Metodos de busqueda

La búsqueda de un elemento dentro de un array es una de las operaciones más importantes en el procesamiento de la información, y permite la recuperación de datos previamente almacenados. El tipo de búsqueda se puede clasificar como interna o externa, según el lugar en el que esté almacenada la información (en memoria o en dispositivos externos). Todos los algoritmos de búsqueda tienen dos finalidades:

- Determinar si el elemento buscado se encuentra en el conjunto en el que se busca.
- Si el elemento está en el conjunto, hallar la posición en la que se encuentra.

Búsqueda lineal (secuencial)

Consiste en recorrer y examinar cada uno de los elementos del array hasta encontrar el o los elementos buscados, o hasta que se han mirado todos los elementos del array.

Este es el método de búsqueda más lento, pero si nuestra información se encuentra completamente desordenada es el único que nos podrá ayudar a encontrar el dato que buscamos. El siguiente algoritmo ilustra un esquema de implementación del algoritmo de búsqueda secuencial:

```
for(i=j=0;i<N;i++)
  if(array[i]==elemento)
  {
    solucion[j]=i;
    j++;
  }
```

Este algoritmo se puede optimizar cuando el array está ordenado.

Complejidad de la Búsqueda Lineal.

(A) MEJOR CASO: El algoritmo de búsqueda lineal termina tan pronto como encuentra el elemento buscado en el array. Si tenemos suerte, puede ser que la primera posición examinada contenga el elemento que buscamos, en cuyo caso el algoritmo informará que tuvo éxito después de una sola comparación. Por tanto, la complejidad en este caso será O(1).

(B) PEOR CASO: Sucede cuando encontramos X en la última posición del array. Como se

requieren n ejecuciones del bucle mientras, la cantidad de tiempo es proporcional a la longitud del array n, más un cierto tiempo para realizar las instrucciones del bucle mientras y para la llamada al método. Por lo tanto, la cantidad de tiempo es de la forma an + b (instrucciones del mientras * tamaño del arreglo + llamada al método) para ciertas constantes a y b, que representan el coste del bucle mientras y el costo de llamar el método respectivamente. Representando esto en notación O, O(an+b) = O(n).

(C) CASO MEDIO: Supongamos que cada elemento almacenado en el array es igualmente probable de ser buscado. La media puede calcularse tomando el tiempo total de encontrar todos los elementos y dividiéndolo por n:

Total = a (1 + 2 + ...+n) + bn = a (n(n+1) / 2) + bn, a representa el costo constante asociado a la ejecución del ciclo y b el costo constante asociado a la evaluación de la condición. 1, 2 , ..n, representan el costo de encontrar el elemento en la primera, segunda, ..., enesima posición dentro del arreglo.
Media = (Total / n) = a((n+1) / 2) + b que es O(n).

Este es el algoritmo de más simple implementación pero no el más efectivo. En el peor de los casos se recorre el array completo y el valor no se encuentra o se recorre el array completo si el valor buscado está en la última posición del array. La ventaja es su implementación sencilla y rápida, la desventaja, su ineficiencia.

Búsqueda binaria (dicotómica)

Si los elementos sobre los que se realiza la búsqueda están ordenados, entonces podemos utilizar un algoritmo de búsqueda mucho más rápido que el secuencial, la búsqueda binaria. El algoritmo consiste en reducir paulatinamente el ámbito de búsqueda a la mitad de los elementos, basándose en comparar el elemento a buscar con el elemento que se encuentra en la mitad del intervalo y en base a esta comparación:

Si el elemento buscado es menor que el elemento medio, entonces sabemos que el elemento está en la mitad inferior de la tabla.
Si es mayor es porque el elemento está en la mitad superior.
Si es igual se finaliza con éxito la búsqueda ya que se ha encontrado el elemento.
Se puede aplicar tanto a datos en listas lineales (Vectores, Matrices, etc.) como en árboles binarios de búsqueda. Los prerrequisitos principales para la búsqueda binaria son:

La lista debe estar ordenada en un orden especifíco de acuerdo al valor de la llave.
Debe conocerse el número de registros.

Referencias:

Estructuras de Datos y Algoritmos, Mark Allen Weiss, Addison-Wesley Iberoamericana, 1995, ISBN 0-201-62571-7.