

# Busqueda Binaria y Ventana Deslizante

Aprende binary search pibe

Matias Ramos

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

Training Camp 2024



# Gracias Sponsors!

Organizador



Universidad  
Nacional  
de Rosario

Facultad de  
Ciencias Exactas,  
Ingeniería y Agrimensura



Diamond



GTS

Gold



- 1 Búsqueda Lineal
- 2 Búsqueda Binaria
- 3 Frase motivadora
- 4 Búsqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM

- 1 Búsqueda Lineal
- 2 Búsqueda Binaria
- 3 Frase motivadora
- 4 Búsqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM

## Problema

Tengo un arreglo *ordenado* de  $N$  elementos. Quiero encontrar el elemento  $X$  dentro de ese arreglo.

# Busqueda Lineal

```
for (int i = 0; i < n; i++) {  
    if (array[i] == X) {  
        // encuentre X  
    }  
}
```

Complejidad

# Busqueda Lineal

```
for (int i = 0; i < n; i++) {  
    if (array[i] == X) {  
        // encuentre X  
    }  
}
```

Complejidad  
 $O(n)$

- 1 Busqueda Lineal
- 2 Busqueda Binaria
- 3 Frase motivadora
- 4 Busqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM



## Problema

Tengo un arreglo ***ordenado*** de  $N$  elementos. Quiero encontrar el elemento  $X$  dentro de ese arreglo.

Recordemos búsqueda binaria

# Veamos un ejemplo

0	1	2	3	4	5	6	7	8
1	1	4	5	6	10	11	11	23

# Implementación Búsqueda Binaria

```
int a = 0, b = n-1;
while (a <= b) {
    int k = (a+b)/2;
    if (array[k] == x) {
        // encuentre X en indice K
    }
    if (array[k] > x) b = k-1;
    else a = k+1;
}
```

Complejidad

# Implementación Búsqueda Binaria

```
int a = 0, b = n-1;
while (a <= b) {
    int k = (a+b)/2;
    if (array[k] == x) {
        // encuentre X en indice K
    }
    if (array[k] > x) b = k-1;
    else a = k+1;
}
```

Complejidad  
 $O(\log n)$

# Implementacion Alternativa/Iterativa

- Iterar eficientemente el arreglo de izquierda a derecha.
- Hacer saltos y disminuir la longitud de los mismos cuando nos acercamos al elemento buscado.
- La longitud inicial es  $n/2$ . En cada salto la longitud del salto se divide a la mitad.  $n/4, n/8, n/16 \dots$
- Hasta que la longitud sea 1.
- Luego de este proceso el elemento es encontrado, o sabemos que no aparece en el arreglo.

# Implementacion Alternativa/Iterativa

```
int k = 0;
for (int b = n/2; b >= 1; b /= 2) {
    while (k+b < n && array[k+b] <= x) k += b;
}
if (array[k] == x) {
    // encuentre X en indice K
}
```

Complejidad  $O(\log n)$

- 1 Busqueda Lineal
- 2 Busqueda Binaria
- 3 Frase motivadora
- 4 Busqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM

...go and solve some problems, learn how to use binary search.

Um\_nik, World Champion 2020.



- 1 Búsqueda Lineal
- 2 Búsqueda Binaria
- 3 Frase motivadora
- 4 Búsqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM

# Problema disparador

Supongamos que queremos encontrar el valor más chico de  $K$  que es una solución válida para un problema.

Tenemos una función  $ok(x)$  que retorna *true* si  $x$  es una solución válida y *false* en caso contrario.

Además sabemos que  $ok(x)$  es *false* cuando  $x < k$  y *true* cuando  $x \geq k$

$x$	0	1	...	$k$	$k+1$	...
$ok(x)$	False	False	False	True	True	True

# Solución al problema disparador

```
int x = -1;
for (int b = z; b >= 1; b /= 2) {
    while (!ok(x+b)) x += b;
}
int k = x+1;
```

Complejidad

# Solución al problema disparador

```
int x = -1;
for (int b = z; b >= 1; b /= 2) {
    while (!ok(x+b)) x += b;
}
int k = x+1;
```

Complejidad  
 $O(O(ok(x))\log(n))$

# Un problema

## Evolucionando Pokémons

Ash acaba de atrapar  $N$  pokémons. Él tiene a su disposición  $M$  barras de caramelo. Evolucionar a un pokémon, le cuesta  $X$  barras de caramelo. A su vez, puede vender a un pokémon y recibir a cambio  $Y$  barras de caramelo. ¿Cuál es la máxima cantidad de pokémons que puede evolucionar?



