Worshop 3: Closest Pair*

Daniel Olivares
Universidad del Norte
Barranquilla, Colombia
daolivares@uninorte.edu.co

***Resumen*—The purpose of this article is to show the performance and functioning of an algorithm that is in charge of generating a file with random numbers and then counting how many times these are repeated this was made by experiments using a specialized software in which the algorithm mentioned was implemented and tested multiple times.**

INTRODUCCIÓN

En la actualidad es importante crear algoritmos que resuelvan problemas con mayor eficiencia. Como ingerieros de sistemas el analisis de como estos algoritmos funcionan y cual es su tiempo de respuesta es impresindible para la optimizacion de los mismos. Es por eso que el analisis de algoritmos como el de *Closet Pair* se hace necesario y fundamental hacerlo. El estudio de un algoritmo como Closest Pair con la implementacion de la lista enlazada, me ayuda a comprender mejor como los cambios de las estrucuturas de datos usadas en los algoritmos hace que cambien factores importante como lo es el tiempo.

DEFINICION DEL PROBLEMA

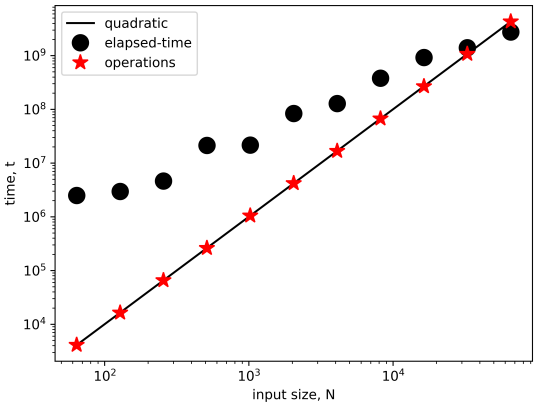
Un algoritmo es un "conjunto ordenado y finito de operaciones que permite hallar la solución de un problema", al implementarlos hacer un analisis detallados del comportamiento del mismo es indispensable para conocer el tiempo de complejidad y los casos en cuales te puedes encontrar al usasr el algoritmo para ver si este es eficiente.

METODOLOGIA

El algoritmo se realizó por medio de un software especializado llamado *Java Netbeans*. Las graficas fueron hechas en *Visual Studio Code* usando *Python*.
Se tuvieron en cuenta, diferentes tamaños y repeticiones por los mismo, para mas precision. El codigo completo se puede encontrar en mi repositorio de GitHub. https://github.com/daolivares/Linked-List-ClosestPair

RESULTADOS

Los resultados obtenidos son los siguientes:



Como podemos observar los resultados nos muestra que la complejidad de este algoritmo implementnado la listas enlazadas y los nodos, es el doble del anterior, siendo $O(n^2)$

DISCUSIÓN

Evidentemente, este algoritmo es medio eficiente que el anterior, en el cual usamos ArrayList y sus metodos. Podemos que este tiene el mismo time complexity que Brutal Force.

CONCLUSIONES

La mejor manera de encontrar el par mas cercano es usando el algoritmo de Divide y Venceras pero usando ArrayList y objetos que tengan las caracteristicas de las coordenadas.
Por otro lado, aunque el algoritmo con los tamaños usados, no nos demuestra exactamente lo aprendido en clase, si nos da un buen aproximado ya que al n tender a valores mas altos se va acercando al valor esperado. Para futuras investigacion, tendre en cuenta la revision y mejora de las funciones usadas para mejorar y mostrar el resultado esperado.
Además, concluyo que este tipo de analisis ayudan a mejorar los algoritmos al conduncir una prueba real aplicada.

REFERENCIAS

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