Métodos de búsqueda

Con mucha frecuencia, los programadores trabajan con grandes cantidades de datos almacenados en arrays y registros, por lo que será necesario determinar si un array contiene un valor que coincida con un cierto valor clave. El proceso de encontrar un elemento específico de un array se denomina búsqueda. En esta sección se examinarán dos técnicas de búsqueda: lineal o secuencial, la técnica más sencilla, y binaria o dicotómica, la más eficiente.

Búsqueda secuencial o lineal

La búsqueda secuencial busca un elemento de una lista utilizando un valor destino llamado clave. En una búsqueda secuencial (a veces llamada búsqueda lineal), los elementos de una lista o vector se exploran (se examinan) en secuencia, uno después de otro. La búsqueda secuencial es necesaria, por ejemplo, si se desea encontrar a la persona cuyo número de teléfono es 958-220000 en un directorio o listado telefónico de su ciudad. Los directorios de teléfonos están organizados alfabéticamente por el nombre del abonado en lugar de por números de teléfono, de modo que deben explorarse todos los números, uno después de otro, esperando encontrar el número 958-220000.

El algoritmo de búsqueda secuencial compara cada elemento del array con la clave de búsqueda. Dado que el array no está en un orden prefijado, es probable que el elemento a buscar pueda ser el primer elemento, el último elemento o cualquier otro. De promedio, al menos, el programa tendrá que comparar la clave de búsqueda con la mitad de los elementos del array. El método de búsqueda lineal funcionará bien con arrays pequeños o no ordenados.

Complejidad

La complejidad de la búsqueda secuencial diferencia entre el comportamiento en el peor y mejor caso. El mejor caso se encuentra cuando aparece una coincidencia en el primer elemento de la lista, por lo que el tiempo de ejecución es O(1). El peor caso se produce cuando el elemento no está en la lista o se encuentra al final de ella. Esto requiere buscar en todos los n términos, lo que implica una complejidad de O(n).

El caso medio requiere un poco de razonamiento probabilista. Para el supuesto de una lista aleatoria es probable que ocurra una coincidencia en cualquier posición. Después de la ejecución de un número grande de búsquedas, la posición media para una coincidencia es el

elemento central n/2. El elemento central se obtiene después de n/2 comparaciones, que definen el coste esperado de la búsqueda. Por esta razón, se dice que la prestación media de la búsqueda secuencial es O(n).

Búsqueda Binaria

La búsqueda secuencial se aplica a cualquier lista. Si la lista está ordenada, la búsqueda binaria proporciona una técnica de búsqueda mejorada. Una búsqueda binaria típica es la búsqueda de una palabra en un diccionario. Dada la palabra, se abre el libro cerca del principio, del centro o del final dependiendo de la primera letra de la palabra que busca. Se puede tener suerte y acertar con la página correcta pero, normalmente, no será así y el lector se mueve a la página anterior o posterior del libro. Por ejemplo, si la palabra comienza con "J" y se está en la "L" se mueve uno hacia atrás. El proceso continúa hasta que se encuentra la página buscada o hasta que se descubre que la palabra no está en la lista.

Una idea similar se aplica en la búsqueda en una lista ordenada. Se sitúa la lectura en el centro de la lista y se comprueba si nuestra clave coincide con el valor del elemento central. Si no se encuentra el valor de la clave, se sitúa uno en la mitad inferior o superior del elemento central de la lista. En general, si los datos de la lista están ordenados, se puede utilizar esa información para acortar el tiempo de búsqueda.

Complejidad

El mejor caso se presenta cuando una coincidencia se encuentra en el punto central de la lista. En este caso, la complejidad es O(1), dado que sólo se realiza una prueba de comparación de igualdad. La complejidad del peor caso es O(log2n), que se produce cuando el elemento no está en la lista o el elemento se encuentra en la última comparación. Se puede deducir intuitivamente esta complejidad. El peor caso se produce cuando se debe continuar la búsqueda y llegar a una sublista de longitud 1. Cada iteración que falla debe continuar disminuyendo la longitud de la sublista por un factor de 2. El tamaño de las sublistas es:

n n/2 n/4 n/8 ... 1

La división de sublistas requiere m iteraciones, y en cada iteración el tamaño de la sublista se reduce a la mitad. La sucesión de tamaños de las sublistas hasta una sublista de longitud 1 será:

n n/2 n/22 n/23 n/24 .... n/2m

siendo n/2m = 1. Tomando logaritmos en base 2 a la expresión anterior tendríamos:

n = 2m

m = log2 n

Por esa razón, la complejidad del peor caso es O(log2n). Cada iteración requiere una operación de comparación:

Total comparaciones ≈ 1 + log2n

Comparación de ambos métodos

La comparación en tiempo entre los algoritmos de búsqueda secuencial y binaria se va haciendo espectacular a medida que crece el tamaño de la lista de elementos. Hay que tener presente que, en el caso de la búsqueda secuencial, en el peor de los casos coincidirá el número de elementos examinados con el número de elementos de la lista, tal como representa su complejidad O(n).

Sin embargo, en el caso de la búsqueda binaria, hay que tener presente, por ejemplo, que 210 = 1.024, lo cual implica el examen de 11 posibles elementos; si se aumenta el número de elementos de una lista a 2.048 y teniendo presente que 211 = 2.048 implicará que el número máximo de elementos examinados en la búsqueda binaria es 12. Si se sigue este planteamiento, se puede encontrar el número m más pequeño para una lista de 1.000.000, tal que:

2n ≥ 1.000.000

Es decir, 219 = 524.288, 220 = 1.048.576 y, por tanto, el número de elementos examinados (en el peor de los casos) es 21.

Bibliografía

Luis Joyanes Aguilar, Ignacio Zahonero Martínez (2008). Estructuras de datos en Java, de Mc Graw Hill

Morgado Jacome Eduardo #17211545