## Observación clave

Si fijamos la cantidad de pokémones que vamos a evolucionar, digamos  $k$  con  $0 \leq k \leq N$ . Entonces podemos vender a los  $N - k$  pokémones que no vamos a evolucionar. Con esto, disponemos de  $M + (N - k) \cdot Y$  barras de caramelo.

Para evolucionar a los  $k$  pokémones, necesitamos tener al menos  $k \cdot X$  barras de caramelos. O sea, debe valer que:

$$k \cdot X \leq (N - k) \cdot Y + M$$

## Observación clave

Si fijamos la cantidad de pokémones que vamos a evolucionar, digamos  $k$  con  $0 \leq k \leq N$ . Entonces podemos vender a los  $N - k$  pokémones que no vamos a evolucionar. Con esto, disponemos de  $M + (N - k) \cdot Y$  barras de caramelo.

Para evolucionar a los  $k$  pokémones, necesitamos tener al menos  $k \cdot X$  barras de caramelos. O sea, debe valer que:

$$k \cdot X \leq (N - k) \cdot Y + M$$

- 1 Solución: Probar todos los valores de  $k$  entre 0 y  $N$  (inclusive), y tomar el de mayor valor que satisfaga la condición. Complejidad :  $\mathcal{O}(N)$





- 2 Solución: Notemos que si podemos evolucionar  $k$  pokemones, también podemos evolucionar  $k - 1$ ,  $k - 2$ ,  $\dots$ ,  $0$  pokemones. En particular siempre podemos evolucionar  $0$  pokemones. De la misma forma, si no podemos evolucionar  $k$  pokemones, menos aún podremos evolucionar  $k + 1$ , o  $k + 2$  pokemones (o cualquier otro número mayor). En particular nunca podremos evolucionar  $N + 1$  pokemones.

- ② Solución: Notemos que si podemos evolucionar  $k$  pokemones, también podemos evolucionar  $k - 1, k - 2, \dots, 0$  pokemones. En particular siempre podemos evolucionar 0 pokemones.
- De la misma forma, si no podemos evolucionar  $k$  pokemones, menos aún podremos evolucionar  $k + 1$ , o  $k + 2$  pokemones (o cualquier otro número mayor). En particular nunca podremos evolucionar  $N + 1$  pokemones.
- Entonces, tomando como propiedad:
- $P(k)$  : “Puedo evolucionar  $k$  pokemones”, podemos hacer búsqueda binaria en la propiedad  $P$ .

- 2 Solución: Notemos que si podemos evolucionar  $k$  pokemones, también podemos evolucionar  $k - 1, k - 2, \dots, 0$  pokemones. En particular siempre podemos evolucionar 0 pokemones. De la misma forma, si no podemos evolucionar  $k$  pokemones, menos aún podremos evolucionar  $k + 1$ , o  $k + 2$  pokemones (o cualquier otro número mayor). En particular nunca podremos evolucionar  $N + 1$  pokemones.

Entonces, tomando como propiedad:

$P(k)$  : “Puedo evolucionar  $k$  pokemones”, podemos hacer búsqueda binaria en la propiedad  $P$ .

