Estructuras y Tipos de Datos

Búsqueda

TEMA III - Búsqueda

CONTENIDO

- Introducción
- Búsqueda Secuencial
- Búsqueda Binaria

Búsqueda

Introducción

Búsqueda y ordenamiento son muy comunes en la programación, dado que gran parte del tiempo del procesamiento de datos se invierte en ellos

Su estudio se hace a objeto de analizar el tratamiento de las estructuras de datos y de determinar la eficiencia de los algoritmos utilizados

Búsqueda

Los procedimientos de búsqueda son frecuentes en el manejo de datos, de allí que gran cantidad del tiempo de procesamiento datos se consume en localizar los que se van a procesar.

Estrategias a analizar:

- *Búsqueda Secuencial*
- *Búsqueda Binaria.*

Objetivo: *es encontrar el elemento "x" en un arreglo "a".*

Búsqueda Secuencial

Estrategia

Inspección secuencial, por el sucesor, de los elementos almacenados en el arreglo hasta conseguir el elemento buscado o hasta agotarse todos ellos.

Ejemplo

Sea A un arreglo de N elementos, con N = 10

A

1	2	3	4	5	6	7	8	9	10
52	15	10	20	18	01	06	13	44	32

Si queremos localizar el número 18, el cuál se encuentra en la celda 5 del arreglo A, primero debemos inspeccionar las celdas 1 hasta la 4 en forma secuencial.

Búsqueda Secuencial

Algoritmo Búsqueda Secuencial


```
int bSecuencial(int value,int array[], int size){
    int i;

    for(i=0;i<size;i++)
        if(value==array[i])
            return i;
    return -1;
}
```

Si el arreglo no esta ordenado, no queda más que recorrer el arreglo desde el primer elemento hasta el último y cuando se encuentre el valor buscado retorna su posición.

Búsqueda Secuencial

Análisis de la Eficiencia

Complejidad

El orden del algoritmo búsqueda secuencial va a ser O(n) = n, ya que el ciclo que tiene se ejecuta a lo sumo n veces

El tiempo de ejecución va a depender en donde se encuentre el elemento buscado, si está entre los primeros términos del arreglo su tiempo T(n) será muy bueno, pero si está al final o no se encuentra el T(n) será pobre.

Búsqueda Secuencial

Mejora

Uso de Centinela

Elemento adicional que agregamos en el arreglo, bien sea al principio o al final, igual al elemento que estamos buscando, con la finalidad de evitar preguntar si ya hemos llegado al final del arreglo

Ejemplo

A

1	2	3	4	5	6	7	8	9	10	11
52	15	10	20	18	01	06	13	44	32	18

centinela

Búsqueda Secuencial

Búsqueda secuencial con centinela

```
int bSecuencialCentinela(int value,int array[], int size){
    int i;

    array[size]=value;
    for(i=0,value!=array[i];i++)
          ;
    if(i<size)
       return i;
    else
       return -1;
}
```

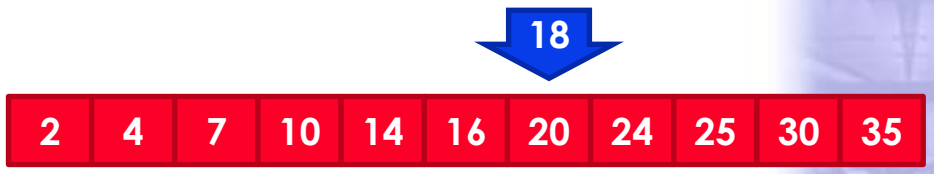
El **centinela** garantiza que se va a encontrar el elemento en el arreglo lo que simplifica la condición.

Sin embargo, en estos algoritmos, si el arreglo está ordenado resultan muy ineficientes.

Búsqueda Secuencial

Búsqueda secuencial con arreglo ordenado

```
int bSecuencialOrdenado(int value,int array[], int size){
    int i;

    for(i=0; i<size && array[i]<value; i++)
                ;
    if(i<size && value==array[i])
        return i;
    return -1;
}
```



Si el arreglo está ordenado no es necesario recorrerlo completamente. En cuanto el valor buscado es encontrado se retorna su posición y en el momento en que aparece un valor mayor que el buscado se puede detener la búsqueda. Pero la eficiencia …

Búsqueda Binaria

Estrategia

Búsqueda mucho mas rápida que la búsqueda secuencial

Requiere que los elementos del arreglo estén ordenados

Algoritmo para encontrar un elemento x en un arreglo A:

