Complejidad

Ejemplo: `https://www.hackerearth.com/practice/algorithms/searching/binary-search/practice-problems/algorithm/discover-the-monk/`

enunciado

Te dan un arreglo A de tamaño n, y q consultas. Para cada consulta, te dan un número entero x y tenes que decir si x está en el arreglo o no.

Input

La primera linea tiene dos enteros, n y q ($1 \leq n,q \leq 10^5$). En la segunda línea hay n números separados por espacios, los elementos ai del arreglo. Luego siguen q líneas con un entero x en cada línea ($1 \leq ai,x \leq 10^9$)

Output

Para cada consulta, imprime en una nueva línea "YES" si pertenece y "NO" en caso contrario.

Primer Intento

```cpp
#include <iostream>
using namespace std;
int a[100000];
int main() {
  ios::sync_with_stdio(false);
  int i,n,q,x;
  cin >> n >> q;
  for(i = 0; i < n; i++)
    cin >> a[i];
  while(q--) {
    cin >> x;
    for(i = 0; i < n; i++) {
      if (a[i] == x) {
        cout << "YES" << endl;
        break;
      }
    }
    if (i == n)
      cout << "NO" << endl;
  }
}
```

Time Limit Exceeded

¿Por qué?

- Una computadora juez puede hacer hasta 10^8 operaciones por segundo.
- Normalmente nos dan 1 segundo para resolver el problema
- Nuestro algoritmo en el peor caso hace 10^{10} operaciones $\rightarrow$ Nuestro algoritmo puede llegar a tardar 100 segundos

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 43,46,51,55 $\rightarrow$

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 43,46,51,55 $\rightarrow$ 51?

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 43,46,51,55 $\rightarrow$ 51?
- 43,46 $\rightarrow$

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 43,46,51,55 $\rightarrow$ 51?
- 43,46 $\rightarrow$ 46?

Búsqueda Binaria: buscando el 50

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55 $\rightarrow$ 12 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 43,46,51,55 $\rightarrow$ 51?
- 43,46 $\rightarrow$ 46?
- [] $\rightarrow$ 5 consultas

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 70,100,1000 $\rightarrow$

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 70,100,1000 $\rightarrow$ 100?

Búsqueda Binaria: buscando el 1000

Arreglo

[1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000] (16 elementos)

Búsqueda lineal

1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 16 consultas.

Búsqueda Binaria

- 1,3,6,8,10,15,24,30,42,43,46,51,55,70,100,1000 $\rightarrow$ 42?
- 43,46,51,55,70,100,1000 $\rightarrow$ 55?
- 70,100,1000 $\rightarrow$ 100?
- 1000 $\rightarrow$ 4 consultas

Ganancia

En particular, en búsqueda binaria siempre divido en 2 el intervalo, entonces en el peor caso hago log2(n)+1 consultas.
En una lista de tamaño 16 (2^4) hago 5 consultas con búsqueda binaria y 16 con búsqueda lineal
En una lista de tamaño 1.073.741.824(2^{30}) son 31 consultas contra más de un millón.
Cuanto más grande el n más grande la ganancia

Volviendo al ejercicio

https://www.hackerearth.com/practice/algorithms/searching/binary-search/practice-problems/algorithm/discover-the-monk/

enunciado

Te dan un arreglo A de tamaño n, y q consultas. Para cada consulta, te dan un número entero x y tenes que decir si x está en el arreglo o no.

Input

La primera linea tiene dos enteros, n y q ($1 \leq n,q \leq 10^5$). En la segunda línea hay n números separados por espacios, los ai elementos del arreglo. Luego siguen q líneas con un entero x en cada línea ($1 \leq ai,x \leq 10^9$)

Output

Para cada consulta, imprime en una nueva línea "YES" si pertenece y "NO" en caso contrario.

Errores frecuentes

Error número 1:

Búsqueda binaria funciona en una lista ordenada
Si van a ordenar, usen el sort de c++.
Complejidad de sort: n*log2(n)

Error número 2:

Se me clava el programa!
Fijensen bien los índices, si se equivocan puede terminar en un loop infinito
Usen cout para ver en qué parte se clava

Solución (parte 1)

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int a[100000];
int main() {
  ios::sync_with_stdio(false);
  int i,n,q,x,l,r,m;
  cin >> n >> q;
  for(i = 0; i < n; i++)
    cin >> a[i];
  sort(a,a+n);
```

Solución (parte 2)

```
    while(q--) {
      cin >> x;
      l = 0, r = n-1;
      while(l <= r) {
        m = (l + r)/2;
        if (a[m] == x) {
          cout << "YES" << endl;
          break;
        } else if (a[m] < x){
          l = m+1;
        } else {
          r = m-1;
        }
      }
      if (l > r)
        cout << "NO" << endl;
    }
}
```

Orden

Consigna

Dado un número n par, devolver la mitad de n.

