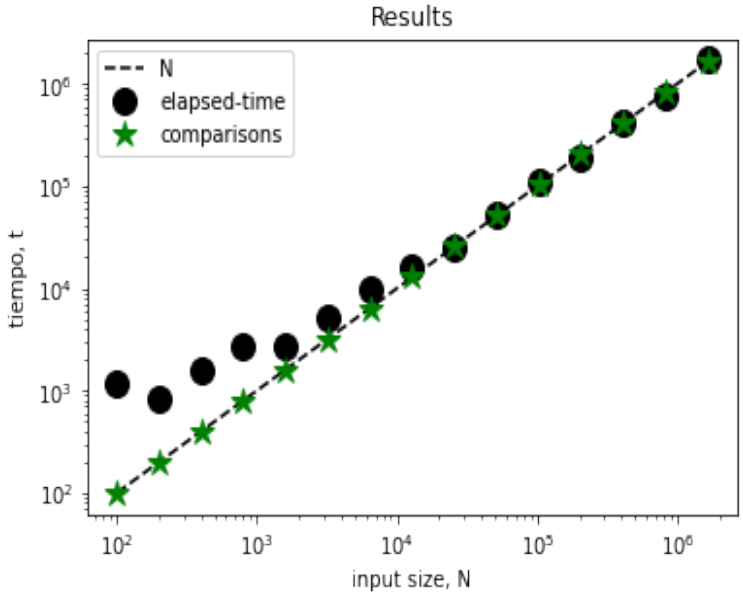
Algorithm 1 Bruteforce

$dmin \leftarrow INF$
for $i = [1, N-1]$ **do**
  for $j = [i+1, N-1$ **do**
    $d \leftarrow distance(coords, i, j)$
    if d $<$ dmin **then**
      $first \leftarrow i$
      $second \leftarrow j$
      $dmin \leftarrow d$
    end if
  end for
end for
$return(first, second, dmin)$

V. Results

After running the trial, a **.txt** file was generated to store the data obtained after the execution of the algorithm with different sizes of **N** separated into 3 different columns, with the size **N** (which it started at 100 and was increased by $N \cdot 2$ in each iteration) , the number of comparisons, and the execution time obtained in each one, which were used to generate a graph with the times with each size of N obtained previously. As can be seen below in the table with the execution times and the graph, the algorithm has a linear behavior $O(N)$.

N	Comparisons	Times
100	314	53604
200	627	51344
400	1269	75291
800	2551	78753
1600	5135	97707
3200	10213	235525
6400	20425	571905
12800	40734	750241
25600	81534	1699976
51200	163125	3191025
102400	325926	6400344
204800	652360	9976307
409600	1304821	21504484
819200	2609541	40926397
1638400	5218773	72677990



VI. Discussion

At the end of this test, it can be seen that the Brute Force Algorithm has a linear behavior and has a time complexity of N, giving it certainty that the execution time is proportional to the array's size(N), besides being able to confirm that the algorithm's complexity is $O(N)$.