Complejidad:  $\mathcal{O}(\lg(N))$



- ③ Sabemos que  $0 \leq k \leq N$ . Y de la condición vista, podemos despejar una inecuación para  $k$ :

- 3 Sabemos que  $0 \leq k \leq N$ . Y de la condición vista, podemos despejar una inecuación para  $k$ :

$$k \cdot X \leq (N - k) \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X \leq N \cdot Y - k \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X + k \cdot Y \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot (X + Y) \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \leq \frac{N \cdot Y + M}{X + Y}$$

- 3 Sabemos que  $0 \leq k \leq N$ . Y de la condición vista, podemos despejar una inecuación para  $k$ :

$$k \cdot X \leq (N - k) \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X \leq N \cdot Y - k \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X + k \cdot Y \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot (X + Y) \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \leq \frac{N \cdot Y + M}{X + Y}$$

Por lo tanto, la respuesta será :

$$\min \left( N, \left\lfloor \frac{N \cdot Y + M}{X + Y} \right\rfloor \right)$$



- 3 Sabemos que  $0 \leq k \leq N$ . Y de la condición vista, podemos despejar una inecuación para  $k$ :

$$k \cdot X \leq (N - k) \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X \leq N \cdot Y - k \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X + k \cdot Y \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot (X + Y) \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \leq \frac{N \cdot Y + M}{X + Y}$$

Por lo tanto, la respuesta será :

$$\min \left( N, \left\lfloor \frac{N \cdot Y + M}{X + Y} \right\rfloor \right)$$

Complejidad:  $\mathcal{O}(1)$

- 3 Sabemos que  $0 \leq k \leq N$ . Y de la condición vista, podemos despejar una inecuación para  $k$ :

$$k \cdot X \leq (N - k) \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X \leq N \cdot Y - k \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot X + k \cdot Y \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \cdot (X + Y) \leq N \cdot Y + M \quad \Longleftrightarrow$$

$$k \leq \frac{N \cdot Y + M}{X + Y}$$

Por lo tanto, la respuesta será :

$$\min \left( N, \left\lfloor \frac{N \cdot Y + M}{X + Y} \right\rfloor \right)$$

Complejidad:  $\mathcal{O}(1)$

<https://csacademy.com/contest/archive/task/pokemon-evolution>

- 1 Búsqueda Lineal
- 2 Búsqueda Binaria
- 3 Frase motivadora
- 4 Búsqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM

## Problema:

Dado un arreglo de  $n$  números positivos, y un número  $x$ , nos interesa saber si existe un subarreglo cuya suma sea  $x$ .

## Problema:

Dado un arreglo de  $n$  números positivos, y un número  $x$ , nos interesa saber si existe un subarreglo cuya suma sea  $x$ .

## Ejemplos:

- $A = \overset{0}{2}, \overset{1}{3}, \overset{2}{2}, \overset{3}{5}, \overset{4}{1}, \overset{5}{5}, \overset{6}{2}, \overset{7}{3}$ . Buscamos sumar  $x = 8$   
La respuesta es :

## Problema:

Dado un arreglo de  $n$  números positivos, y un número  $x$ , nos interesa saber si existe un subarreglo cuya suma sea  $x$ .

## Ejemplos:

- $A = \overset{0}{[2, \overset{1}{3, \overset{2}{2, \overset{3}{5, \overset{4}{1, \overset{5}{5, \overset{6}{2, \overset{7}{3}}}}}}]}$ . Buscamos sumar  $x = 8$   
La respuesta es : “El subarreglo [2..4] suma 8”

# Ventanas Deslizantes

## Problema:

Dado un arreglo de  $n$  números positivos, y un número  $x$ , nos interesa saber si existe un subarreglo cuya suma sea  $x$ .

## Ejemplos:

- $A = [2, 3, \underbrace{2, 5, 1}_{2+5+1=8}, 5, 2, 3]$ . Buscamos sumar  $x = 8$

La respuesta es : “El subarreglo [2..4] suma 8”

- $A = [2, 3, 2, 5, 1, 5, 2, 3]$ . Buscamos sumar  $x = 4$
- La respuesta es :

# Ventanas Deslizantes

## Problema:

Dado un arreglo de  $n$  números positivos, y un número  $x$ , nos interesa saber si existe un subarreglo cuya suma sea  $x$ .