```
    int n;
2   cin >> n;
    cout << n/2;
4
```

Como escala?

¿Qué pasa si n es 10?
¿Si n es 10^6?
¿Si n es 10^{18}?

Orden

Consigna

Decir cual es el número más grande en un arreglo de números positivos
Sea el arreglo a[n]

```
1   int max = -INFINITO;
    for(int i = 0; i < n; i++)
3     if (a[i] > max)
        max = a[i];
5   cout << max << endl;
```

Como escala?

¿Qué pasa si n es 10?
¿Si n es 10^6?
¿Si n es 10^{18}?

¿Cómo calcular el orden?

```
1   int n, tmp, res = -1;
    cin  >> n;
3   for(i = 0; i < n; i++) {
      cin >> tmp;
5     res = max(res,tmp);
      cout << max << endl;
7   }
```

¿Cómo calcular el orden?

```
1   int n, tmp, res = -1;
    cin  >> n;
3   for(i = 0; i < n; i++) {
      cin >> tmp;
5     res = max(res,tmp);
      cout << max << endl;
7   }
```

Operaciones

3*n + 4 operaciones

O(n)

¿Cómo calcular el orden?

```
int i, j, n;
cin >> n;
int a[n][n];
for(i = 0; i < n; i++) {
    for(j = 0; j < n; j++) {
        cin >> a[i][j];
        cout << "exito" << endl;
  }
}
for(i = 0; i < n; i++)
    cout << a[i][0] << endl;
```

¿Cómo calcular el orden?

```
1  int i, j, n;
   cin >> n;
3  int a[n][n];
   for(i = 0; i < n; i++) {
5      for(j = 0; j < n; j++) {
           cin >> a[i][j];
7          cout << "exito" << endl;
       }
9  }
   for(i = 0; i < n; i++)
11     cout << a[i][0] << endl;
```

Operaciones

$5 + 2 * n^2 +$ n operaciones

$O(n^2)$

Maximum subarray sum

Problema

Dado un arreglo de enteros de tamaño n, encuentre la mayor suma que se obtiene por sumar posiciones adyacentes de arreglo

Maximum subarray sum

Problema

Dado un arreglo de enteros de tamaño n, encuentre la mayor suma que se obtiene por sumar posiciones adyacentes de arreglo

Ejemplo

a[8] = {-1, 2, 4, -3, 5 ,2, -5, 2};
Respuesta $\rightarrow$ [2, 4, -3, 5, 2] = 10

Solución 1

Idea

Para cada posible subarreglo, calculo la suma y me quedo con la mayor.

```
1  int mejor = 0;
   for (int i = 0; i < n; i++) {
3    for (int j = i; j < n; j++) {
       int sum = 0;
5      for (int k = i; k <= j; k++) {
         sum += array[k];
7      }
       mejor = max(mejor,sum);
9    }
   }
11 cout << mejor << endl;
```

a[5] = {-1, 2, 4, -3, 5}
i k j

Solución 1

{-1, 2, 4, -3, 5}

{-1}
{-1, 2}
{-1, 2, 4}
{-1, 2, 4, -3}
{-1, 2, 4, -3, 5}
{2}
{2, 4}
{2, 4, -3}
{2, 4, -3, 5}
{4}
{4, -3}
{4, -3, 5} ...

Total de subarreglos o rangos

$5 + 4 + 3 + 2 + 1$

Solución 1

Total de rangos

Para un arreglo de tamaño n hay $n * (n+1)/2$ rangos $= n^2/2 + n/2$ $= O(n^2)$

En cada rango

En cada rango recorro a lo sumo n elementos, entonces recorrer un rango es O(n)

En total

hay $O(n^2)$ rangos y en cada rango hago O(n) operaciones, entonces el costo total es $O(n^3)$

Solución 2

Idea

Calculo la suma del rango mientras voy armando el rango. Con esto se ahorra recorrer el rango luego de haberlo elegido y por o tanto se hacen O(n) operaciones menos en cada rango

```
int mejor = 0;
for (int i = 0; i < n; i++) {
 int sum = 0;
 for (int j = i; j < n; j++) {
  sum += a[j];
  mejor = max(mejor,sum);
 }
}
cout << mejor << endl;
```

Solución 2

Idea

Calculo la suma del rango mientras voy armando el rango. Con esto se ahorra recorrer el rango luego de haberlo elegido y por o tanto se hacen O(n) operaciones menos en cada rango

```
int mejor = 0;
for (int i = 0; i < n; i++) {
 int sum = 0;
 for (int j = i; j < n; j++) {
  sum += a[j];
  mejor = max(mejor,sum);
 }
}
cout << mejor << endl;
```

