

CURSO DE PROGRAMACIÓN COMPETITIVA

ALGORITMOS VORACES



CURSO DE PROGRAMACIÓN COMPETITIVA

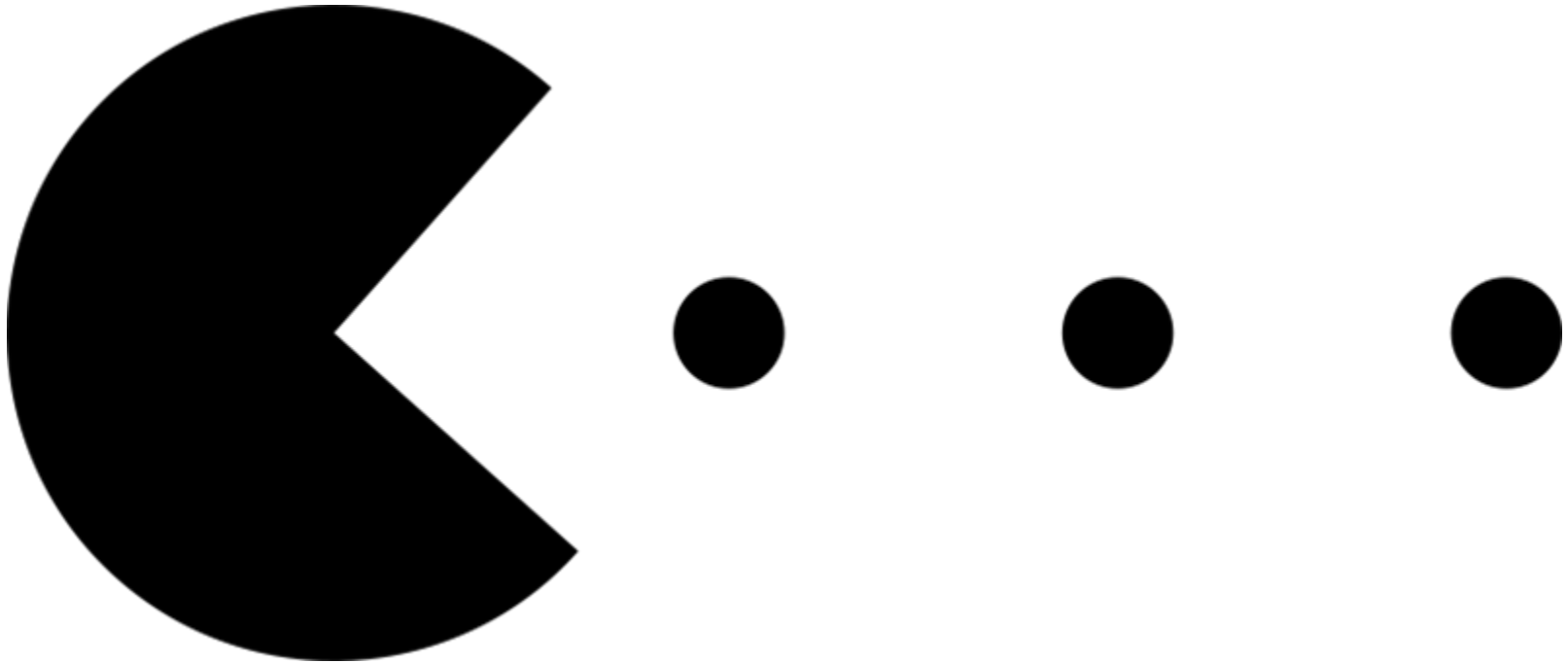
URJC - 2025

Organizadores:

- Isaac Lozano (isaac.lozano@urjc.es)
- **Sergio Salazar** (sergio.salazar@urjc.es)
- Adaya Ruiz (am.ruiz.2020@alumnos.urjc.es)
- Eva Gómez (e.gomezf.2020@alumnos.urjc.es)
- Lucas Martín (lucas.martin@urjc.es)
- Iván Penedo (ivan.penedo@urjc.es)
- Alicia Pina (alicia.pina@urjc.es)
- Sara García (sara.garcia@urjc.es)
- **Raúl Fauste** (r.fauste.2020@alumnos.urjc.es)
- **Alejandro Mayoral** (a.mayoralg.2020@alumnos.urjc.es)
- David Orna (de.orna.2020@alumnos.urjc.es)



Algoritmos Voraces



Algoritmos Voraces

¿Qué características tiene que tener un problema para ser resuelto de manera voraz (greedy)?

