Búsqueda Binaria

Bach. Rodolfo Mercado Gonzales
Universidad Nacional de Ingeniería

Búsqueda binaria

- Algoritmo que nos permite realizar búsquedas de manera eficiente.
- Se basa en el enfoque de “divide y vencerás”.

Buscando en listas ordenadas

El problema básico que nos permite entender este algoritmo es el de encontrar un número en una lista ordenada.

- **Target**: el valor buscado.
- **Espacio de búsqueda**: la lista

Buscando en listas ordenadas

Buscar el número 7 en la siguiente lista

1	5	7	12	14	18	21	31

Buscando en listas ordenadas

La forma trivial de resolverlo sería recorriendo todo el arreglo, pero nos tomaría $O(n)$, donde n es el tamaño de la lista.

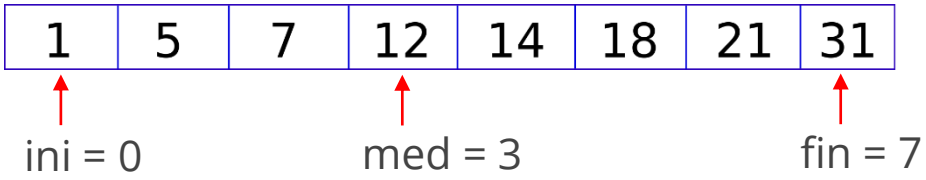
¿ Si necesitamos millones de búsquedas sobre la lista ?

Podemos aprovechar que la lista se encuentra ordenada y aplicar nuestro enfoque de “ divide y vencerás”.

Búsqueda binaria

Nuestro problema es buscar el número $x = 7$ en la lista de $n = 8$ elementos

Dividamos nuestro espacio de búsqueda en 2 partes: $[ini, med]$ y $[med + 1, fin]$



Búsqueda binaria

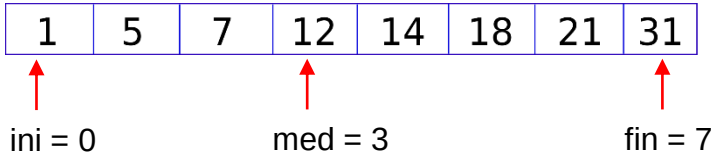
Antes de proceder a realizar el paso de vencer, podemos observar que:

$A[med] < x?$ → x se encontraría en la segunda mitad

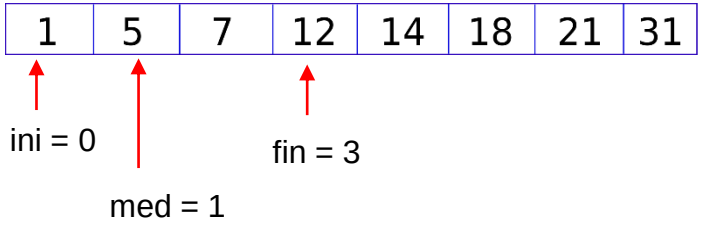
$A[med] \geq x?$ → x se encontraría en la primera mitad

Ahora solo es necesario buscar x en una de las 2 partes, es decir sólo debemos **vencer** recursivamente uno de los subproblemas.

Búsqueda binaria

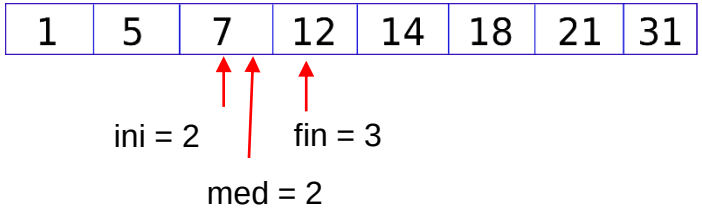


$A[med] < 7$? NO → $[0, 3]$

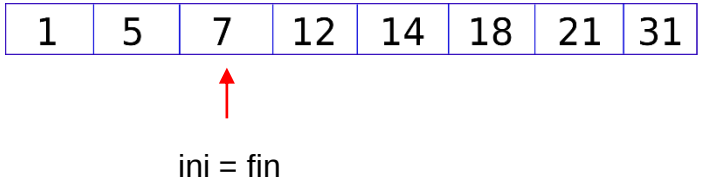


$A[med] < 7$? SI → $[2, 3]$

Búsqueda binaria



$A[med] < 7$? NO → $[2, 2]$



Caso directo, buscamos 7 en el rango $[ini, ini]$

Búsqueda binaria

```
int search( int A[], int ini, int fin, int x ){
    if( ini == fin ){ //caso trivial
        if( A[ ini ] == x ) return ini; //devolvemos posicion
        return -1; // no existe
    }
    int med = ( ini + fin ) / 2;
    if( A[ med ] < x ) return search( A, med + 1, fin, x );
    return search( A, ini, med, x );
}
```

