Divide y Vencerás

Bach. Rodolfo Mercado Gonzales
Universidad Nacional de Ingeniería

Divide y Vencerás

- Enfoque de resolución de problemas basado en la recursividad.
- En cada estado de la recursión realiza 3 pasos: dividir, vencer y combinar.

Divide y Vencerás

- **Dividir** el problema en una cantidad determinada de subproblemas, que son instancias más pequeñas del mismo problema.
- **Vencer** los subproblemas resolviéndolos de forma recursiva. Si el tamaño del subproblema es lo suficientemente pequeño, basta resolverlo de forma directa.
- **Combinar** la soluciones de los subproblemas para obtener la solución del problema original.

Divide y Vencerás

- Generalmente un problema se divide en 2 subproblemas de igual tamaño
- Si se divide en más partes la implementación puede resultar más compleja y la diferencia en complejidad no es considerable.

Exponenciación Rápida

Calcular a^b, donde a y $b \in Z+$

- Se puede hacer fácilmente en $O(b)$. ¿Qué pasa si nos hacen demasiadas consultas o el valor de b es demasiado grande?
- Apliquemos el enfoque divide y vencerás.

$$a^b \begin{cases} a^{\frac{b}{2}} * a^{\frac{b}{2}} & , b\ par \\ a^{\frac{b}{2}} * a^{\frac{b}{2}} * a & , b\ impar \end{cases}$$

Exponenciación Rápida

```
int fastPow( int a, int b ){
    if( b == 0 ) return 1;
    if( b % 2 == 0 ) return fastPow( a, b / 2) * fastPow( a, b / 2);
    return fastPow( a, b / 2) * fastPow( a, b / 2) * a;
}
```

5^4 ..... nivel 0

5^2 * 5^2 ..... nivel 1

5^1 * 5^1 5^1 * 5^1 ..... nivel 2 = log2 b

$O(b)$ **... Se puede reducir?**

Exponenciación Rápida

```
int fastPow( int a, int b ){
    if( b == 0 ) return 1;
    int temp = fastPow( a, b / 2);
    if( b % 2 == 0 ) return temp * temp;
    return temp * temp * a;
}
```

$\mathbf{5^4}$ …….. nivel 0

$\mathbf{5^2}$ * 5^2 …… nivel 1

$\mathbf{5^1}$ * 5^1 5^1 * 5^1 …… nivel 2 = log2 b

$$\boldsymbol{O(log\ b)}$$

Exponenciación Rápida

Como hallar la potencia puede hacer overflow rápidamente, el problema se redefine de la siguiente manera:

Calcular $a^b \% c$, donde a, b y $c \in Z+$

```
int fastPow( int a, int b, int c ){
    if( b == 0 ) return 1 % c;
    int temp = fastPow( a, b / 2, c );
    temp = ( temp * temp ) % c;
    if( b % 2 == 0 ) return temp;
    return  ( temp * ( a % c ) ) % c;
}
```

$O(log\ b)$

Problemas

UVA 374 – Big Mod

Multiplicación Rápida

Calcular $(a * b)\ \%\ c$, donde a , b y c $\in Z$ en el rango $[1, 10^{18}]$

Búsqueda Binaria

Buscar un valor x en un arreglo ordenado A

1	5	7	12	14	18	21	31

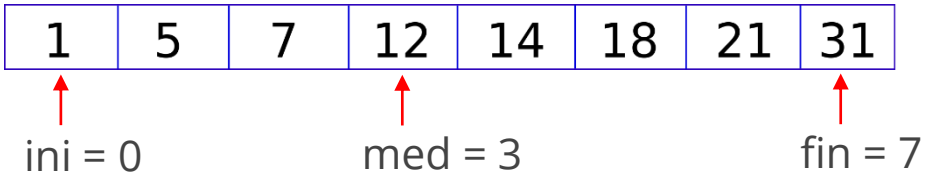
Búsqueda Binaria

- La búsqueda binaria usa gran parte del enfoque divide y vencerás.
- Notaremos que aparecen los pasos de dividir y vencer, pero el combinar no esta explícito.

Búsqueda Binaria

Nuestro problema es buscar el número $x = 7$ en el arreglo $A[0..n-1]$

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



Búsqueda Binaria

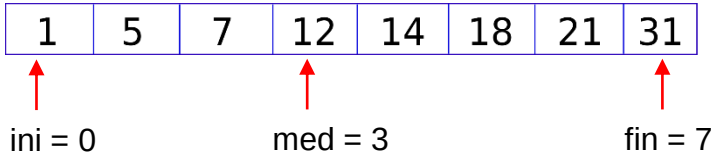
Antes de proceder a realizar el paso de vencer, podemos observar que:

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

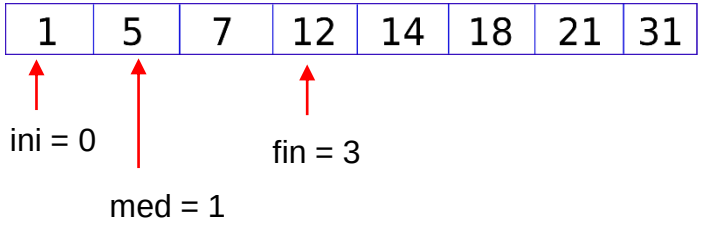
$A[med] \geq x?$ → x se encuentra en la primera mitad

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

Búsqueda Binaria

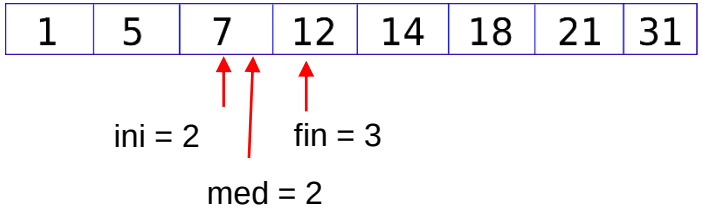


A[med] < 7 ? NO → [0, 3]

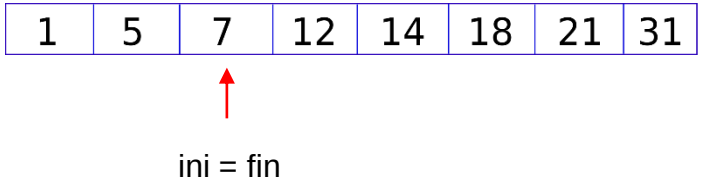


A[med] < 7 ? SI → [2, 3]

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 );
}
```

¡ Good luck and have fun !