1. Introducción

Se debe realizar un software en el cual se implementará para lograr una optimización

El software deberá tener una GUI que permita generar números aleatorios, números

ordenados,permita mostrar el estado actual de los datos almacenados en memoria principal

y un frame que visualice los tiempos que se demora el programa en ordenar la secuencia de

datos utilizando distintos algoritmos. Se debe hacer distintas pruebas cuando el software esté

implementado estas son: Datos del orden entre 1 y 6 órdenes de magnitud. Los algoritmos de

ordenamiento para este proyecto son basado en comparación, por tanto tienen un costo en el

peor caso como máximo

Ω(

nlogn

)

. Esto se observa para MergeSort. En el caso de QuickSort

este algoritmo tiene costo en el peor caso de

O

(

n

2

)

, esto ocurre cuando los elementos ya están

ordenado. InsertionSort tiene costo en el caso promedio de

O

(

n

2

)

y en el mejor caso de

O

(

n

)

cuando los datos ya están almacenados. Este proyecto nos permitirá verificar estos resultados

no solo mediante la implementación de la GUI, sino que la implementación de los códigos en

un lenguaje como Java, nos mostrará como se van sumandos los distintos costos y por qué

tienen estos costos.

Como corresponde a un laboratorio del curso diseño y análisis de algoritmo de la carrera

de Ingienería en computación de la universidad de La Serena nos enfocaremos en explicar

y señalar cómo implementar estos algoritmos, con ello compararemos los métodos utilizados

por java para ordenar.

Se divide la implementación en insertionSort y métodos array.sort y collection.sort para

un desarrollador y la implementación de mergeSort y QuickSort para otro desarrollador. El

trabajo se debe efectuar en 6 días. En 4 días debemos tener las implementaciones de los 3

algortimos y la documentación de este plan general y de un informe con los requerimientos,

utilizando los modelos vistos en la clase de laboratorio. En los dos últimos días se deben

realizar la GUI y testear nuestro software.

2. Ámbito

Software desarrollado para un usuario. Software que visualiza datos ordenados en una GUI

utilizando datos aleatorios y datos ordenados para poder tener una aproximación heurística

de los costos de los algoritmos. Este software es llevado a cabo como parte de una tarea de

un curso de laboratorio de diseño y análisis de algoritmos por tanto su ámbito esta basado en

un usuario exclusivo ( el profesor). El tiempo máximo de respuesta no debe superar el orden

de horas para el caso de datos de ordenes de magnitud 6, esto porque consideramos que en el

peor caso tendremos un tiempo asintótico de

O

(

n

2

)

. La interfaz estará basada en frames de a librería Swing de Java y permitirá al usuario elegir en: tipo de datos generados ( números

aleatorios u ordenados), algoritmo de ordenamiento( insertionSort, mergeSort o quickSort) y

observar en el instante los tiempos estimados para cada algoritmo. La fiabilidad del software

es simple, ya que al ser una tarea no requeriremos de datos personales del usuario.

3. Alcance

Esto va dirigido a usuarios que requieran ordenar datos de distintos tamaños sin importar

la eficiencia de este ordenamiento, debido a que este software utiliza distintos algoritmos de

ordenamiento siendo su principal objetivo la comparación de estos algoritmos más que la

eficiencia del ordenamiento. Se podrán ordenar datos de tipo entero.

4. Restricciones

El software no ordena por otro tipo de datos que no sean enteros. Para la visualización se

decidirá solo mostrar los primeros 1000 elementos por pantalla, esto debido a que la llamada

al método

System.out.println

aumenta el costo en tiempo. El programa está diseñado

para ordenar arreglos y no listas. Solo como ejercicio de reforzamiento se implementará los

algoritmos de ordenamiento para listas, pero no serán utilizadas en la GUI. No permite

calcular tiempos estimados de ordenamiento para datos de orden 6 en magnitud, por lo que

las conclusiones serán hechas a partir de tiempos estimados con un máximo de 10000 datos.

5. Meta

Realizar un programa que permita ordenar una serie de datos aleatorios u ordenados

utilizando los algoritmos insertionSort, mergeSort, quickSort, para luego obtener tiempos

estimados del ordenamiento de cada uno y así concluir sus respectivas eficiencias.

6. Objetivos

El objetivo de este trabajo será poder implementar en una GUI una visualización de el

tiempo estimado que tienen los algoritmos de ordenamiento insertionSort, quickSort, mer-

geSort. Esto se hará visualizando en un frame los tiempos respectivos para un arreglo de

tamaño N dado por el usuario. El usuario debe ser capaz no solo de ver el arreglo ordenado,

sino que recibir en pantalla los tiempos obtenidos.

Utilizar los métodos de ordenamiento que utiliza Java array.sort y collections.sort. Esto

con el objetivo de comparar con los algoritmos implementado por este grupo de trabajo.

Estos métodos de la liberia

Java.util

no utilizan estos algoritmos, por lo que tendremos

que realizar una investigación para poder ver por qué Java utiliza otros algoritmos y no

los utilizados para este trabajo. La investigación requiere entender: TimSort y DoblePivote.

El primero como una implementación de InsertionSort y MergeSort, el segundo como una

variación en el número de pivotes de QuickSort.

Sabemos los costos asintóticos de los algoritmos, pero en este trabajo realizaremos una

interfaz que nos permita visualizar en nanosegundos los tiempos y así tener una idea de como

se comportan los algoritmos, esperamos ver el algoritmo más eficiente a array.sort, ya que este

implementa quicksort doble pivote, luego mergesort y deberiamos observar que para mayor

cantidad de datos insertionSort es un algoritmo que requiere de más recursos.