Closest Pair Algorithm*

Llach, Tabata
Departamento de Ingeniería de Sistemas
Universidad del Norte
Barranquilla, Colombia
tllach@uninorte.edu.co

***Resumen*—The purpose of this article is to study the performance, functioning, and time complexity of an algorithm that is in charge of organize and show which are the closest coordinates given a data set.**

***Index Terms*—coordinates, algorithm, complexity.**

I. INTRODUCCIÓN

Un algoritmo es un "conjunto ordenado y finito de operaciones que permite hallar la solución de un problema", al implementarlos hacer un analisis detallados del comportamiento de los mismos es indispensable para conocer el tiempo de complejidad y los casos que presenta y saber si el algoritmo es eficiente. El estudio de los algoritmos y el uso de las herramientas adquiridas por el estudio de ingenieria deberia ayudar a crear mejores algoritmos, mas eficientes.

II. DEFINICION DEL PROBLEMA

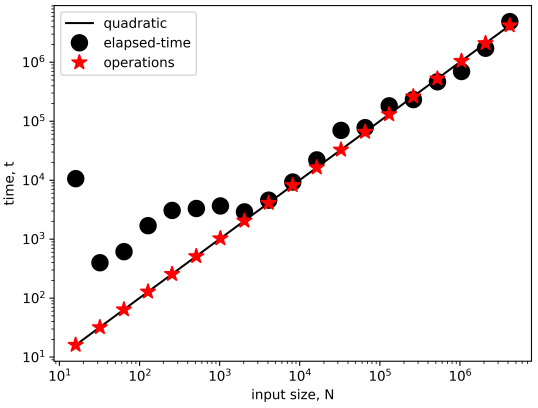
Hallar la distancia entre los puntos en un plano es importante para poder darle al usuario mejores respuestas en su busquedas. Este algoritmo a estudiar busca encontrarlo de la manera mas eficiente, rapida y que gaste menos recursos. Debido que ya se implemento el algoritmo *Brutal Force* pero este nos deja un $0(n^2)$, por eso se busca encontrar un algoritmo que resulva ese problema pero más eficiente y rapido.

III. METODOLOGIA

El algoritmo se realizó por medio de un software especializado llamado *IntelliJ IDEA*.
Las graficas fueron hechas en *Visual Studio Code* usando *Python*.
Para comprobar la hipótesis inicial del tiempo de complejidad se ejecutó el programa con diferentes tamaños.
Con tiempo de ejecución hacemos referencia en el tiempo que tarda el algoritmo en encontrar las coordenadas mas cercanas.
Para mayor precision de este experimento se deberia usar varios hardware para comparar las diferencias del tiempo dependiendo de las condiciones del equipo. Ahora, el algoritmo es el siguiente:
El programa divide a la mitad la lista de coordanadas, organizando los datos por X primariamente y secundariamente por Y. Despues de ello el algoritmo verifica que el numero de coordenadas en cada particion sea menor a 3 para usar el algoritmo implementado *BrutalForce*, de no serlo, seguira haciendo las particiones de forma recursiva. Este codido se puede encontrar en mi Github.

IV. RESULTADOS

Los resultados obtenidos son los siguientes:



Como se puede notar, la teoria aplicada a la vida real dan el mismo tiempo de complejidad, lo cual ahora con pruebas se puede decir que es $O(n)$.

V. DISCUSIÓN

Los valores de N muy pequeños el tiempo de ejecucion es mayor al esperado, esto se podria producir debido que la computadora esta usando recurso de otro lado y por eso la demora. Del resto de la grafica, los valores son mas o menos lo esperado, tenuendo una tendencia de ser alineado con la recta lineal

cuando los n (número de coordendas) son mas grandes. Ahora este algoritmo hace uso del algoritmo *Brutal Force* el cual tiene $O(n^2)$, como este es usado en N ppequeños, lo cual no afecta el rendimiento del algoritmo, ya que como vemos es lineal.

VI. CONCLUSIONES

Como *Brutal Force* es usado por el algoritmo analizado queda claro que *Brutal Force* puede ser usado para coordenadas pequeñas debido a su tiempo de ejecucion pero para tamaños (n) mas grandes es mejor usar un algoritmo como *Divide And Conquer* para las particiones de las coordenadas.

REFERENCIAS

[1] Time Elapsed: https://stackify.com/heres-how-to-calculate-elapsed-time-in-java
[2] Graphic creator code: . Diaz Maldonado, M. loglog-plot.py. https://github.com/misael-diaz/computer-programming/blob/main/src/io/java/loglogPlot.py (2022).