Búsqueda binaria

También lo podemos hacer iterativamente.

```
int search( int A[], int ini, int fin, int x ){
    while( fin - ini > 0 ){ //mientras exista más de un elemento
        int med = ( ini + fin ) / 2;
        if( A[ med ] < x ) ini = med + 1;
        else fin = med;
    }
    if( A[ ini ] == x ) return ini;
    return -1;
}
```

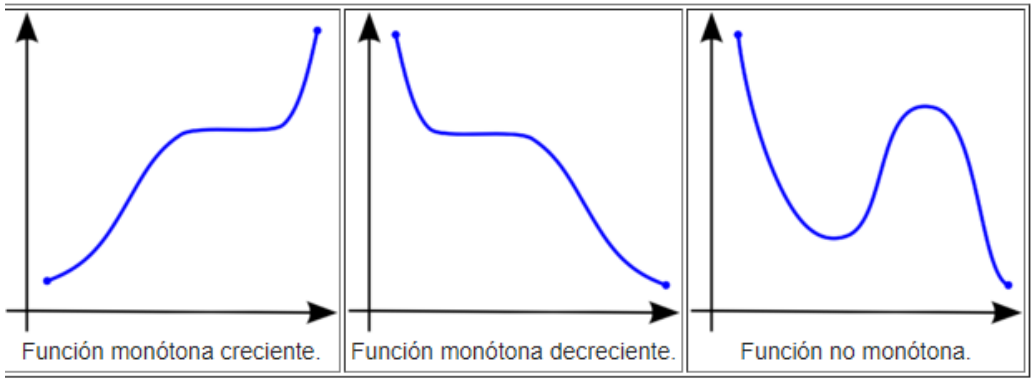
Consideraciones

- La complejidad de la búsqueda binaria es $O(\log n)$, ya que en cada paso se reduce el espacio de búsqueda a la mitad.
- Siempre revisar como se comporta nuestro algoritmo cuando el espacio de búsqueda tiene 1 o 2 elementos.
- La búsqueda binaria va más allá de las listas.

Función Monótona

- Una función en monótona si se cumple que:

$$x \geq y \rightarrow f(x) \geq f(y) \qquad o \qquad x \geq y \rightarrow f(x) \leq f(y)$$



Búsqueda binaria discreta

- La búsqueda binaria se puede aplicar a funciones monótonas cuyo dominio es el de los números enteros (espacio de búsqueda).

Dado un entero positivo n, indicar si es un cuadrado perfecto.

Trabajando con rango booleano

- Definamos como “predicado” a una función que tiene como dominio el espacio de búsqueda y que retorna un valor booleano (verdadero o falso).
- El predicado nos dirá si un elemento del espacio de búsqueda es solución o no (cumple todas la restricciones dada en el problema).
- Podremos usar búsqueda binaria cuando el predicado es monótono, por ende el rango de la función debe tener alguna de las siguientes formas:

falso falso … falso verdadero verdadero … verdadero

verdadero verdadero … verdadero falso falso … falso

Trabajando con rango booleano

- Ahora podemos usar este enfoque para encontrar el primer x para el cual nuestro predicado $\mathrm{p}(x)$ es verdadero o el último x para el cual nuestro predicado es falso.
- Es la manera más sencilla de enfocar los problemas de búsqueda binaria.

Dada una lista ordenada, encontrar la posición del menor número que es mayor a x

[10, 10, 10, 20, 20, 30, 30, 50, 80]

Problemas

UVA 11876 – N + NOD (N)

UVA 10856 – Recover Factorial

Búsqueda binaria continua

- La búsqueda binaria se puede aplicar a funciones monótonas cuyo dominio es el de los números reales (espacio de búsqueda).
- Al trabajar con número reales no encontraremos valores exactos.
- El algoritmo puede terminar cuando el espacio de búsqueda es menor que un límite predefinido (10^{-6}) o también se puede fijar el número de iteraciones (100).

Hallar la raíz cuadrada de un número n

Problemas

UVA 10341 – Solve It

¡ Good luck and have fun !