## Ejemplos:

- $A = [2, 3, \underbrace{2, 5, 1}_{2+5+1=8}, 5, 2, 3]$ . Buscamos sumar  $x = 8$

La respuesta es : “El subarreglo [2..4] suma 8”

- $A = [2, 3, 2, 5, 1, 5, 2, 3]$ . Buscamos sumar  $x = 4$
- La respuesta es : “No hay subarreglo de  $A$  que sume 4”



# Ventanas Deslizantes

## Problema:

Dado un arreglo de  $n$  números positivos, y un número  $x$ , nos interesa saber si existe un subarreglo cuya suma sea  $x$ .

## Ejemplos:

- $A = [2, 3, \underbrace{2, 5, 1}_{2+5+1=8}, 5, 2, 3]$ . Buscamos sumar  $x = 8$

La respuesta es : “El subarreglo [2..4] suma 8”

- $A = [2, 3, 2, 5, 1, 5, 2, 3]$ . Buscamos sumar  $x = 4$
- La respuesta es : “No hay subarreglo de  $A$  que sume 4”

- 1 Probar todos los subarreglos posibles. Es decir, todos los subarreglos  $[i..j]$  con  $0 \leq j < n$  e  $0 \leq i \leq j$ . Para cada uno de ellos calcular  $\text{suma}(i,j)$ , la suma de los números en el subarreglo especificado y ver si es exactamente  $x$ .

Complejidad :

- 1 Probar todos los subarreglos posibles. Es decir, todos los subarreglos  $[i..j]$  con  $0 \leq j < n$  e  $0 \leq i \leq j$ . Para cada uno de ellos calcular  $\text{suma}(i,j)$ , la suma de los números en el subarreglo especificado y ver si es exactamente  $x$ .

Complejidad :  $\mathcal{O}(n^3)$  u  $\mathcal{O}(n^2)$  dependiendo de la implementación de la función suma.

- 1 Probar todos los subarreglos posibles. Es decir, todos los subarreglos  $[i..j]$  con  $0 \leq j < n$  e  $0 \leq i \leq j$ . Para cada uno de ellos calcular  $\text{suma}(i,j)$ , la suma de los números en el subarreglo especificado y ver si es exactamente  $x$ .  
Complejidad :  $\mathcal{O}(n^3)$  u  $\mathcal{O}(n^2)$  dependiendo de la implementación de la función  $\text{suma}$ .
- 2 Utilizar una ventana deslizante para resolver el problema. La idea es mantener dos índices que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos “ $i$ ” y al derecho lo llamaremos “ $j$ ”. Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a mantener cuánto vale la suma de los números dentro de la ventana  $[i..j]$  (notar que no incluimos el extremo derecho).

- ② Utilizar una ventana deslizante para resolver el problema. La idea es mantener dos índices que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos “ $i$ ” y al derecho lo llamaremos “ $j$ ”. Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a mantener cuánto vale la suma de los números dentro de la ventana  $[i..j)$  (notar que no incluimos el extremo derecho).

- 2 Utilizar una ventana deslizante para resolver el problema. La idea es mantener dos índices que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos “ $i$ ” y al derecho lo llamaremos “ $j$ ”. Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a mantener cuánto vale la suma de los números dentro de la ventana  $[i..j)$  (notar que no incluimos el extremo derecho).

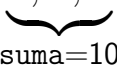
Ejemplo: Si  $i = 1$  y  $j = 4$ , mantenemos la suma en el subarreglo  $[i..j) = [1,4) = [1..3]$ .

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$

- 2 Utilizar una ventana deslizante para resolver el problema. La idea es mantener dos índices que son los extremos de nuestra ventana deslizante, al extremo izquierdo lo llamaremos “ $i$ ” y al derecho lo llamaremos “ $j$ ”. Ambos extremos comienzan en el principio del arreglo, y en todo momento vamos a mantener cuánto vale la suma de los números dentro de la ventana  $[i..j)$  (notar que no incluimos el extremo derecho).

