

BÚSQUEDA BINARIA



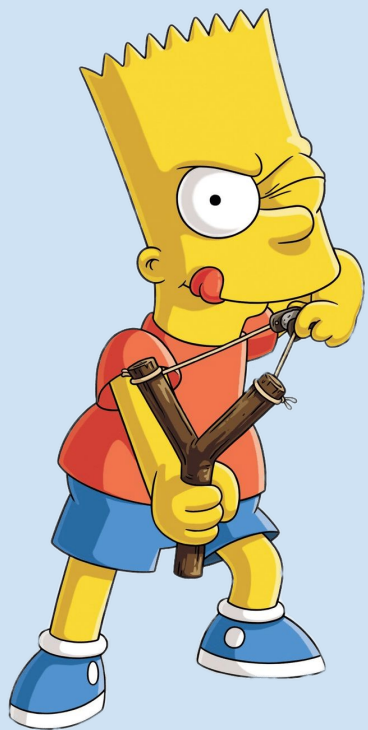
Si tenemos el siguiente vector...

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130



Con lo que sabemos hasta ahora, ¿Cómo buscamos el valor 61?

Recorremos el vector hasta encontrar el valor



0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

FUERZA BRUTA!

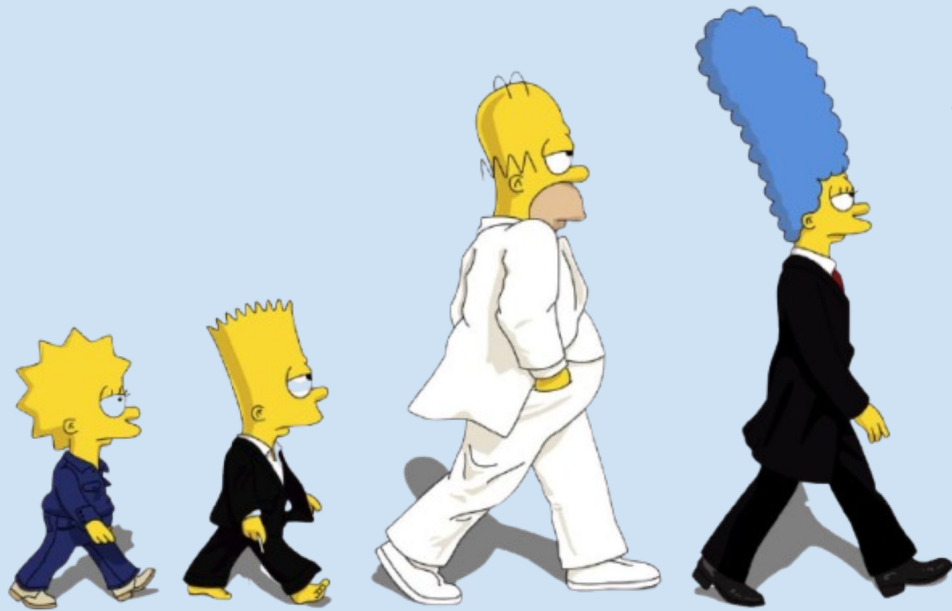
Qué particularidad tiene el vector?

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130



0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

**ESTA ORDENADO
ASCENDENTEMENTE!**



BÚSQUEDA BINARIA

El algoritmo de búsqueda binaria puede ser utilizado ÚNICAMENTE cuando el vector en el cual se realiza la operación posee sus elementos ordenados.

Procedimiento...

	0	1	2	3	4	5	6	7
Paso 1)	1	3	15	43	61	79	105	130
Paso 2)					61	79	105	130
Paso 3)					61	79		
Paso 4)					61			



Pero... como funciona?

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

Inicio= ?

Fin= ?

Centro = ?

¿Qué elementos que ya vimos podemos tener en cuenta para encontrar sus valores?



Tenemos que usar el índice

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

Inicio= 0

Fin= ?

Centro = ?



Tenemos que usar el índice

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

Inicio= 0

Fin= tope -1

Centro = ?