```
int mejor = 0;
for (int i = 0; i < n; i++) {
 for (int j = i; j < n; j++) {
  int sum = 0;
  for (int k = i; k <= j; k++)
     {
   sum += array[k];
  }
  mejor = max(mejor,sum);
 }
}
cout << mejor << endl;
```

Solución 2

{-1, 2, 4, -3, 5}

$\{-1\} \rightarrow -1$
$\{-1, 2\} \rightarrow 1$
$\{-1, 2, 4\} \rightarrow 5$
$\{-1, 2, 4, -3\} \rightarrow 2$
$\{-1, 2, 4, -3, 5\} \rightarrow 7$
$\{2\} \rightarrow 2$
$\{2, 4\} \rightarrow 6$
$\{2, 4, -3\} \rightarrow 3$
$\{2, 4, -3, 5\} \rightarrow 8$
$\{4\} \rightarrow 4$
$\{4, -3\} \rightarrow 1$
$\{4, -3, 5\} \rightarrow 6$ ...

Total de rangos:

$5 + 4 + 3 + 2 + 1$

Solución 2

Total de rangos

Siguen habiendo $O(n^2)$ rangos

En cada rango

Termino de encontrar el rango, y ya puedo obtener la respuesta en orden $O(1)$

En total

hay $O(n^2)$ rangos y en cada rango hago $O(1)$ operaciones, entonces el costo total es $O(n^2)$

Solución 3

Idea

O me quedo con todo lo que tenía antes, o lo descarto y empiezo de 0.

```
1 int mejor = 0, sum = 0;
  for (int i = 0; i < n; i++) {
3   sum = max(a[i],sum+a[i]);
    mejor = max(mejor,sum);
5 }
  cout << mejor << endl;
7
```

Solución 3

Idea

O me quedo con todo lo que tenía antes, o lo descarto y empiezo de 0.

```
int mejor = 0, sum = 0;
for (int i = 0; i < n; i++) {
  sum = max(a[i],sum+a[i]);
  mejor = max(mejor,sum);
}
cout << mejor << endl;
```

{2, 4, -7, 5, 2}

{2} → 2
{-2, 4} → 6
{-2, 4, -7} → -1
{2, 4, -7, 5} → 5
{2, 4, -7, 5 2} → 7

Solución 3

Idea

O me quedo con todo lo que tenía antes, o lo descarto y empiezo de 0.

```
int mejor = 0, sum = 0;
for (int i = 0; i < n; i++) {
  sum = max(a[i],sum+a[i]);
  mejor = max(mejor,sum);
}
cout << mejor << endl;
```

```
int mejor = 0;
for (int i = 0; i < n; i++) {
 int sum = 0;
 for (int j = i; j < n; j++) {
  sum += a[j];
  mejor = max(mejor,sum);
 }
}
cout << mejor << endl;
```

Solución 3

Idea

O me quedo con todo lo que tenía antes, o lo descarto y empiezo de 0.

```
int mejor = 0, sum = 0;
for (int i = 0; i < n; i++) {
  sum = max(a[i],sum+a[i]);
  mejor = max(mejor,sum);
}
cout << mejor << endl;
```

```
int mejor = 0;
for (int i = 0; i < n; i++) {
 int sum = 0;
 for (int j = i; j < n; j++) {
  sum += a[j];
  mejor = max(mejor,sum);
 }
}
cout << mejor << endl;
```

Cantidad de operaciones

Recorro una sola vez el arreglo, y siempre hago 2 operaciones

Operaciones = 2*n = O(n)

Input

Prestar atención

Los datos del input te limitan de qué orden puede ser tu algoritmo.
Muchas veces conviene leer el tamaño del input antes de empezar a pensar en una solución

Tamaño de Input	Complejidad máxima
$n \leq 10$	$O(n!)$
$n \leq 20$	$O(2^n)$
$n \leq 500$	$O(n^3)$
$n \leq 5000$	$O(n^2)$
$n \leq 10^6$	$O(n\log(n))$ o $O(n)$
$n > 10^6$	$O(1)$ o $O(\log(n))$

Si no les queda muy claro qué complejidad tiene algún algoritmo o programa y por qué, no duden en consultarnos

Límites (Valores aproximados)

Enteros

- int: $-2 * 10^9$ hasta $2 * 10^9$
- long long: $-9 * 10^{18}$ hasta $9 * 10^{18}$

Decimales

Para decimales les conviene usar double

Prestar atención

Muchas veces el tamaño de un n es menor que 10^9 pero se suman muchas veces y podes obtener números hasta 10^{18}

Límites de memoria

Suelen ser cerca de 1gb ($1.5 * 10^8 bytes$)

Regla general

Que el orden de espacio en memoria no supere el orden en límite de tiempo

Errores frecuentes

Menos evitables:

- Entender mal el enunciado
- Creer que tu solución es correcta, pero en algún caso falla.
  ¿Será un bug o será una solución incorrecta?

Totalmente evitables:

- No revisar o revisar mal el orden y tener un TLE
- Demorarse haciendo un algoritmo eficiente complicado cuando uno simple y mayor orden entra en el límite de tiempo y se escribe mucho más rápido
- No revisar los límites, y usar int cuando en algunos casos supera el límite de los int (Integer Overflow)
- Bugs de "Wrong by 1":
  Puse $\leq$ cuando era $<$
  Me olvide del caso n $= 0$

Ejercicios con búsqueda binaria

http://codeforces.com/problemset/problem/600/B
http://www.spoj.com/problems/AGGRCOW/
https://www.hackerearth.com/practice/algorithms/searching/binary-search/practice-problems/algorithm/bishu-and-soldiers/
http://codeforces.com/problemset/problem/348/A
http://codeforces.com/problemset/problem/743/B