- La solución debe construirse paso a paso
- La mejor decisión en cada paso debe conducir a la mejor decisión global



Algoritmos Voraces

¿Qué características tiene que tener un problema para ser resuelto de manera voraz (greedy)?

- La solución debe construirse paso a paso
- La mejor decisión en cada paso debe conducir a la mejor decisión global

SIEMPRE



Algoritmos Voraces

¿Cómo sabemos si un algoritmo greedy funciona?

- Encontrar una demostración formal que certifica que es óptimo
- Encontrar un contraejemplo que indique que NO es óptimo
- **EXPERIENCIA**



Algoritmos Voraces

¿Cómo sabemos si un algoritmo greedy funciona?

- Encontrar una demostración formal que certifica que es óptimo
- Encontrar un contraejemplo que indique que NO es óptimo
- **EXPERIENCIA**



¡Complicado! Se premia la velocidad, solemos tener una idea de por qué funcionará



Algoritmos Voraces

¿Cómo sabemos si un algoritmo greedy funciona?

- Encontrar una demostración formal que certifica que es óptimo
- Encontrar un contraejemplo que indique que NO es óptimo
- **EXPERIENCIA**




Sirve para descartar propuestas, pero no tiene porque conducir a una solución.



Algoritmos Voraces

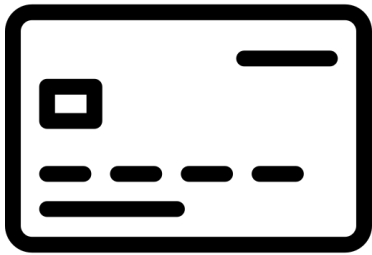
¿Cómo sabemos si un algoritmo greedy funciona?

- Encontrar una demostración formal que certifica que es óptimo
- Encontrar un contraejemplo que indique que NO es óptimo
- **EXPERIENCIA** 
 - Trabaja en equipo.
 - Prueba casos
 - Es mejor que nada



Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas



138

50

20

10

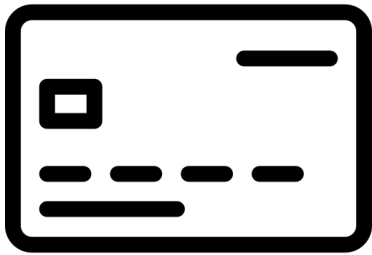
5

1

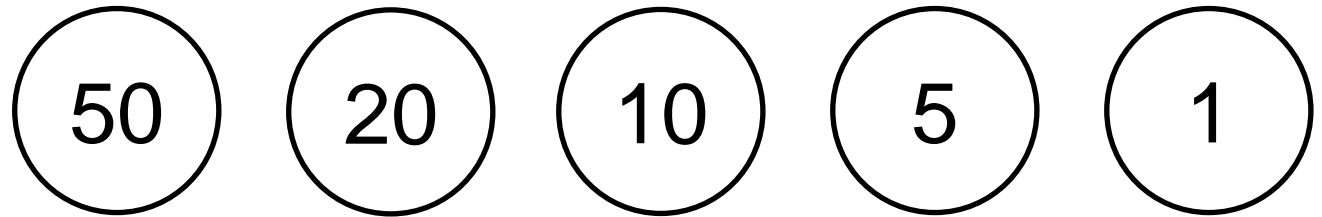


Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas



138

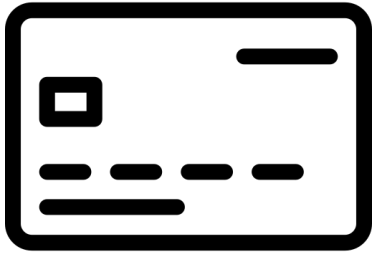


Esto siempre tiene solución, podemos intercambiar todas las monedas por la de 1.



Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas



138

50

20

10

5

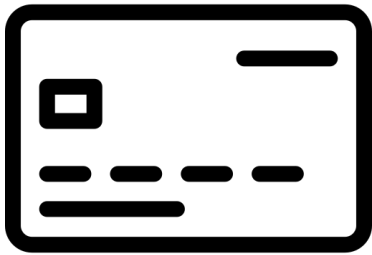
1

Solución Voraz:
Coger la moneda de más valor.