Tenemos que usar el índice

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

Inicio= 0

Fin= tope -1

Centro = (inicio + fin)/2



Siguiendo con el ejemplo

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

Inicio= 0

Fin= 7

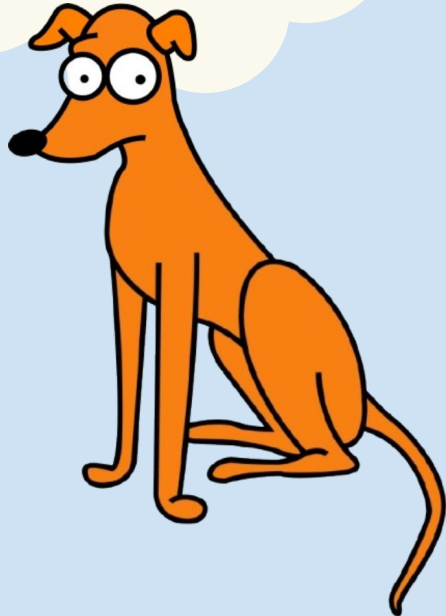
Centro = 3



Pero...

inicio = 0
fin = 7
centro = 3

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130



¿ `vector[centro] == valor buscado` ?

¿ `vector[3] == 61` ?

¿ `43 == 61` ?

NO

¿Entonces?

inicio = 0
fin = 7
centro = 3

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

¿ **vector[centro] < valor buscado ?**

¿ **vector[3] < 61 ?**

¿ **43 < 61?**

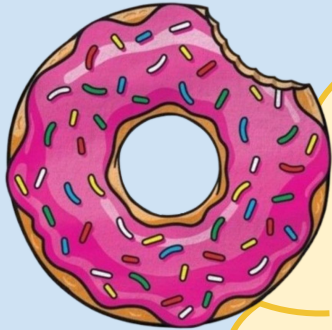
SII



Qué conclusión podemos sacar?

inicio = 0
fin = 7
centro = 3

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

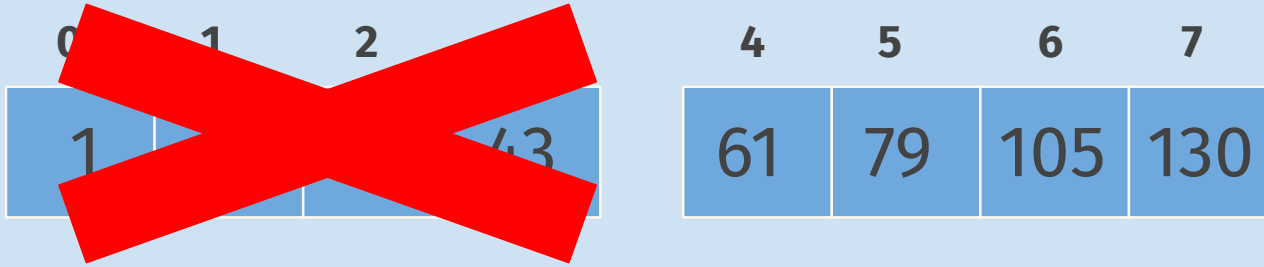


Si estamos buscando el 61 y este es mayor al valor que está en el centro... Sabemos que si dividimos el vector en dos (por el CENTRO), este valor va a estar en la segunda mitad



Qué conclusión podemos sacar?

inicio = 0
fin = 7
centro = 3



Podemos descartar esta primera mitad

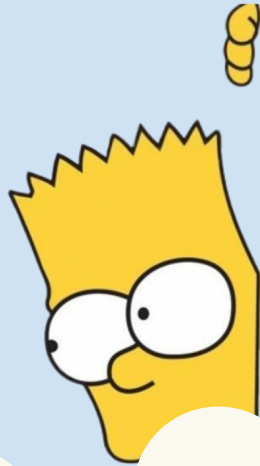


Recalculamos inicio, centro y fin...

inicio = $c_{\text{ant}} + 1 = 4$
fin = 7
centro = 5

4	5	6	7
61	79	105	130

¿ vector[centro] == valor buscado ?
¿ vector[5] == 61 ?
¿ 79 == 61?
NOP





$\text{inicio} = c_{\text{ant}} + 1 = 4$
 $\text{fin} = 7$
 $\text{centro} = 5$

4	5	6	7
61	79	105	130

¿ $\text{vector}[\text{centro}] < \text{valor buscado}$?

¿ $\text{vector}[5] < 61$?

¿ $79 < 61$?

