

Búsquedas Internas

Universidad Distrital Francisco José de Caldas



**UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS**

Ingeniería de Sistemas

Profesor: Julio Cesar Florez Baez.

Brayan Estiven Aguirre Aristizabal - 20231020156

Lucia Avila Bermúdez - 20211020024

Juan Diego Contreras Yepes - 20211020069

Ciencias de la Computación II

Bogotá D.C

Algoritmos de Búsqueda Interna

Resumen Los algoritmos de búsqueda interna son herramientas fundamentales en la informática para localizar datos dentro de estructuras en memoria. Entre los métodos más utilizados se encuentran la búsqueda lineal, la búsqueda binaria, la búsqueda por interpolación y el uso de tablas hash. Cada uno de estos algoritmos presenta ventajas y desventajas dependiendo del contexto en el que se utilicen.

Introducción En informática, la eficiencia de un algoritmo de búsqueda es crucial para el rendimiento de un sistema. La selección de un método adecuado depende del tamaño de los datos, la estructura en la que se encuentran y la frecuencia con la que se realizará la búsqueda. En este trabajo se explican los principales algoritmos de búsqueda interna, detallando su funcionamiento, ventajas y desventajas.

Búsquedas Internas

Las búsquedas internas son operaciones críticas en computación para recuperar información almacenada en memoria. Son aquellas que se realizan dentro de estructuras de datos almacenadas en la memoria principal del sistema, sin necesidad de acceder a dispositivos de almacenamiento externos.

Tipos de Algoritmos de Búsqueda Interna

1. Búsqueda Lineal

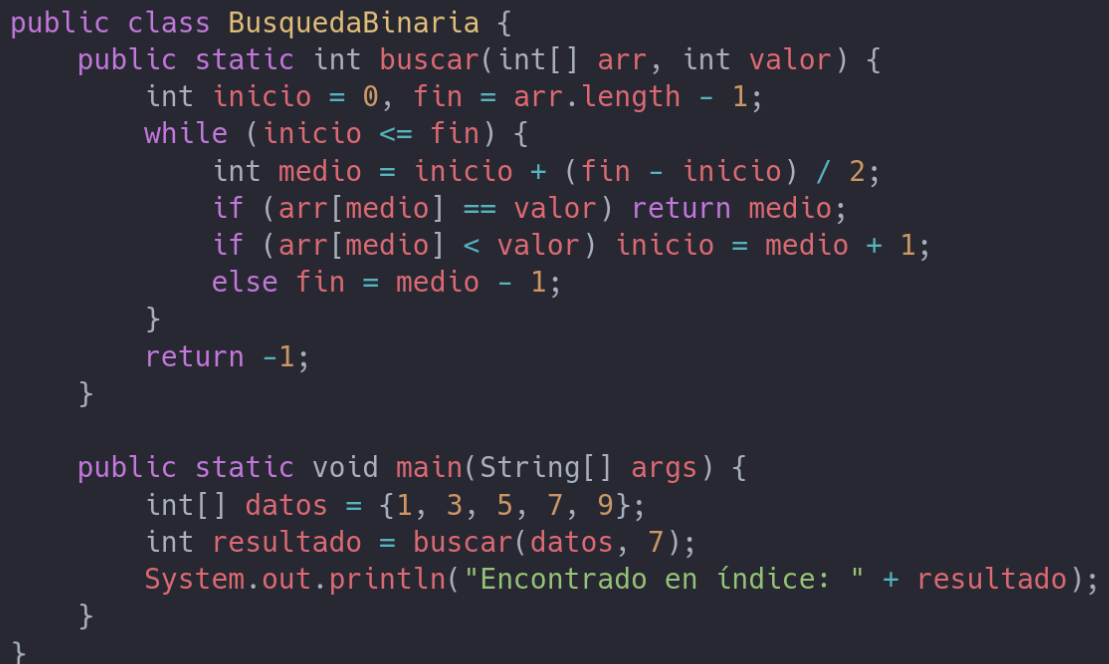
La búsqueda lineal consiste en recorrer secuencialmente una lista hasta encontrar el elemento deseado o llegar al final de la misma. Es un método simple pero ineficiente para grandes volúmenes de datos, ya que su complejidad temporal es $O(n)$ (Ponce, 2021)