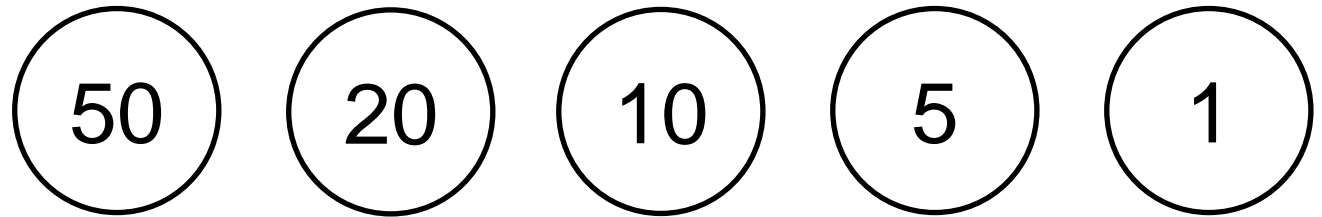


Algoritmos Voraces - Ejemplo

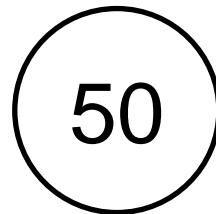
Minimizar el número de monedas utilizadas



88

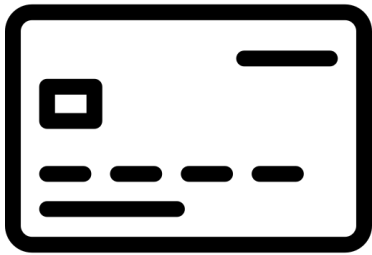


Solución Voraz:
Coger la moneda de más valor.

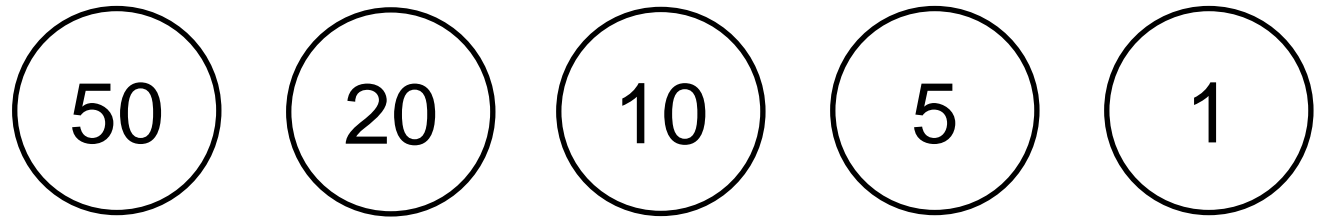


Algoritmos Voraces - Ejemplo

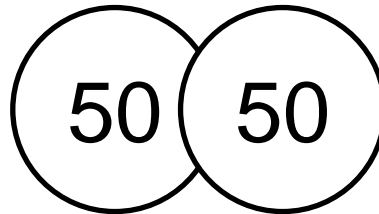
Minimizar el número de monedas utilizadas



38

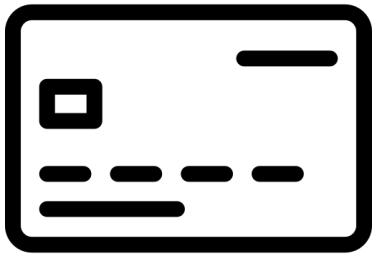


Solución Voraz:
Coger la moneda de más valor.

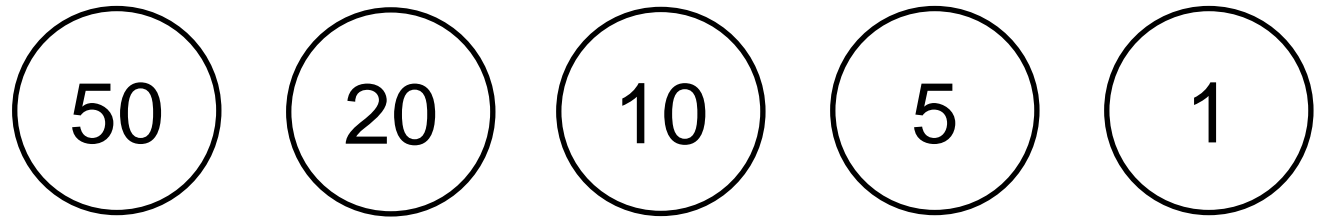


Algoritmos Voraces - Ejemplo

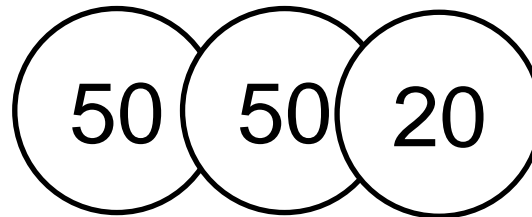
Minimizar el número de monedas utilizadas



