Universidad Rafael Landívar

Facultad de Ingeniería, Ingeniería en Electrónica y Telecomunicaciones

Laboratorio de Programación Avanzada

Sección: 02

Análisis de Complejidad Temporal de Métodos de Búsqueda aplicados en el proyecto InMemoryDB

Presentado por: Carlos Andrés Beltran Tello

Carné: 1001122

Catedrático: Ingeniero Abraham Soto

Fecha de Entrega: November 18, 2023

1 Objetivos

- **Específico**: Analizar detalladamente la complejidad temporal de los métodos de búsqueda binaria y secuencial implementados en el proyecto.
- **Medible**: Comparar la eficiencia de estos métodos cuantitativamente en términos de operaciones y tiempo de ejecución.
- **Alcanzable**: Proporcionar una base sólida para la elección de algoritmos de búsqueda adecuados para conjuntos de datos de diferentes tamaños y características.
- **Relevante**: Entender cómo la selección del algoritmo de búsqueda impacta el rendimiento general del software en escenarios de datos reales.
- **Temporales**: Completar el análisis y presentar los hallazgos antes de la fecha límite del proyecto.

2 Introducción

La eficiencia en el manejo y búsqueda de datos es fundamental en el desarrollo de software, particularmente en aplicaciones que manejan grandes volúmenes de información. Elegir el algoritmo de búsqueda correcto es crucial para optimizar el rendimiento y asegurar la escalabilidad. Este documento presenta un análisis detallado de la complejidad temporal de dos métodos de búsqueda comunes: la búsqueda binaria y la búsqueda secuencial, ambos implementados en el contexto de un sistema de gestión de datos.

3 Análisis de Complejidad Temporal

3.1 Búsqueda Binaria

Implementada en el método `binarySearchByValue`, la búsqueda binaria es un algoritmo eficiente para encontrar elementos en listas ordenadas. Su complejidad temporal es $O(\log n)$ [2]. En cada paso, el algoritmo divide el conjunto de datos por la mitad, lo que reduce exponencialmente el número de comparaciones necesarias. En el contexto del proyecto, esta técnica se aplica a conjuntos de datos ordenados almacenados en cada nodo de la lista doblemente enlazada, permitiendo búsquedas rápidas y eficientes incluso en grandes volúmenes de datos.

3.2 Búsqueda Secuencial

La búsqueda secuencial, implementada en `searchByValue`, revisa cada elemento de la lista hasta encontrar el valor deseado o hasta llegar al final de la lista. Su complejidad es $O(n)$ [1], lo que significa que el tiempo de búsqueda aumenta linealmente con el número de elementos. En nuestro proyecto, esta técnica se utiliza para buscar valores en conjuntos de datos no ordenados. A pesar de su simplicidad, la búsqueda secuencial puede ser ineficiente para grandes conjuntos de datos.

3.3 Inserción y Búsqueda en la Lista Doblemente Enlazada

La inserción en la lista doblemente enlazada (`insertNode`) y la búsqueda de nodos por clave hash (`searchNode`) tienen una complejidad de $O(n)$. Estas operaciones requieren en el peor de los casos recorrer toda la lista para encontrar la posición correcta o el nodo deseado. Aunque adecuadas para listas de tamaño moderado, estas operaciones pueden convertirse en cuellos de botella en aplicaciones con grandes volúmenes de datos.

4 Conclusiones

1. La elección del algoritmo de búsqueda tiene un impacto significativo en el rendimiento del software, especialmente en aplicaciones que manejan grandes conjuntos de datos. Mientras que la búsqueda binaria ofrece una eficiencia notable en listas ordenadas, la búsqueda secuencial, aunque más simple, resulta menos eficiente para conjuntos de datos grandes.

2. Las operaciones de inserción y búsqueda en la lista doblemente enlazada, aunque adecuadas para ciertos escenarios, deben ser cuidadosamente consideradas en aplicaciones de gran escala, donde su complejidad lineal puede ser un factor limitante.

3. Este análisis subraya la importancia de comprender las implicaciones de la complejidad algorítmica en la ingeniería de software, destacando la necesidad de elegir algoritmos apropiados basados en el tamaño y la naturaleza de los datos a procesar.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

[2] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 1998.