Ejemplo: Si  $i = 1$  y  $j = 4$ , mantenemos la suma en el subarreglo  $[i..j) = [1,4) = [1..3]$ .

$$A = \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ & i & & & & j & & & \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$

  
suma=10

Buscamos sumar  $x = 8$

$$A = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{array}$$



# Paso 1

Buscamos sumar  $x = 8$

$$\text{suma} = 0 < x$$

$$A = \begin{array}{cccccccc} & ij & & & & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{array}$$

# Paso 1

Buscamos sumar  $x = 8$

$\text{suma} = 0 < x \rightarrow$  Aumento  $j$

$$A = \begin{array}{cccccccc} & \overset{ij}{0} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{array}$$

## Paso 2

Buscamos sumar  $x = 8$

$$\text{suma} = 2 < x$$

$$A = \left[ \overset{i}{\underset{\text{2}}{2}}, \overset{j}{3}, 2, 5, 1, 5, 2, 3 \right]$$

## Paso 2

Buscamos sumar  $x = 8$

$\text{suma} = 2 < x \rightarrow$  Aumento  $j$

$$A = \left[ \overset{i}{\underbrace{2}}, \overset{j}{3}, 2, 5, 1, 5, 2, 3 \right]$$

## Paso 3

Buscamos sumar  $x = 8$

$$\text{suma} = 5 < x$$

$$A = \begin{matrix} & \overset{i}{0} & 1 & \overset{j}{2} & 3 & 4 & 5 & 6 & 7 \\ & \underbrace{2, 3} & , & 2, & 5, & 1, & 5, & 2, & 3 \end{matrix}$$

## Paso 3

Buscamos sumar  $x = 8$

$\text{suma} = 5 < x \rightarrow$  Aumento  $j$

$$A = \begin{matrix} & \overset{i}{0} & 1 & \overset{j}{2} & 3 & 4 & 5 & 6 & 7 \\ & \underbrace{2, 3} & , & 2, & 5, & 1, & 5, & 2, & 3 \end{matrix}$$

## Paso 4

Buscamos sumar  $x = 8$

$$\text{suma} = 7 < x$$

$$A = \begin{matrix} & \overset{i}{0} & 1 & 2 & \overset{j}{3} & 4 & 5 & 6 & 7 \\ & \underbrace{2, 3, 2} & 5, & 1, & 5, & 2, & 3 \end{matrix}$$

## Paso 4

Buscamos sumar  $x = 8$

$\text{suma} = 7 < x \rightarrow$  Aumento  $j$

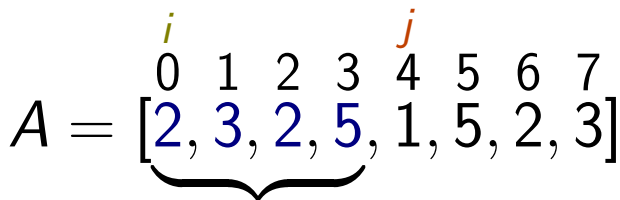
$$A = \begin{matrix} & \overset{i}{0} & 1 & 2 & \overset{j}{3} & 4 & 5 & 6 & 7 \\ & \underbrace{2, 3, 2} & 5 & 1 & 5 & 2 & 3 \end{matrix}$$



## Paso 5

Buscamos sumar  $x = 8$

$$\text{suma} = 12 > x$$

$$A = \begin{matrix} & i & & & j & & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ & 2 & 3 & 2 & 5 & 1 & 5 & 2 & 3 \end{matrix}$$


## Paso 5

Buscamos sumar  $x = 8$

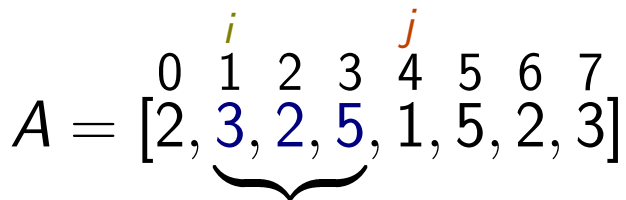
$\text{suma} = 12 > x \rightarrow$  Aumento  $i$

$$A = \begin{matrix} & \overset{i}{0} & 1 & 2 & 3 & \overset{j}{4} & 5 & 6 & 7 \\ & \underbrace{[2, 3, 2, 5]}_{\text{suma}} & , & 1 & , & 5 & , & 2 & , & 3 \end{matrix}$$