18

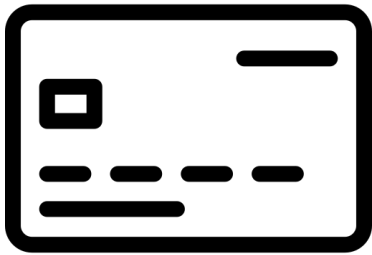


Solución Voraz:
Coger la moneda de más valor.

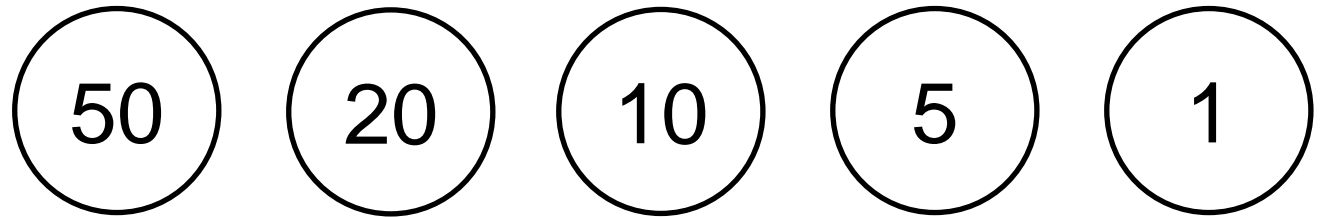


Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas

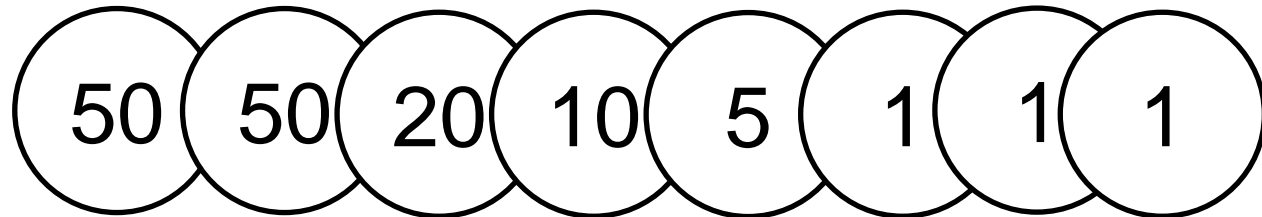
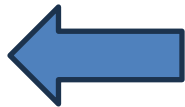


0



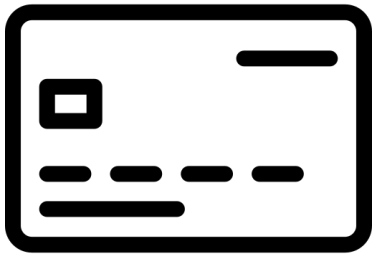
Solución Voraz:
Coger la moneda de más valor.

8

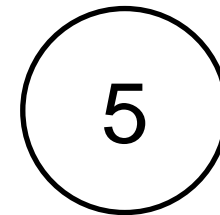
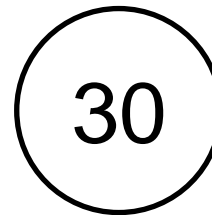
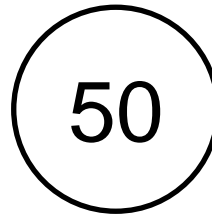


Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas

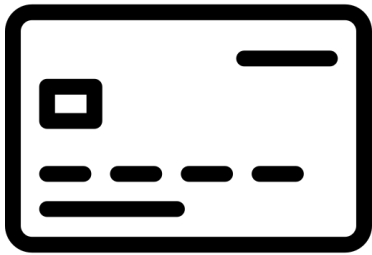


65

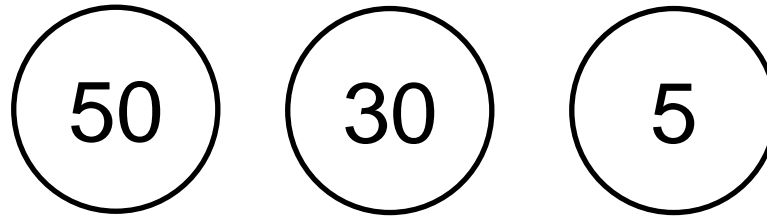


Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas



65

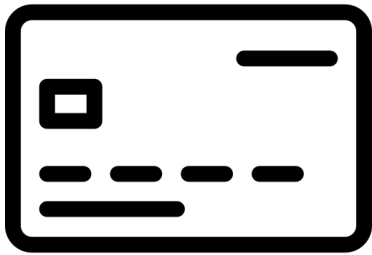


Solución Voraz:
Coger la moneda de más valor.



Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas



15

50

30

5

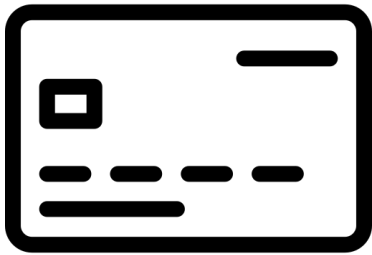
Solución Voraz:
Coger la moneda de más valor.

50

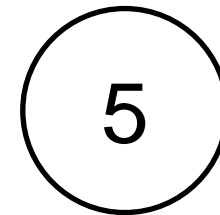
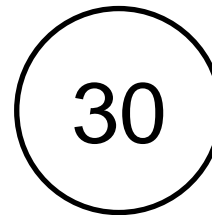
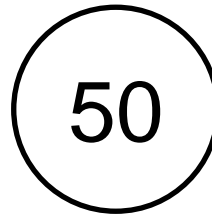


Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas



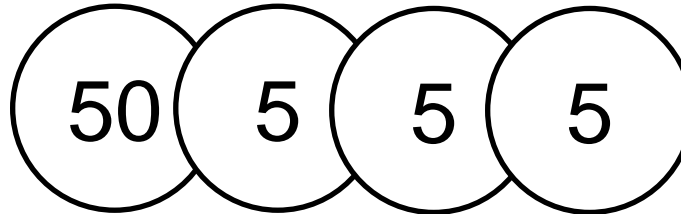
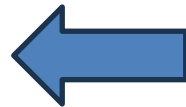
65



Solución Voraz:

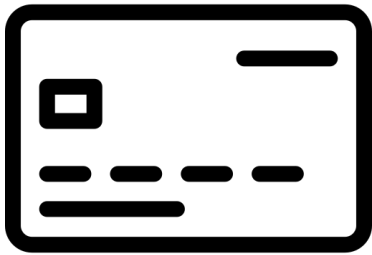
Coger la moneda de más valor.

4

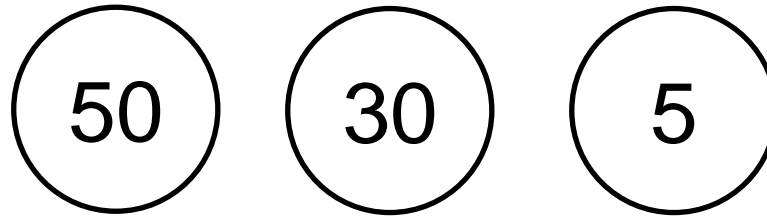


Algoritmos Voraces - Ejemplo

Minimizar el número de monedas utilizadas

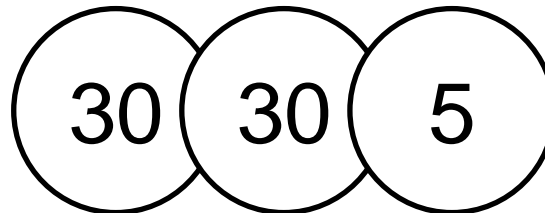
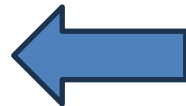


65



Solución NO voraz:

