Métodos de Búsqueda: Lineal y Hashing

Profesor Cosijopii García

November 26, 2024

Búsqueda Lineal

- La **búsqueda lineal** es un algoritmo de búsqueda que recorre una lista de elementos de manera secuencial, comparando cada elemento con el valor buscado.
- Es un método simple, pero no eficiente en listas grandes, ya que tiene una complejidad de tiempo de $O(n)$, donde n es el número de elementos en la lista.
- La búsqueda lineal se aplica principalmente en listas no ordenadas.

Búsqueda Lineal: Ejemplo

- Dada la lista: $\{3, 7, 2, 8, 6, 1\}$ y el valor a buscar: 8
- Se comienza en el primer elemento y se compara con el valor buscado.
- Si no coincide, se avanza al siguiente elemento hasta encontrar la coincidencia o llegar al final de la lista.

Pasos de la búsqueda lineal

- Compara 3 con 8 (no coincide).
- Compara 7 con 8 (no coincide).
- Compara 2 con 8 (no coincide).
- Compara 8 con 8 (¡coincide!).

Búsqueda Hash

- La **búsqueda hash** es un método eficiente para buscar elementos en una colección, utilizando una función hash.
- Una **función hash** toma un valor (como un número o una cadena) y lo mapea a un índice dentro de una estructura de datos, como un arreglo.
- El índice calculado nos permite acceder directamente al elemento deseado, lo que hace que la búsqueda sea rápida, con un tiempo promedio de $O(1)$ si se manejan correctamente las colisiones.

Método Hash

- La idea del **método hash** es usar una **función hash** para convertir un valor en un índice dentro de una tabla (arreglo).
- El índice nos permite acceder directamente al valor en la tabla, lo que acelera el proceso de búsqueda.
- En la práctica, los métodos hash son muy utilizados en estructuras como las tablas hash.

Función Hash

- Una **función hash** es un algoritmo que toma un valor de entrada y devuelve un índice en una tabla.
- Ejemplo de función hash:

$$\text{índice} = \text{valor}\%\text{tamaño de la tabla}$$

- Este índice determina dónde se almacenará el valor en la tabla.
- La función hash debe ser diseñada para minimizar las **colisiones**, es decir, que diferentes valores generen el mismo índice.

Ejemplo de Búsqueda Hash

- Supongamos que tenemos los siguientes números: $\{12, 24, 36, 48, 60\}$
- La tabla hash tiene un tamaño de 7, y utilizamos la función hash índice = número%7.

Pasos de la búsqueda

- Para el número 12, la función hash calcula $12\%7 = 5$.
- Para el número 24, la función hash calcula $24\%7 = 3$.
- Para el número 36, la función hash calcula $36\%7 = 1$.
- Para el número 48, la función hash calcula $48\%7 = 6$.
- Para el número 60, la función hash calcula $60\%7 = 4$.

Tabla Hash: Ejemplo

- Tabla Hash de tamaño 7 después de aplicar la función hash a los números:

Índice	0	1	2	3	4	5	6
Elemento	-	36	-	24	60	12	48

- Como podemos ver, los números han sido almacenados en sus índices correspondientes según la función hash.
- Y si queremos agregar el 38?

Colisiones en Hashing

- Las **colisiones** ocurren cuando dos valores diferentes generan el mismo índice en la tabla hash.
- Existen varios métodos para manejar las colisiones:
  - **Encadenamiento**: Almacenar los elementos que colisionan en una lista enlazada.
  - **Dirección Abierta**: Buscar el siguiente índice vacío en la tabla.

Conclusión

- La **búsqueda lineal** es sencilla pero ineficiente en listas grandes.
- La **búsqueda hash** es más eficiente para acceder a los datos, pero depende de una buena función hash.
- Las tablas hash son muy útiles en aplicaciones que requieren búsquedas rápidas.

Algoritmo de Búsqueda Hash

El algoritmo básico de búsqueda en una tabla hash con colisiones por encadenamiento es el siguiente:

```
Algoritmo BusquedaHash(tabla, clave):
    indice = hash(clave) % longitud de la tabla
    si tabla[indice] esta vacia:
        retornar "Valor-no-encontrado"
    sino:
        recorrer la lista enlazada en tabla[indice]
        mientras nodo no sea nulo:
            si nodo.valor == clave:
                retornar "Valor-encontrado"
            fin si
            nodo = nodo.siguiente
        fin mientras
        retornar "Valor-no-encontrado"
```