## Paso 6

Buscamos sumar  $x = 8$

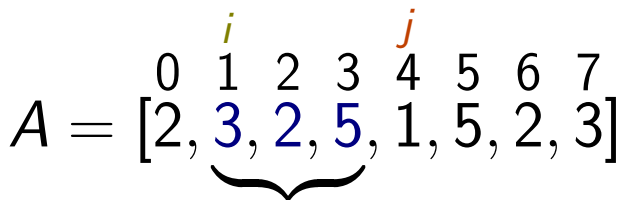
$$\text{suma} = 10 > x$$

$$A = \begin{matrix} & & i & & & & j & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$


## Paso 6

Buscamos sumar  $x = 8$


$\text{suma} = 10 > x \rightarrow$  Aumento  $i$

$$A = \begin{matrix} & i & & & j & & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$


## Paso 7

Buscamos sumar  $x = 8$

$$\text{suma} = 7 < x$$

$$A = [2, 3, \overset{i}{2}, \overset{j}{5}, 1, 5, 2, 3]$$


## Paso 7

Buscamos sumar  $x = 8$


$\text{suma} = 7 < x \rightarrow$  Aumento  $j$

$$A = [2, 3, \overset{i}{\underset{\underbrace{\hspace{1.5cm}}}{2}}, \overset{j}{5}, 1, 5, 2, 3]$$

## Paso 8

Buscamos sumar  $x = 8$


$$\text{suma} = 8 = x$$

$$A = \begin{matrix} & 0 & 1 & \overset{i}{2} & 3 & 4 & \overset{j}{5} & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$


## Paso 8

Buscamos sumar  $x = 8$

$\text{suma} = 8 = x \rightarrow$  Podemos terminar

$$A = \begin{matrix} & 0 & 1 & \overset{i}{2} & 3 & 4 & \overset{j}{5} & 6 & 7 \\ [2, & 3, & 2, & 5, & 1, & 5, & 2, & 3] \end{matrix}$$




- ¿Cuántos pasos hace el algoritmo?  
Respuesta :

- ¿Cuántos pasos hace el algoritmo?

Respuesta : En cada paso, o bien aumenta  $i$  o aumenta  $j$ . Cada uno de ellos puede ser aumentado a lo sumo  $n$  veces. Por lo tanto al cabo de  $2n$  pasos en el peor caso, finaliza nuestro algoritmo.

- ¿Cuántos pasos hace el algoritmo?

Respuesta : En cada paso, o bien aumenta  $i$  o aumenta  $j$ . Cada uno de ellos puede ser aumentado a lo sumo  $n$  veces. Por lo tanto al cabo de  $2n$  pasos en el peor caso, finaliza nuestro algoritmo.

- ¿Por qué es importante que los números sean positivos?

Respuesta :

- ¿Cuántos pasos hace el algoritmo?

Respuesta : En cada paso, o bien aumenta  $i$  o aumenta  $j$ . Cada uno de ellos puede ser aumentado a lo sumo  $n$  veces. Por lo tanto al cabo de  $2n$  pasos en el peor caso, finaliza nuestro algoritmo.

- ¿Por qué es importante que los números sean positivos?

Respuesta : Porque nos permite saber que si en algún momento  $\text{suma} > x$ , entonces todas las ventanas que tienen el mismo valor de  $i$  y un  $j$  mayor tendrán una suma mayor y podemos no miraras (de alguna forma, podemos afirmar que “ya nos pasamos”).

```
// En iR, jR devolvemos la respuesta
void ventanaDeslizante(vector<int> &A, int x, int &iR, int &jR)
{
    int n = int(A.size()), j = 0, suma = 0; // La suma en [0,0) es 0
    for(int i = 0; i < n; i++) // para cada ventana que empieza en i:
    {
        // Ingresamos a la ventana hasta pasarnos con la suma desde i
        while (j < n && suma < x)
        {
            suma += A[j];
            j++;
        }
        // Al salir del while: (suma >= x) o (j == n)
        if (suma == x)
        {
            iR = i;
            jR = j;
        }
        // Al avanzar i, sale de la ventana
        suma -= A[i];
    }
}
```

- 1 Búsqueda Lineal
- 2 Búsqueda Binaria
- 3 Frase motivadora
- 4 Búsqueda Binaria sobre la respuesta
- 5 Ventanas Deslizantes
- 6 2SUM

## Problema:

Dado un arreglo de  $n$  números, y un número  $x$ . Queremos encontrar dos números del arreglo que sumen  $x$ , o reportar que no existe tal par.

