Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Sonora Norte

Act 1.2 - Algoritmos de Búsqueda y Ordenamiento

Curso:

Programación de estructuras de datos y algoritmos fundamentales

(TC1031)

Estudiante:

Daniel Alfredo Barreras Meraz A01254805

Docente:

Baldomero Olvera Villanueva

Fecha de entrega:

10 de septiembre de 2023

Código fuente

- Adjunto en el archivo zip.

Casos de prueba

Caso de prueba	**Entrada**	**Iteraciones intercambio**	**Iteraciones Burbuja**	**Iteraciones Merge**
1	[10, 4, 8, 12, 20, 15, 54, 18]	28	28	14

Caso de prueba	**Entrada**	**Posición**	**Iteraciones búsqueda secuencial**	**Iteraciones búsqueda binaria**
1	20	6	7	3
2	54	7	8	4
3	100	-1	8	4
4	12	3	4	1
5	8	1	2	2
6	4	0	1	3

Comprobación de casos de prueba

```
Comparaciones intercambio, burbuja, merge:
28 28 14

Posicion, iteraciones secuencial, iteraciones binaria:
6 7 3
7 8 4
-1 8 4
3 4 1
1 2 2
0 1 3
```

Explicación de complejidad algorítmica utilizando ecuaciones recurrentes

Ordenamiento intercambio: Este algoritmo compara cada par de elementos y los intercambia si están en el orden incorrecto. La ecuación recurrente para este algoritmo es T(n) = T(n-1) + n. Esto se debe a que cada pasada a través de la lista toma tiempo O(n) y hacemos n pasadas1. Por lo tanto, la complejidad de tiempo del algoritmo de intercambio es O(n^2).

Ordenamiento burbuja: Este algoritmo también compara cada par de elementos adyacentes y los intercambia si están en el orden incorrecto. Sin embargo, a diferencia del algoritmo de intercambio, el algoritmo de burbuja hace múltiples pasadas a través de la lista hasta que no se requieren más intercambios. La ecuación recurrente para este algoritmo es similar a la del algoritmo de intercambio: T(n) = T(n-1) + n2. Por lo tanto, la complejidad de tiempo del algoritmo de burbuja también es O(n^2).

Ordenamiento Merge: Este algoritmo divide la lista en dos partes iguales, ordena cada parte y luego las combina. La ecuación recurrente para este algoritmo es T(n) = 2T(n/2) + n3. Esto se debe a que el algoritmo divide la lista en dos partes iguales (cada una toma tiempo T(n/2)), y luego las combina (lo que toma tiempo n). Por lo tanto, la complejidad de tiempo del algoritmo Merge Sort es O(n log n).

Búsqueda secuencial: Este algoritmo examina cada elemento de la lista hasta que encuentra el elemento objetivo o hasta que ha buscado todos los elementos. La ecuación recurrente para este algoritmo es T(n) = T(n-1) + 14. Esto se debe a que cada búsqueda en la lista toma tiempo constante y hacemos n búsquedas en el peor caso. Por lo tanto, la complejidad de tiempo del algoritmo de búsqueda secuencial es O(n).

Búsqueda binaria: Este algoritmo busca un elemento objetivo en una lista ordenada dividiéndola repetidamente por la mitad hasta que encuentra el objetivo o hasta que la sublista es vacía. La ecuación recurrente para este algoritmo es T(n) = T(n/2) + 15. Esto se debe a que el algoritmo divide la lista en dos partes iguales (cada una toma tiempo T(n/2)), y luego realiza una comparación (lo que toma tiempo constante). Por lo tanto, la complejidad de tiempo del algoritmo de búsqueda binaria es O(log n).