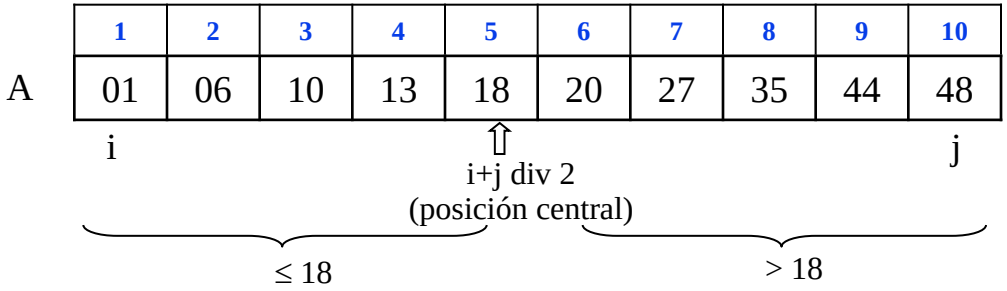
1. *Seleccionar el elemento que se encuentra en la posición central del arreglo A, de esta manera éste queda dividido en dos partes iguales.*

2. *Comparar el elemento del centro del arreglo A con el elemento x buscado, si no son iguales entonces el elemento x estaría a la derecha o la izquierda del arreglo A de acuerdo a si es mayor o menor al elemento de la posición central. Si es mayor repetimos los pasos 1 y 2 pero con la mitad del arreglo que se encuentra en el lado derecho, y si es menor lo haríamos con la mitad que se encuentra en el lado izquierdo.*

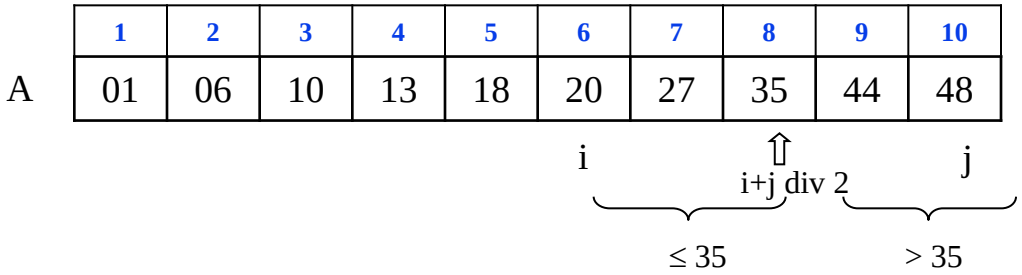
Búsqueda Binaria

Ejemplo

Sea A un arreglo de N elementos, con N = 10, en donde queremos localizar el número 35, el cuál se encuentra en la celda 8



35 está a la derecha del 18 entonces nos quedamos con la mitad del arreglo que esta del lado derecho.



Condiciones de parada

- *Contenido de la celda es igual al valor de la clave que estamos buscando,*
- *Los índices i y j se cruzan, resultado no exitoso, el elemento no se encuentra.*

Búsqueda Binaria

Algoritmo

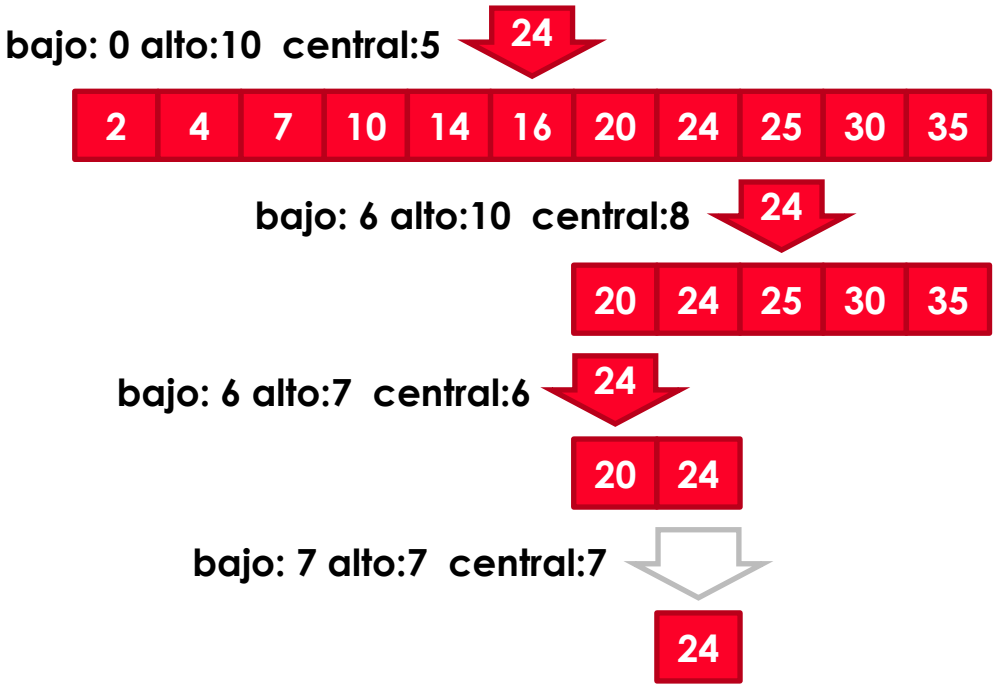
```
int bBinaria(int value,int array[], int size){
   int bajo=0,alto=size-1,central;

   while(bajo<=alto){
      central=(alto+bajo)/2;
      if(value == array[central])
          return central;
      if(value < array[central])
         alto=central-1;
      else
         bajo=central+1;
   }
   return -1;
}
```