### Problema:

Dado un arreglo de  $n$  números, y un número  $x$ . Queremos encontrar dos números del arreglo que sumen  $x$ , o reportar que no existe tal par.

### Ejemplos:

- $A = \overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}$ . Buscamos sumar  $x = 12$   
La respuesta es :



### Problema:

Dado un arreglo de  $n$  números, y un número  $x$ . Queremos encontrar dos números del arreglo que sumen  $x$ , o reportar que no existe tal par.

### Ejemplos:

- $A = \overset{0}{[7, \overset{1}{10, \overset{2}{4, \overset{3}{1, \overset{4}{9, \overset{5}{5, \overset{6}{9, \overset{7}{6}}}}}}]}$ . Buscamos sumar  $x = 12$   
La respuesta es : “Sumando  $A[0]$  y  $A[5]$  obtenemos 12”

## Problema:

Dado un arreglo de  $n$  números, y un número  $x$ . Queremos encontrar dos números del arreglo que sumen  $x$ , o reportar que no existe tal par.

## Ejemplos:

- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$ . Buscamos sumar  $x = 12$   
 La respuesta es : "Sumando  $A[0]$  y  $A[5]$  obtenemos 12"
- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$ . Buscamos sumar  $x = 4$   
 La respuesta es :

## Problema:

Dado un arreglo de  $n$  números, y un número  $x$ . Queremos encontrar dos números del arreglo que sumen  $x$ , o reportar que no existe tal par.

## Ejemplos:

- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$ . Buscamos sumar  $x = 12$   
La respuesta es : “Sumando  $A[0]$  y  $A[5]$  obtenemos 12”
- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$ . Buscamos sumar  $x = 4$   
La respuesta es : “No hay dos elementos de  $A$  que sumen 4”

## Problema:

Dado un arreglo de  $n$  números, y un número  $x$ . Queremos encontrar dos números del arreglo que sumen  $x$ , o reportar que no existe tal par.

## Ejemplos:

- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$ . Buscamos sumar  $x = 12$   
 La respuesta es : “Sumando  $A[0]$  y  $A[5]$  obtenemos 12”
- $A = [\overset{0}{7}, \overset{1}{10}, \overset{2}{4}, \overset{3}{1}, \overset{4}{9}, \overset{5}{5}, \overset{6}{9}, \overset{7}{6}]$ . Buscamos sumar  $x = 4$   
 La respuesta es : “No hay dos elementos de  $A$  que sumen 4”

## Observación clave

Es importante notar que el orden de los números en el arreglo  $A$  no juega ningún rol. Entonces podemos asumir que tienen el orden que nos convenga. En particular, podemos asumir que el arreglo está ordenado en forma creciente (u ordenarlo en  $\mathcal{O}(n \lg(n))$  )

## Observación clave

Es importante notar que el orden de los números en el arreglo  $A$  no juega ningún rol. Entonces podemos asumir que tienen el orden que nos convenga. En particular, podemos asumir que el arreglo está ordenado en forma creciente (u ordenarlo en  $\mathcal{O}(n \lg(n))$  )

- 1 Probar todos los pares de índices, y ver si esos dos números suman o no  $x$ . De existir un par que sí, reportarlo. Complejidad  $\mathcal{O}(n^2)$

## Observación clave

Es importante notar que el orden de los números en el arreglo A no juega ningún rol. Entonces podemos asumir que tienen el orden que nos convenga. En particular, podemos asumir que el arreglo está ordenado en forma creciente (u ordenarlo en  $\mathcal{O}(n \lg(n))$  )

- 1 Probar todos los pares de índices, y ver si esos dos números suman o no  $x$ . De existir un par que sí, reportarlo. Complejidad  $\mathcal{O}(n^2)$
- 2 Utilizar una variante de la técnica que vimos recién para resolver el problema. En lugar de tener dos índices que siempre aumentan, ahora tendremos dos índices. Uno siempre aumenta, el otro siempre disminuye. Por lo tanto también haremos a lo sumo  $2n$  pasos. Complejidad  $\mathcal{O}(n)$ .