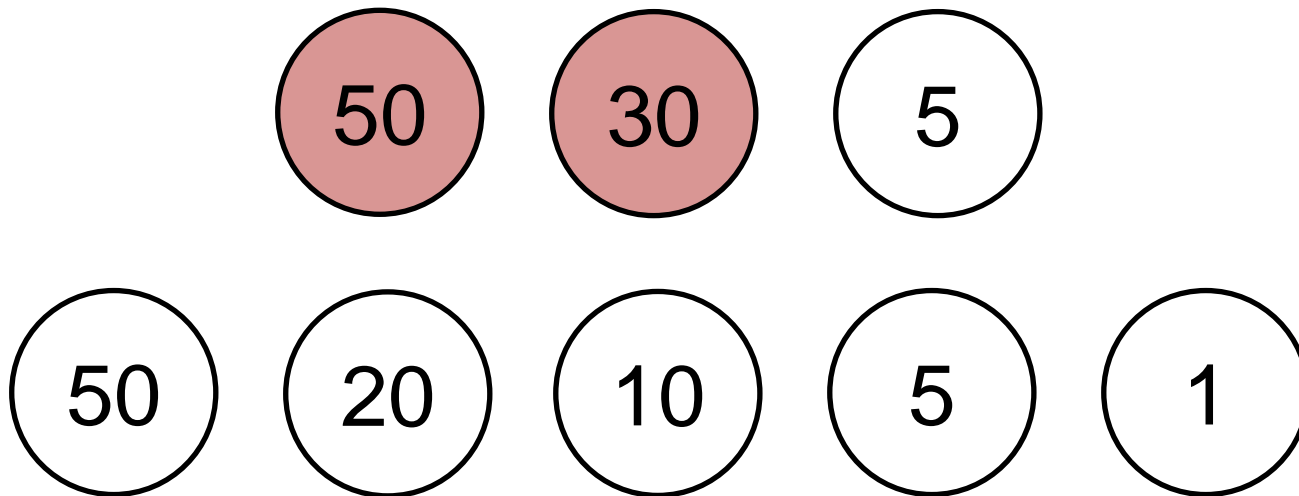
3



Algoritmos Voraces - Ejemplo

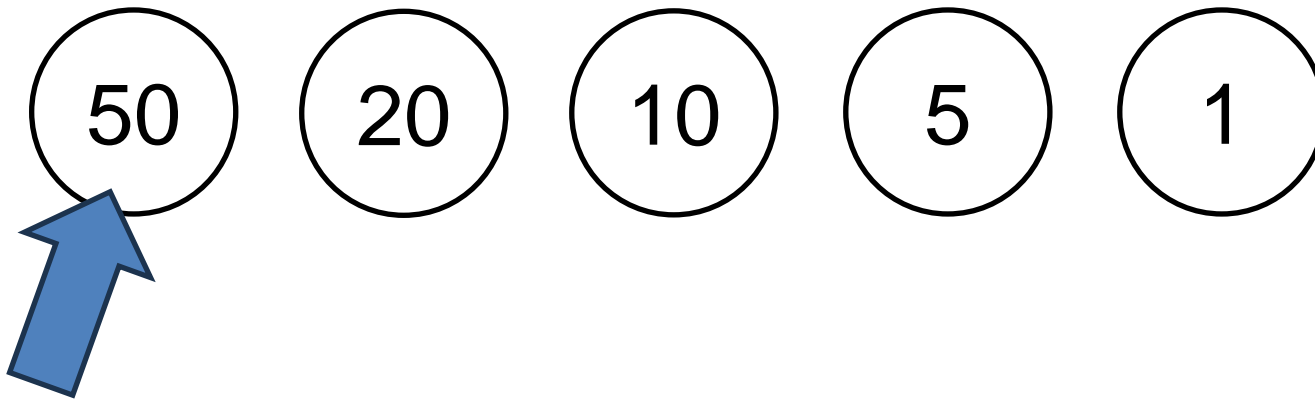
¿Por qué?

Es posible que coger dos monedas pequeñas sea más rentable que coger 1 grande



Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?

