I+D Report

Cristian Dj. Chavez Sarmiento - ID 200153656 Y Valeria Andrea Jimenez Silvera - 200135722 IST 4310-01

I. Abstract

During the course we have analyzed the time complexity of different algorithms and how it is related to the number of operations it performs. For this RD Project (**I+D**) we seek to analyze the time complexity of the algorithm Brute Force, implementing a data structure with a lower efficiency (**LinkedList**) as opposed to using **ArrayList** as it was done in the previous laboratory. The program randomly generates a series of coordinates of size **N** with respect to the **X** axis.

II. Introduction

To carry out this project we study the time complexity of the **ClosestPair** algorithm using **LinkedList**, to make a comparison with the previous lab where we used **ArrayList**, to observe how it has a less efficient behavior using **LinkedList** to find the smallest distance between **N** particles.

III. Methods and Materials

For the development of this project, a set of coordinates **X, Y** are created randomly and then stored in a list that is converted into a **LinkedList** for use. The coordinates stored in the list are then sorted with respect to the X axis. Then in the recursive BruteForce subroutine that uses the previously sorted list to find the closest pair. To know the behavior of the algorithm the data is stored in a .txt file with 3 columns, which are the size **N**, the number of comparisons, and their execution time in each iteration. This is done starting with a set of coordinates of size **50** up to a size of **20000** with an increase of **3/2**.

- Github
- Netbeans IDE(Algorithm)
- Python Colab(Graph)
- Latex (Documentation)

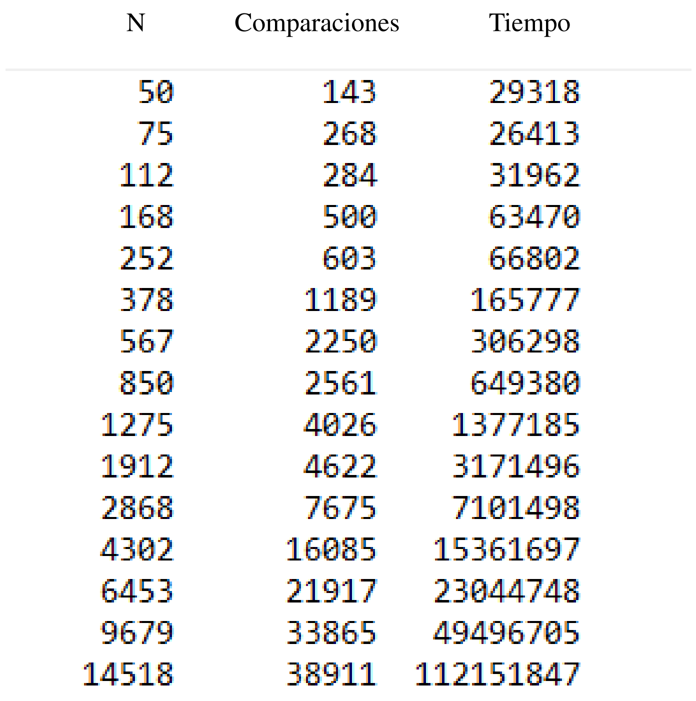
IV. Code

Algorithm 1 Brute force

```
$dmin \leftarrow INF$
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

For the results, a file **.txt** 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** , the number of comparisons, and the execution time obtained in each one, which were then used to generate a graph with the times with respect to size N obtained previously. As can be seen below in the table with the execution times and the graph, the algorithm has a quadratic behavior $O(N^2)$ as can be seen.



N	Comparaciones	Tiempo
50	143	29318
75	268	26413
112	284	31962
168	500	63470
252	603	66802
378	1189	165777
567	2250	306298
850	2561	649380
1275	4026	1377185
1912	4622	3171496
2868	7675	7101498
4302	16085	15361697
6453	21917	23044748
9679	33865	49496705
14518	38911	112151847

Figure 1. Results

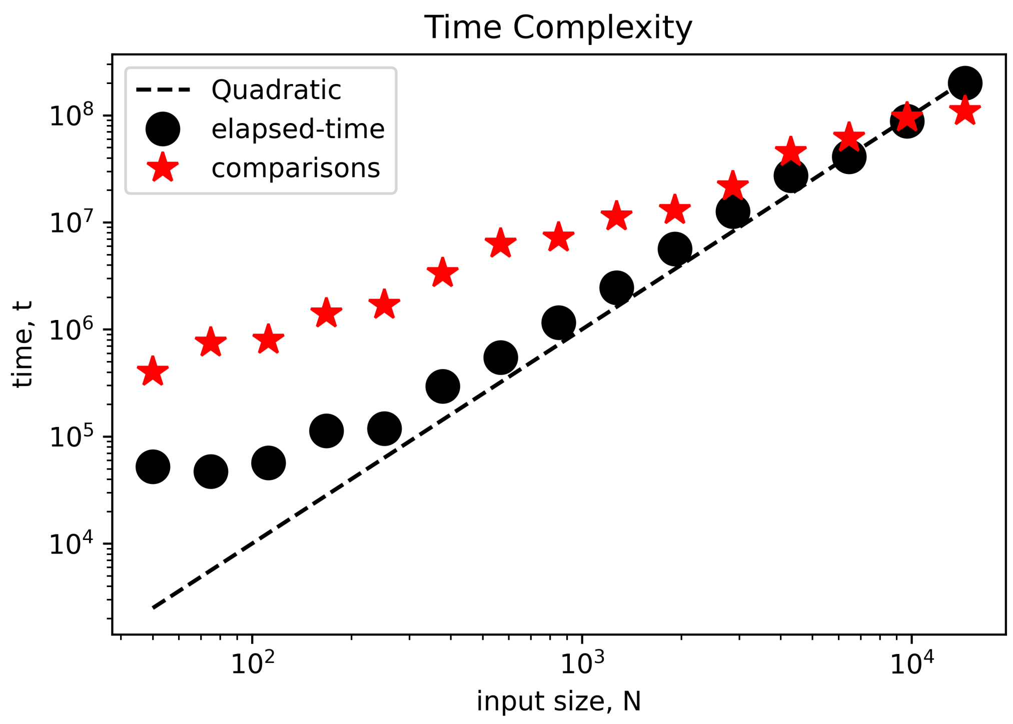


Figure 2. ALgorithm Complexity

As can be seen in the graph, the data have a quadratic behavior, as expected when using **LinkedList**.

VI. Discussion

Finally, we can conclude that the expected complexity was successfully met for this project. As we can observe from the results, the execution time and the comparisons are proportional to the size n of the coordinates. In addition, that the time complexity of the algorithm is $O(N^2)$ as expected. According to these results, we can compare the time complexity of the previous algorithm which was $O(N)$ in which **ArrayList** was used with the time complexity of this implementation which was $O(N^2)$ in which **LinkedList** is used as requested in this RD project. During the development of this project there were no serious drawbacks, since most of the algorithm was to reuse what was done in the past lab, making a few small changes so that it could work with **LinkedList**.