“La búsqueda secuencial consiste en comparar secuencialmente el elemento deseado con los valores contenidos en las posiciones 1,...,n. El proceso termina cuando o bien encontramos el elemento o bien se alcanza el final del vector” (Álvarez, s.f)

```
public class BusquedaLineal {  
    public static int buscar(int[] arr, int valor) {  
        for (int i = 0; i < arr.length; i++) {  
            if (arr[i] == valor) {  
                return i;  
            }  
        }  
        return -1;  
    }  
  
    public static void main(String[] args) {  
        int[] datos = {3, 5, 7, 9, 11};  
        int resultado = buscar(datos, 7);  
        System.out.println("Encontrado en índice: " + resultado);  
    }  
}
```

2. Búsqueda Binaria

Requiere que la lista esté ordenada y reduce el espacio de búsqueda dividiéndolo en mitades de forma iterativa. Según Ponce (2021): “es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado. Compara el valor con el elemento en el medio del array, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre”



```
public class BusquedaBinaria {  
    public static int buscar(int[] arr, int valor) {  
        int inicio = 0, fin = arr.length - 1;  
        while (inicio <= fin) {  
            int medio = inicio + (fin - inicio) / 2;  
            if (arr[medio] == valor) return medio;  
            if (arr[medio] < valor) inicio = medio + 1;  
            else fin = medio - 1;  
        }  
        return -1;  
    }  
  
    public static void main(String[] args) {  
        int[] datos = {1, 3, 5, 7, 9};  
        int resultado = buscar(datos, 7);  
        System.out.println("Encontrado en índice: " + resultado);  
    }  
}
```

3. Búsqueda por transferencia de claves

Según Rodríguez Baena la búsqueda por transformación de claves consiste en aplicar una

función hash a una clave para obtener una posición de almacenamiento válida en un vector.

1. La **función hash** convierte la clave en un número entre 1 y n (siendo n el tamaño del vector).
2. Todos los elementos se almacenan usando la misma función hash.
3. Si las claves son consecutivas, se puede asignar directamente la clave a la posición, pero esto puede desperdiciar espacio si no todas las claves son correlativas.

```
public class BusquedaTransferencia {
    public static void main(String[] args) {
        HashMap<Integer, String> tabla = new HashMap<>();
        tabla.put(101, "Juan");
        tabla.put(202, "Maria");
        tabla.put(303, "Carlos");

        int claveBuscada = 202;
        if (tabla.containsKey(claveBuscada)) {
            System.out.println("Encontrado: " +
tabla.get(claveBuscada));
        } else {
            System.out.println("No encontrado.");
        }
    }
}
```

Conclusiones La elección del algoritmo de búsqueda más adecuado depende del contexto y las características de los datos. Mientras que la búsqueda lineal es simple pero ineficiente, la búsqueda binaria y la interpolación mejoran el rendimiento si los datos están ordenados. Por otro lado, las tablas hash ofrecen la búsqueda más rápida, pero requieren más espacio de memoria.

Bibliografía

- Álvarez Bravo, J. V. (s.f.). *ALGORITMOS DE BÚSQUEDA Y ORDENACIÓN*. Departamento de Informática – Departamento de Informática (ATC, CCIA, LSI). <https://www.infor.uva.es/~jvalvarez/docencia/pII%20antiguo/tema6.pdf>
- Ponce, J. (2021, 21 de febrero). *Algoritmos de búsqueda - Jahaziel Ponce*. Jahaziel Ponce. <https://jahazielponce.com/algoritmos-de-busqueda/>
- Rodríguez Baena, L. (2013). *Tema 1. Ordenación, búsqueda e intercalación interna*. Colimbo.net. [https://www.colimbo.net/documentos/documentacion/113/FPII01_Ordenacion_Busqueda_Intercalacion_\(12-13\).pdf](https://www.colimbo.net/documentos/documentacion/113/FPII01_Ordenacion_Busqueda_Intercalacion_(12-13).pdf)