NOP

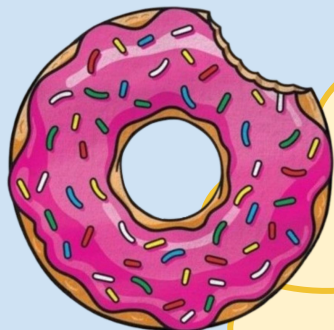


¿Qué hacemos ahora?

inicio = $c_{\text{ant}} + 1 = 4$
fin = 7
centro = 5



**Descarto la
segunda mitad**



**En este caso como 61 es mas chico que 79.
⇒ Sabemos que si dividimos el vector
por el centro este valor va a estar en la
primera mitad**



Ahora recalculando...

inicio = 4
fin = $c_{\text{ant}} - 1 = 4$
centro = 4

4

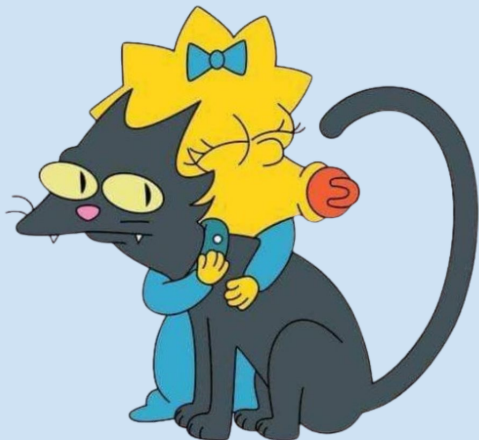
61

¿ vector[centro] == valor buscado ?

¿ vector[4] == 61 ?

¿ 61 == 61?

SI



¿Cuál es el valor de retorno?



4

61

return...?

inicio = 4

fin = $c_{ant} - 1 = 4$

centro = 4



¿Cuál es el valor de retorno?

4

61

return centro



¿Porqué?

Este algoritmo tiene como objetivo BUSCAR un valor en un vector, por lo que yo ya sé cuál es ese valor, lo que necesito encontrar es en qué posición está



¿Qué pasa si no encuentro el valor?

0	1	2	3	4	5	6	7
1	3	15	43	61	79	105	130

Si ahora yo quiero buscar el valor 50, no lo voy a encontrar. ¿Cómo me doy cuenta?

¿Qué pasa si no encuentro el valor?

	0	1	2	3	4	5	6	7	
Paso 1)	1	3	15	43	61	79	105	130	inicio = 0 fin = 7 centro = 3
Paso 2)					61	79	105	130	inicio = 4 fin = 7 centro = 5
Paso 3)					61				inicio = 4 fin = 4 centro = 4
Paso 4)									inicio = 4 fin = $c_{\text{ant}} - 1 = 3$

En resumen...

Calcular/recalcular

CASO:
INICIAL

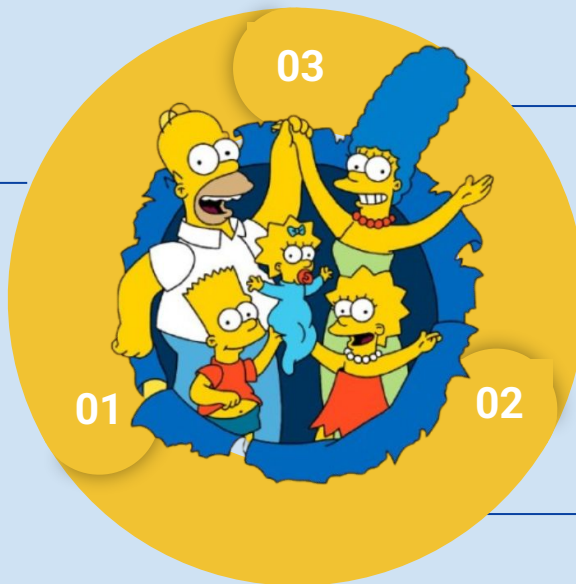
$\text{inicio} = 0$
 $\text{fin} = \text{tope} - 1$
 $\text{centro} = (\text{inicio} + \text{fin})/2$

CASO:
MENOR

$\text{inicio} = \text{inicio}_{\text{ant}}$
 $\text{fin} = \text{c}_{\text{ant}} - 1$
 $\text{centro} = (\text{inicio} + \text{fin})/2$

CASO:
MAYOR

$\text{inicio} = \text{c}_{\text{ant}} + 1$
 $\text{fin} = \text{fin}_{\text{ant}}$
 $\text{centro} = (\text{inicio} + \text{fin})/2$



Descartar

Descarto la parte del vector en donde no se va a encontrar el valor que busco (si es menor descarto la segunda mitad, si es mayor descarto la primera)