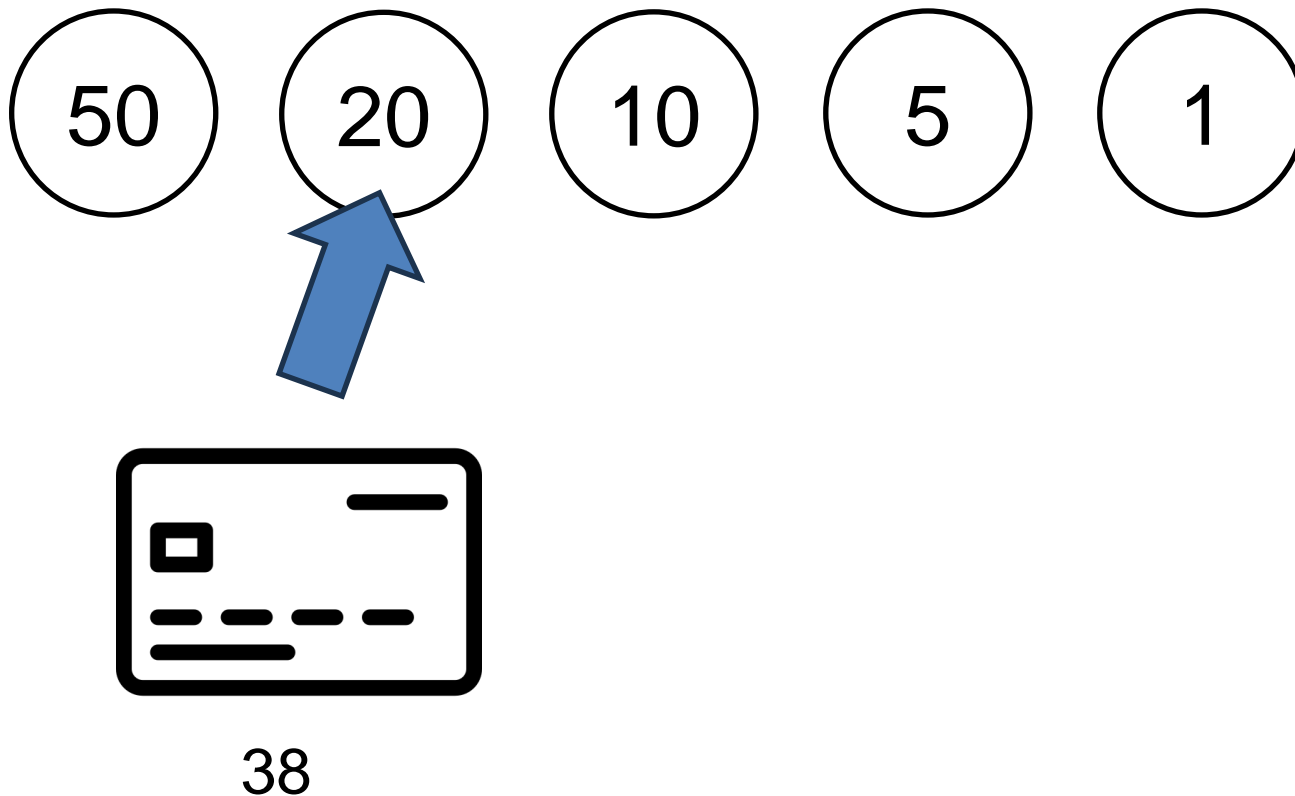


138



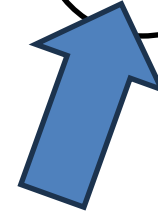
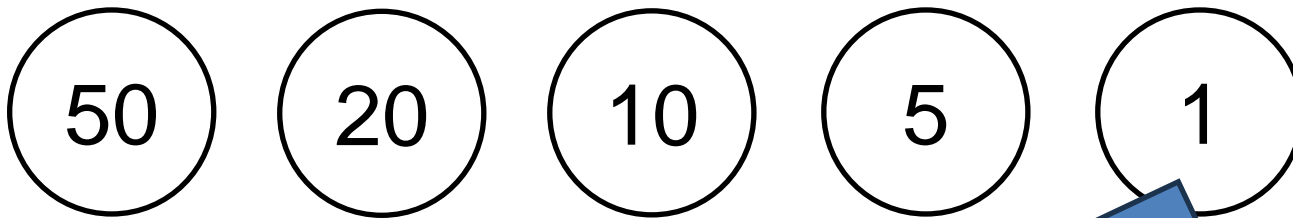
Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?

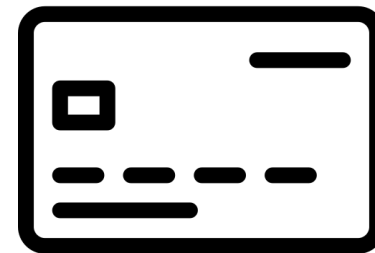


Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



????

