UNIVERSIDAD AMERICANA

Caso de Estudio: Sistemas de Búsqueda para una Biblioteca Digital Estudiantil

Estudiante:

Jose Augusto Salgado Murillo

Repositorio GitHub:

https://github.com/EtherealWeb5167/SistemaBusquedaBiblioteca

1. Introducción

El presente proyecto tiene como objetivo desarrollar un prototipo funcional de un sistema de búsqueda para una futura Biblioteca Digital Estudiantil. En el ámbito de la ingeniería de software y las ciencias de la computación, la capacidad de localizar y recuperar información de manera eficiente es un requisito crítico para la usabilidad de cualquier sistema. Este informe documenta la investigación teórica realizada sobre los algoritmos de búsqueda fundamentales, detalla la implementación técnica en lenguaje C# de algoritmos de búsqueda lineal y binaria, y presenta las conclusiones derivadas del trabajo colaborativo gestionado mediante el sistema de control de versiones Git y la plataforma GitHub.

2. Marco Teórico

Un algoritmo de búsqueda se define formalmente como un procedimiento computacional paso a paso diseñado para localizar un elemento específico, conocido como "target" u objetivo, dentro de una colección o estructura de datos (Cormen et al., 2022). Existen diversos enfoques para resolver este problema, siendo los más comunes en estructuras lineales:

Búsqueda Lineal: También conocida como búsqueda secuencial, es el método más intuitivo y directo. Consiste en recorrer la estructura de datos

elemento por elemento, desde el inicio hasta el final, comparando cada valor con el objetivo buscado. Su complejidad temporal es O(n), lo que implica que el tiempo de ejecución aumenta proporcionalmente al tamaño de la lista de datos.

Búsqueda Binaria: Es un algoritmo considerablemente más eficiente basado en la técnica de "divide y vencerás". Su funcionamiento consiste en dividir repetidamente el intervalo de búsqueda a la mitad. Si el valor buscado es menor que el elemento central, el algoritmo descarta la mitad superior y repite el proceso en la mitad inferior. Este método tiene una complejidad de O(log n), pero posee un requisito indispensable: la colección de datos debe encontrarse previamente ordenada (Bhargava, 2016).

3. Explicación de cada algoritmo implementado

Para satisfacer los requerimientos del caso de estudio, se desarrolló una aplicación de consola en C# que implementa cuatro módulos de búsqueda específicos:

En primer lugar, se implementó la **Búsqueda Lineal** para localizar libros por título. Se utilizó un ciclo iterativo *foreach* que recorre la lista completa de objetos `Libro`. El algoritmo normaliza las cadenas de texto a minúsculas para asegurar que la comparación sea insensible a mayúsculas y minúsculas, deteniendo la ejecución al encontrar la primera coincidencia.

En segundo lugar, se desarrolló la **Búsqueda Binaria** para localizar autores. Dado que este algoritmo requiere ordenamiento previo, se utilizó primero el método LINQ `.OrderBy` para organizar la lista alfabéticamente por el nombre del autor. Posteriormente, se implementó la lógica de división de intervalos mediante un ciclo *while*, calculando el índice medio y ajustando los límites superior e inferior según la comparación alfanumérica.

Adicionalmente, se incluyó una función de **Búsqueda de Extremos**. Esta función inicializa las variables de "mínimo" y "máximo" con el primer elemento de la lista y realiza un único recorrido lineal para comparar los años de publicación, identificando así el libro más antiguo y el más reciente en una sola pasada.

4. Conclusiones

Tras la finalización del desarrollo y la investigación teórica, se concluye que la elección del algoritmo de búsqueda adecuado es fundamental para el rendimiento del software. Mientras que la búsqueda lineal resulta sencilla de implementar y efectiva para listas pequeñas y desordenadas, la búsqueda binaria se vuelve indispensable a medida que el volumen de datos crece, ofreciendo tiempos de respuesta drásticamente inferiores a cambio de un costo computacional previo de ordenamiento.

Asimismo, la implementación de este proyecto permitió evidenciar la importancia de las herramientas de control de versiones. El uso de Git y

GitHub facilitó la organización del código mediante ramas, permitiendo desarrollar y probar características de forma aislada antes de integrarlas al proyecto principal, simulando un entorno de trabajo profesional real.

5. Referencias

Bhargava, A. Y. (2016). *Grokking Algorithms: An illustrated guide for programmers and other curious people*. Manning Publications.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (4.ª ed.). MIT Press.

Microsoft. (2023). *Documentación de C#: Colecciones y algoritmos*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/csharp/