Informe técnico

por

Juan Pablo de Jesús Avendaño

Brayan Santiago González

María Acevedo Suárez

Parcial I

Estructura de datos y algoritmos II

Édison Valencia Díaz

PhD. en Ingeniería óptica por la UPC

Medellín

Universidad EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

2024

El presente escrito pretende realizar una comparativa de la complejidad temporal de los algoritmos de ordenamiento Bubblesort, Mergesort y Quicksort en una lista enlazada de 250.000 registros.

Datos registrados

A continuación, se presenta la tabla comparativa donde se incluye el tiempo de ejecución para cada algoritmo con un criterio de comparación dado. Note que las filas representan el criterio de comparación elegido, y las columnas representan el metodo de ordenamiento con el cual se ordenó la lista, la intersección entre filas y columnas corresponde al tiempo de ejecución del algoritmo seleccionado con el criterio de comparación dado, este tiempo se mide en **segundos**.

	Lexicográfico	Stock	Código	Precio
Bubblesort	8999.65	10979.29	10929.55	10952.19
Mergesort	0.45	0.34	0.35	0.34
Quicksort	13.7	1.21	0.55	0.57

Los datos registrados se obtuvieron a partir del promedio aritmético de 3 mediciones diferentes, este principio se siguió para los algoritmos Mergesort y Quicksort (El algoritmo Bubblesort fue descartado para realizar este principio ya que el tiempo de cada ejecución hacía que esta práctica se volviera cuanto menos engorrosa, dado esto se registró solo el primer dato obtenido).

El tiempo de ejecución de un algoritmo no solo depende del criterio de comparación y del propio algoritmo, sino también de las características de la maquina en el que se ejecuta. Por ello, a continuación, se detallan las especificaciones del entorno en el que se ejecutaron los algoritmos.

Prestaciones de la máquina

La máquina utilizada cuenta con un **procesador Intel Celeron 5205U 2-Core**, a su vez está equipado con una **memoria de acceso aleatorio de 4GB DDR4**, tiene unos **gráficos integrados Intel UHD** y finalmente el sistema operativo utilizado fue **Ubuntu**, en su versión **20.04.6 LTS**.

Observaciones y conclusiones

Analizando los datos obtenidos se puede ver que el tiempo de ejecución del algoritmo Mergesort es óptimo en cada uno de los criterios de comparación. A su vez el algoritmo Quicksort presenta resultados satisfactorios aun cuando su implementación es deficiente. Finalmente vemos que el rendimiento del algoritmo Bubblesort es realmente decepcionante, siendo la opción para descartar en cuanto a algoritmos eficientes se refiere, pero cabe recalcar que la dificultad en la implementación es mínima, lo cual se hace facil de aprender.

Se dice que el algoritmo Quicksort tiene una implementación deficiente dado que la elección del pivote no fue la más adecuada. El pivote elegido correspondía al último elemento de cada lista generada, este hecho contribuye a que las particiones de cada lista se encuentren desbalanceadas, consecuencia de esto el algoritmo se degrada hasta tener una complejidad $O(n^2)$. Para solucionar este problema se debe hacer una selección del pivote utilizando métodos más sofisticados, un ejemplo de esto es un pivote aleatorio.

Se experimentó también con el tiempo de ejecución para una lista ya ordenada para los algoritmos Mergesort y Quicksort, los resultados en el Mergesort fueron prácticamente iguales, a diferencia del Quicksort, el cual tuvo un rendimiento mediocre, este hecho se respalda con la reflexión realizada en el párrafo anterior, debido a que la lista está ordenada y el pivote es el último elemento, entonces las llamadas recursivas muy seguramente igualarían al número de elementos.

Es importante recalcar que los registros de tiempo de ejecución para el algoritmo Bubblesort están sesgados para los criterios de stock, precio y codigo, esto se debe a que el ordenamiento de las listas para estos criterios se hizo simultáneamente, lo que reduce la capacidad de la máquina para cada ordenamiento individual.

Finalmente, los autores del presente escrito eligieron al Mergesort como su algoritmo de ordenamiento preferido, no solo por su rendimiento óptimo, sino porque es un algoritmo estable, esto es, el orden relativo de elementos iguales se conserva.