Buscamos sumar  $x = 12$

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [1, & 4, & 5, & 6, & 7, & 9, & 9, & 10] \end{matrix}$$



# Paso 1

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x$$

$$A = \begin{matrix} & i & & & & & & & j \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A = [ & 1, & 4, & 5, & 6, & 7, & 9, & 9, & 10 ] \end{matrix}$$

# Paso 1

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x \rightarrow \text{Aumento } i$$

$$A = \begin{matrix} & i & & & & & & & j \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A = [ & 1, & 4, & 5, & 6, & 7, & 9, & 9, & 10 ] \end{matrix}$$

## Paso 2

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{14} > x$$

$$A = [1, \overset{i}{4}, 5, 6, 7, 9, 9, \overset{j}{10}]$$

## Paso 2

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{14} > x \rightarrow \text{Disminuyo } j$$

$$A = [1, \overset{i}{4}, 5, 6, 7, 9, 9, \overset{j}{10}]$$

## Paso 3

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x$$

$$A = [1, \overset{i}{4}, 5, 6, 7, 9, \overset{j}{9}, 10]$$

## Paso 3

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x \rightarrow \text{Disminuyo } j$$

$$A = [1, \overset{i}{4}, 5, 6, 7, 9, \overset{j}{9}, 10]$$

## Paso 4

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x$$

$$A = [1, \overset{i}{\underset{\text{blue}}{4}}, 5, 6, 7, \overset{j}{\underset{\text{blue}}{9}}, 9, 10]$$

## Paso 4

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{13} > x \rightarrow \text{Disminuyo } j$$

$$A = [1, \overset{i}{4}, 5, 6, 7, \overset{j}{9}, 9, 10]$$



## Paso 5

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x$$

$$A = [1, \overset{i}{\underset{0}{4}}, 5, 6, \overset{j}{\underset{4}{7}}, 9, 9, 10]$$

## Paso 5

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{11} < x \rightarrow \text{Aumento } i$$

$$A = [1, \overset{i}{\underset{4}{4}}, 5, 6, \overset{j}{\underset{7}{7}}, 9, 9, 10]$$

## Paso 6

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{12} = x$$

$$A = [1, 4, \overset{i}{\underset{5}{\textcolor{blue}{5}}}, 6, \overset{j}{\underset{7}{\textcolor{blue}{7}}}, 9, 9, 10]$$

## Paso 6

Buscamos sumar  $x = 12$

$$\overbrace{A[i] + A[j]}^{12} = x \rightarrow \text{Podemos terminar}$$

$$A = [1, 4, \overset{i}{\underset{5}{5}}, 6, \overset{j}{\underset{7}{7}}, 9, 9, 10]$$

```
// En iR, jR devolvemos la respuesta
void ventanaDeslizante(vector<int> &A, int x, int &iR,
                      int &jR)
{
    int n = int(A.size()), j = n-1;
    for(int i = 0; i < n; i++) // fijando el extremo i:
    { // mientras la suma sea > x, disminuimos j
        while (j > i && A[i] + A[j] > x)
            j--;
        // Sale del while : (A[i] + A[j] <= x) o (j == i)
        if (j > i && A[i] + A[j] == x)
        {
            iR = i;
            jR = j;
        }
    }
}
```

## Fuentes

- Competitive Programmer's Handbook
- Mezcla de charla Gutty y Julian Ferres

## Problemitas

- Factory Machines
- Magic powder - 1
- Array Division
- Multiplication Table

Pueden consultarme durante esta semana, o me pueden enviar un mail a:

- [mramos@frsf.utn.edu.ar](mailto:mramos@frsf.utn.edu.ar)