



Tecnológico de Monterrey

Act 1.2 - Algoritmos de Búsqueda y Ordenamiento

Aaron Hernandez Jimenez

A01642529

28 de agosto del 2023

Programación de estructuras de datos y algoritmos fundamentales (Gpo 602)

Jorge Enrique González Zapata

El objetivo principal de esta actividad es comprender y aplicar diferentes técnicas de búsqueda y ordenamiento, así como analizar la complejidad temporal de estos algoritmos.

Algoritmos de ordenamiento

Método de Inserción

En cuanto a los algoritmos de ordenamiento, implementamos el Método de Inserción. Este algoritmo ordena una lista insertando cada elemento en su posición adecuada. Su complejidad temporal es $O(n^2)$ en el peor caso, por lo que es más adecuado para listas pequeñas.

Método de Burbuja

El Método de Burbuja es otro algoritmo de ordenamiento que analizamos. Este método compara y ordena pares de elementos adyacentes en la lista. Su complejidad temporal es $O(n^2)$ en el peor caso, lo que lo hace apropiado para listas pequeñas o como ejemplo didáctico.

Método de Merge

También nos adentramos en el Método de Merge, que utiliza la técnica de divide y conquista para ordenar una lista. Su complejidad temporal es $O(n \log n)$, lo que lo convierte en una opción eficiente para listas de cualquier tamaño.

Algoritmos de Búsqueda

Búsqueda Secuencial

Uno de los algoritmos de búsqueda que exploramos es la Búsqueda Secuencial. Este método consiste en recorrer secuencialmente cada elemento de una lista hasta encontrar el valor deseado. La complejidad temporal de la Búsqueda Secuencial es $O(n)$, donde 'n' es el número de elementos en la lista.

Búsqueda Binaria Iterativa

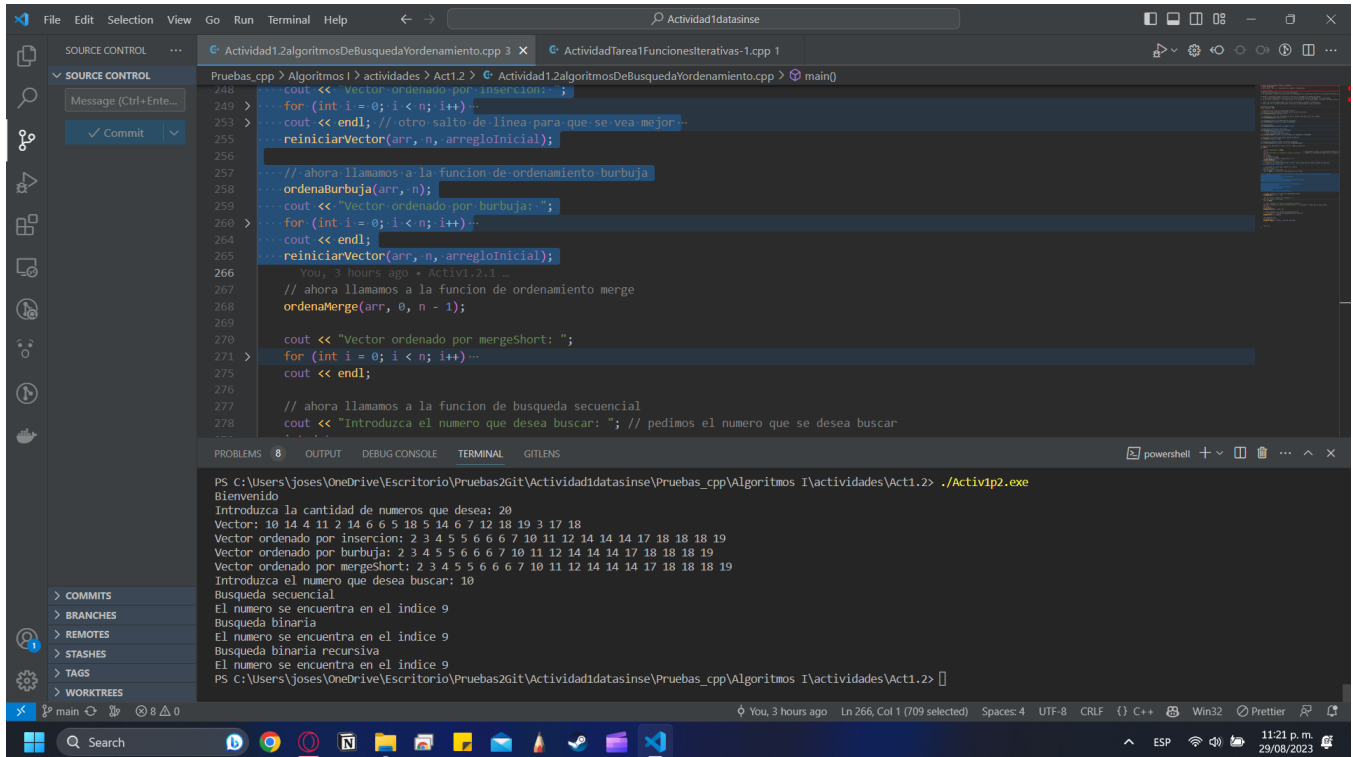
Otro algoritmo de búsqueda es la Búsqueda Binaria Iterativa, que se aplica en listas ordenadas. Este método divide repetidamente la lista por la mitad y compara el valor deseado con el elemento central. Su complejidad temporal es $O(\log n)$, lo que lo hace eficiente para listas grandes.

