Reflexión Actividad 1.3 (Evidencia)

Santiago Reyes Moran - A01639030 - 22/09/2021

Uso de Merge Sort

Para realizar esta actividad utilice el algoritmo de merge sort para ordenar el vector de objetos conforme su mes, día y hora. Este es uno de los algoritmos de ordenamiento más eficientes que hemos aprendido, junto con el quicksort, los cuales son los dos algoritmos de ordenamientos más recomendables para utilizar en esta actividad. Decidí utilizar merge sort en lugar del quicksort porque como se puede observar en la siguiente tabla, el mergesort siempre va a tener una complejidad temporal de $O(nlog(n))$, mientras que el peor caso del quicksort es de $O(n^2)$.

Tabla: Complejidad de los algoritmos de ordenamiento.

Algoritmo	Mejor	Promedio	Peor	Estable	Espacio
Swap sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	No	$O(1)$
Buble sort	$O(n)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	Sí	$O(1)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Sí	$O(n)$
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	Sí	$O(\log n)$

Es poco probable que el peor caso posible llegue a tocar con cualquiera de estos métodos, pero considero que es importante tomar en cuenta el peor caso. En resumen, utilicé el mergesort en lugar del quicksort debido a que la complejidad temporal del merge sort es constante a través de todos los casos mientras que el quicksort tiene una complejidad temporal más grande en el peor de los casos.

Es importante notar que el merge sort tiene una complejidad espacial “peor” que cualquiera de los otros algoritmos de ordenamiento que pudiéramos utilizar, pero para esta actividad le di prioridad a la complejidad temporal. Tomar en cuenta la complejidad espacial es de suma importancia cuando se están manejando cantidades gigantescas de datos, pero para este ejercicio no es el caso.

Uso de Búsqueda Binaria

El uso del algoritmo de búsqueda binaria para este ejercicio es algo bastante obvio, sobre todo cuando comparamos su complejidad temporal con la búsqueda secuencial, en donde se tendría que recorrer el arreglo completo para completar la búsqueda $O(log(n))$. Al tener grandes cantidades de datos (Como en este ejercicio en donde se tienen que crear 16807 objetos) es sumamente importante utilizar algoritmos eficientes, como lo viene siendo la búsqueda binaria $O(log(n))$.

De los algoritmos de búsqueda que cubrimos en la clase, el de búsqueda binaria es el más eficiente, por lo cual decidí implementarlo en mi programa.

Conclusión

Primeramente es importante notar que la complejidad total de la solución que programé para esta actividad es $O(nlog(n))$, debido al uso del algoritmo de mergesort; este resultado es bastante apropiado para la solución inclusive para casos con un número de entradas mucho mayores a las que se vieron en el caso de prueba (bitacora.txt).

Finalmente, considero que el trabajo realizado en clase sí me ayudó mucho para el desarrollo de esta actividad. No me había dado cuenta de la utilidad e importancia que estos algoritmos pueden tener en la vida real. Imaginar un sistema de consultas/reportes como el que se implementa en esta actividad me ayudó a comprender mejor la utilidad de los algoritmos cubiertos en clase y cómo podríamos hacer uso de ellos en nuestra vida profesional. Me quedaba muy claro que estos algoritmos son importantes para las entrevistas técnicas en compañías grandes, pero hasta ahora veo la importancia que tienen fuera de eso.