Comparar

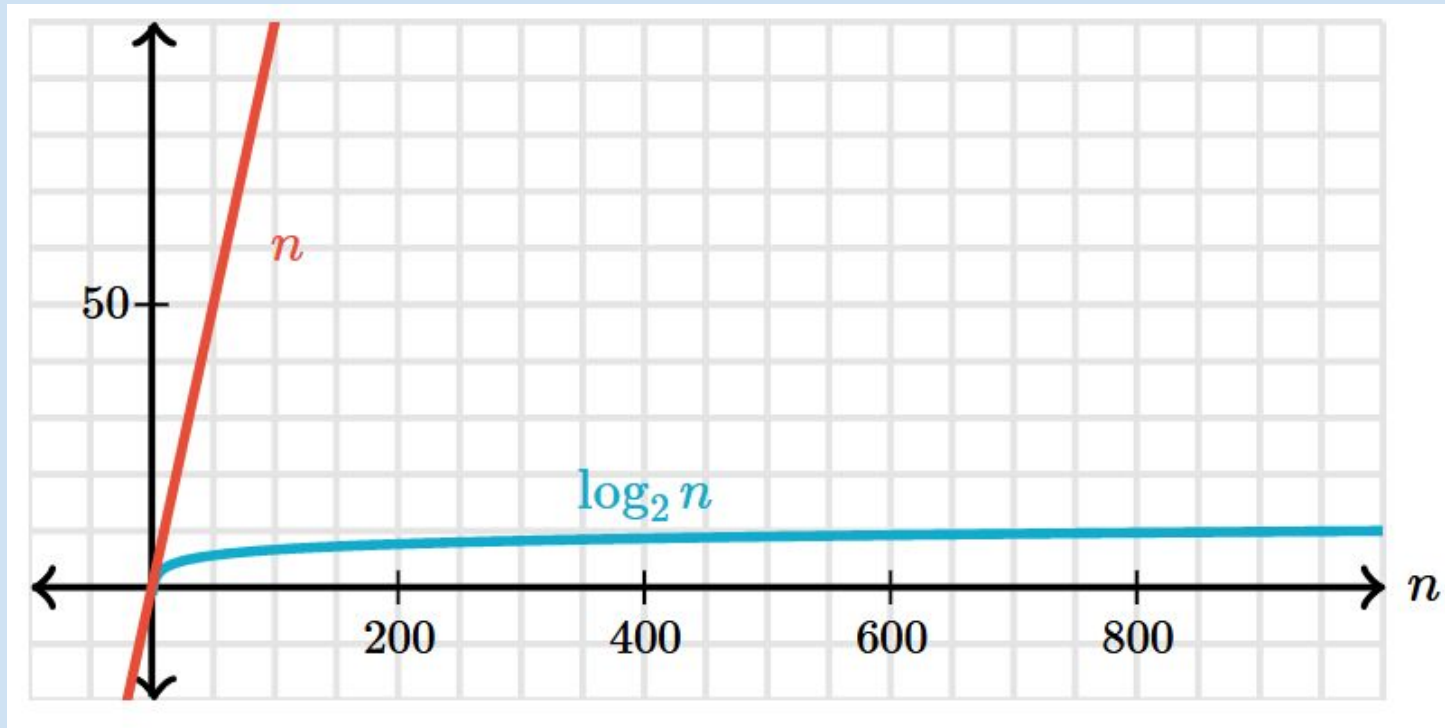
Veo si el valor en el centro es mayor o menor al valor que busco

OBS: el centro siempre se calcula truncado

Rendimiento

Este método es tan bueno que puede reducir la cantidad de comparaciones en una búsqueda significativamente y tanto así que si tenemos un arreglo de n cantidad de elementos su orden es $\log_2(n)$, es decir, que en un vector con 1.000.000 elementos con solo realizar $\log_2(1.000.000) = 20$ (aprox) comparaciones sabremos si el elemento se encuentra en el mismo. se calcula truncado

Rendimiento



Pasando a código: **ITERATIVO**

Pasando a código: ITERATIVO

```
const int NO_ENCONTRADO = -1;

// Precondición: vector está ordenado ascendentemente.
int busqueda_binaria(const int vector[], int tope, int elemento_buscado) {
    int inicio = 0;
    int fin = tope - 1;
    int centro = 0;
    bool encontrado = false;

    while (inicio <= fin && !encontrado) {
        centro = (fin + inicio) / 2;

        if (vector[centro] == elemento_buscado) {
            encontrado = true;
        } else if (vector[centro] < elemento_buscado) {
            inicio = centro + 1;           // voy a la mitad derecha
        } else {
            fin = centro - 1;             // voy a la mitad izquierda
        }
    }

    if (encontrado) {
        return centro;                   // índice donde se encontró
    }
    return NO_ENCONTRADO;
}
```

Preguntas?