Búsqueda Binaria Recursiva

La Búsqueda Binaria Recursiva es una variante del método anterior, pero implementada de manera recursiva. Al igual que la versión iterativa, la complejidad temporal de la Búsqueda Binaria Recursiva es $O(\log n)$, lo que la hace igualmente eficiente para listas ordenadas.

Conclusión

La Actividad 1.2 me dio la oportunidad de profundizar en la implementación y análisis de una variedad de algoritmos de búsqueda y ordenamiento en C++. Aprendí mucho en esta actividad sobre el uso de las branches en git como un recurso adicional debido a que paralelamente en unos cursos particulares estoy llevando a cabo un diplomado del manejo de Github y Git, por lo que no únicamente aprendí lo necesario para poder llevar a cabo esta actividad sino que también otras muchas cosas importantes que se tenían que llevar a cabo.



```
File Edit Selection View Go Run Terminal Help
Actividad1datasinse
SOURCE CONTROL
Message (Ctrl+Ente...)
Commit
Pruebas.cpp > Algoritmos I > actividades > Act1.2 > C: Actividad1.2algoritmosDeBusquedaYordenamiento.cpp > main()
248 cout << "vector ordenado por insercion: ";
249 for (int i = 0; i < n; i++) {
253 cout << endl; // otro salto de linea para que se vea mejor
255 reiniciarVector(arr, n, arregloInicial);
256
257 // ahora llamamos a la funcion de ordenamiento burbuja
258 ordenaBurbuja(arr, n);
259 cout << "vector ordenado por burbuja: ";
260 for (int i = 0; i < n; i++) {
264 cout << endl;
265 reiniciarVector(arr, n, arregloInicial);
266
267 // ahora llamamos a la funcion de ordenamiento merge
268 ordenaMerge(arr, 0, n - 1);
269
270 cout << "vector ordenado por mergeSort: ";
271 for (int i = 0; i < n; i++) {
275 cout << endl;
276
277 // ahora llamamos a la funcion de busqueda secuencial
278 cout << "Introduzca el numero que desea buscar: "; // pedimos el numero que se desea buscar
...
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL GITLENS
PS C:\Users\joses\OneDrive\Escritorio\PruebasGit\Actividad1datasinse\Pruebas_cpp\Algoritmos I\actividades\Act1.2> ./Activip2.exe
Bienvenido
Introduzca la cantidad de numeros que desea: 20
Vector: 10 14 4 11 2 14 6 6 5 18 5 14 6 7 12 18 19 3 17 18
Vector ordenado por insercion: 2 3 4 5 5 6 6 6 7 10 11 12 14 14 14 17 18 18 19
Vector ordenado por burbuja: 2 3 4 5 5 6 6 6 7 10 11 12 14 14 14 17 18 18 19
Vector ordenado por mergeSort: 2 3 4 5 5 6 6 6 7 10 11 12 14 14 14 17 18 18 19
Introduzca el numero que desea buscar: 10
Busqueda secuencial
El numero se encuentra en el indice 9
Busqueda binaria
El numero se encuentra en el indice 9
Busqueda binaria recursiva
El numero se encuentra en el indice 9
PS C:\Users\joses\OneDrive\Escritorio\PruebasGit\Actividad1datasinse\Pruebas_cpp\Algoritmos I\actividades\Act1.2>
```