Búsqueda Binaria

Algoritmo

Búsqueda binaria en un arreglo ordenado

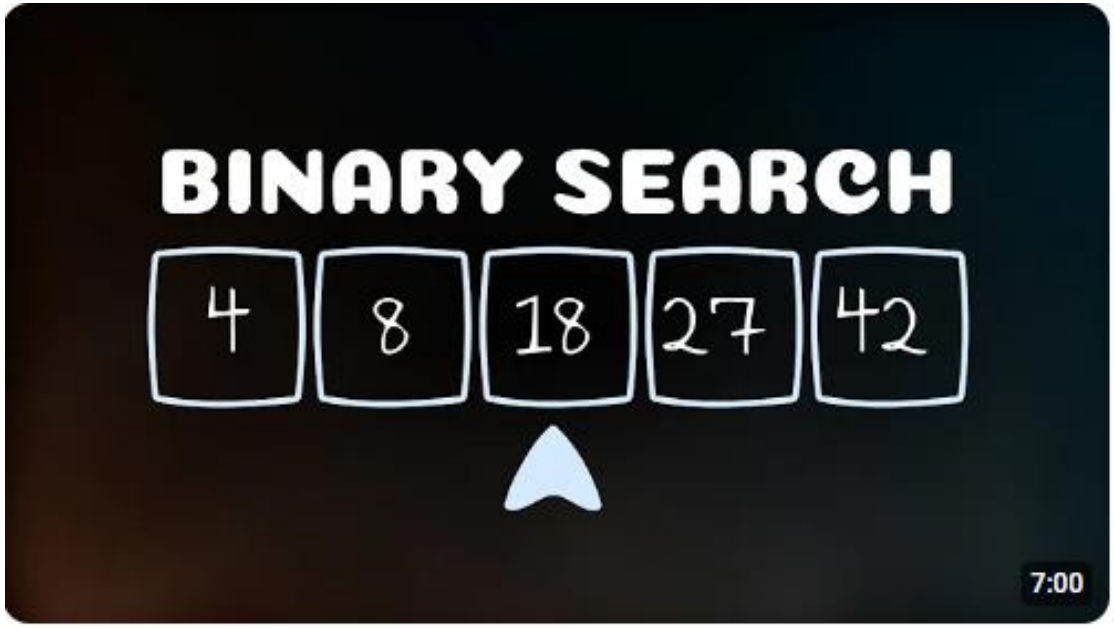
Buscar: 24




```
while(bajo<=alto){
    central=(alto+bajo)/2;
    if(value == array[central])
        return central;
    if(value < array[central])
        alto=central-1;
    else
        bajo=central+1;
}
return -1;
```

```
Resultado de buscar 24 en:
2 4 7 10 14 16 20 24 25 30 35
bajo: 0 alto:10 central: 5
bajo: 6 alto:10 central: 8
bajo: 6 alto: 7 central: 6
bajo: 7 alto: 7 central: 7
Con busqueda binaria es:7
```

Búsqueda Binaria



https://www.youtube.com/watch?v=eVuPCG5eIr4

Búsqueda Binaria

Análisis de la Eficiencia

Complejidad

Por cada iteración del WHILE se descarta la mitad de los elementos del arreglo, por lo tanto cuando hacemos una iteración nos quedan N/2 elementos,

1 iteración	$N/2$	$= N/2^1$
2 iteraciones	$N/4$	$= N/2^2$
3 iteraciones	$N/8$	$= N/2^3$
4 iteraciones	$N/16$	$= N/2^4$
iteración k		$= N/2^k$

donde $1 \leq N/2^k$

cuando $[N/2^k] = 1$ *paramos la búsqueda.*

Lo que queremos saber es cuantas veces iteramos, es decir k, despejamos.
Como k está en el exponente, aplicamos logaritmo de base 2.

- $Log_2 (N/2^k) \geq 0$
- $Log_2 (N) - k \geq 0$
- $k \leq Log_2 (N)$
- $k = [Log_2 (N)]$
- *Complejidad* $O(n) = Log_2 (N)$

Búsqueda Binaria

Consideraciones sobre los algoritmos de búsqueda

Si los elementos de la lista no están ordenados se debe recorrer la lista, hacer una búsqueda secuencial con centinela puede ahorrar muchas comparaciones, ahora si los elementos están ordenado en una búsqueda secuencial no es necesario recorrerlo completamente. Pero una comparación entre una búsqueda secuencial y una binaria se va haciendo espectacular a medida que crece el tamaño de la lista. Una búsqueda secuencial tiene una complejidad de O(n) mientras binaria O(log n).

N	N	Log N
10	10	3
100	100	7
1.000	1.000	10
10.000	10.000	13
100.000	100.000	17
1.000.000	1.000.000	20