Búsquedas

El proceso de encontrar un elemento específico de un array se denomina *búsqueda*.

Se examinarán dos técnicas de búsqueda:

•*búsqueda lineal* o *secuencial*, la técnica más sencilla, y

•*búsqueda binaria* o *dicotómica*, la técnica más eficiente.

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.

•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.

•El método de búsqueda lineal funcionará bien con arrays o vectores pequeños o no ordenados.

•La eficiencia de la búsqueda secuencial es pobre, tiene complejidad lineal, $O(n)$.

Búsqueda binaria

•Se aplica a cualquier lista.

•Si la lista está ordenada, la *búsqueda binaria* proporciona una técnica de búsqueda mejorada.

•Básicamente se trata de, en una lista ordenada, situarse en el centro de la lista y se comprueba si la clave coincide con el valor del elemento central. Si no se encuentra el valor de la clave, se sigue la búsqueda en la mitad inferior o superior del elemento central de la lista.

•Al estar los datos de la lista ordenados se puede acortar el tiempo de búsqueda.

Ejemplo de búsqueda binaria

Se desea buscar el elemento 225 y ver si se encuentra en el conjunto de datos siguiente:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
13	44	75	100	120	275	325	510

El punto central de la lista es el elemento `a[3] (100)`. El valor que se busca es `225`, mayor que `100`; por consiguiente, la búsqueda continúa en la mitad superior del conjunto de datos de la lista, es decir, en la sublista,

a[4]	a[5]	a[6]	a[7]
120	275	325	510

Ejemplo de búsqueda binaria

```
a[4]    a[5]    a[6]    a[7]
120     275     325     510
```

Ahora el elemento mitad de esta sublista `a[5] (275)`. El valor buscado, `225`, es menor que `275` y, por consiguiente, la búsqueda continúa en la mitad inferior del conjunto de datos de la lista actual; es decir, en la sublista de un único elemento:

```
a[4]
120
```

El elemento mitad de esta sublista es el propio elemento `a[4] (120)`. Al ser `225` mayor que `120`, la búsqueda debe continuar en una sublista vacía. Se concluye indicando que no se ha encontrado la clave en la lista.

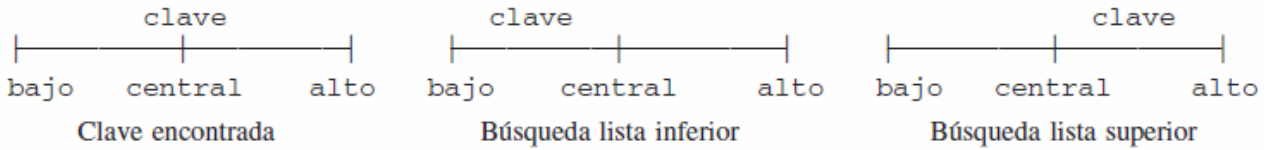
Algoritmo y codificación de la búsqueda binaria

Suponiendo que la lista está almacenada como un array, los índices de la lista son: `bajo = 0` y `alto = n-1` y `n` es el número de elementos del array, los pasos a seguir:

1. Calcular el índice del punto central del array

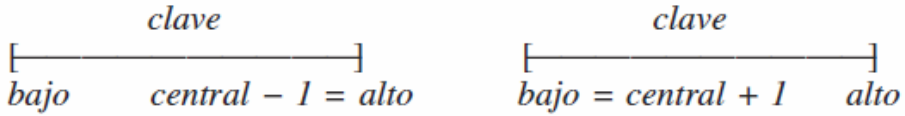
   `central = (bajo + alto)/2` (división entera)

2. Comparar el valor de este elemento central con la clave:



Búsqueda binaria de un elemento.

- Si `a[central] < clave`, la nueva sublista de búsqueda tiene por valores extremos de su rango `bajo = central+1..alto`.
- Si `clave < a[central]`, la nueva sublista de búsqueda tiene por valores extremos de su rango `bajo..central-1`.



El algoritmo se termina bien porque se ha encontrado la clave o porque el valor de `bajo` excede a `alto` y el algoritmo devuelve el indicador de fallo de −1 (*búsqueda no encontrada*).

Ejemplo

Sea el array de enteros `A (-8, 4, 5, 9, 12, 18, 25, 40, 60)`, buscar la clave, clave = 40.

1.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
-8	4	5	9	12	18	25	40	60

bajo = 0
alto = 8

↑ *central*

$$central = \frac{bajo + alto}{2} = \frac{0+8}{2} = 4$$

`clave (40) > a[4] (12)`

2. Buscar en sublista derecha

18	25	40	60

bajo = 5
alto = 8

↑

$$central = \frac{bajo + alto}{2} = \frac{5+8}{2} = 6 \quad (división\ entera)$$

`clave (40) > a[6] (25)`

3. Buscar en sublista derecha

40	60

bajo = 7
alto = 8

↑

$$central = \frac{bajo + alto}{2} = \frac{7+8}{2} = 7$$

`clave (40) = a[7] (40)` (*búsqueda con éxito*)

4. El algoritmo ha requerido 3 comparaciones frente a 8 comparaciones ($n - 1$, $9 - 1 = 8$) que se hubieran realizado con la búsqueda secuencial.

Conclusiones

• La búsqueda secuencial se aplica para localizar una clave en un vector no ordenado.

• Para aplicar el algoritmo de búsqueda binaria la lista, o vector, donde se busca debe de estar ordenado.

• La complejidad de la búsqueda binaria es logarítmica, **O(log n),** más eficiente que la búsqueda secuencial que tiene complejidad lineal, **O(n).**

Ejercicios

1. Ejecutar y analizar el código fuente dado.
2. Retomar el ejercicio y finalizar el punto 5 con el ordenamiento por burbujeo.
3. Agregar al mismo la función de búsqueda binaria de un elemento.

Fin