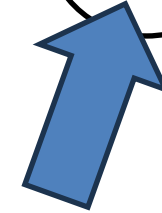
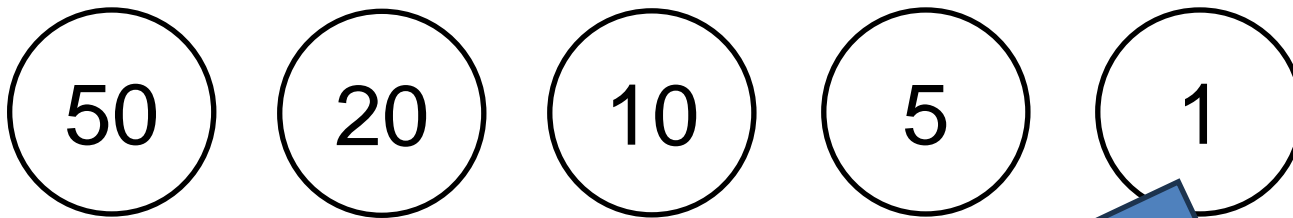


3

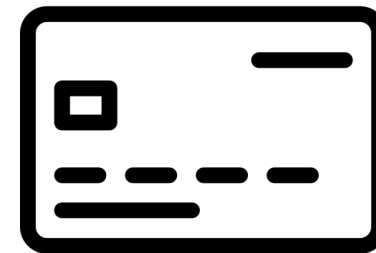


Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



$O(n)$

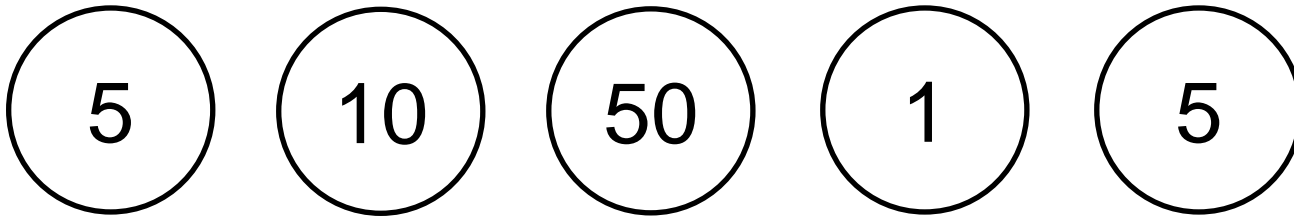


3

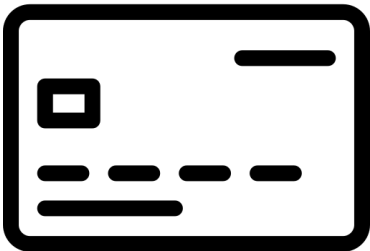


Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



?

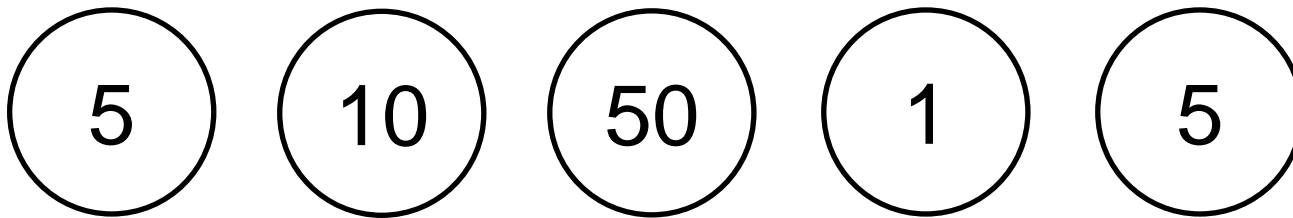


138

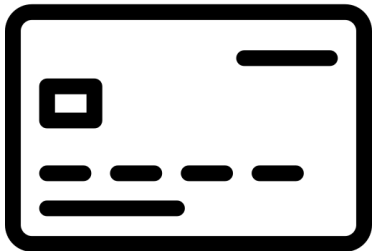


Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



←————→ $O(n)$

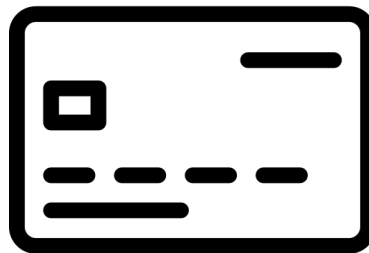
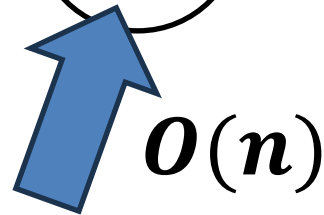
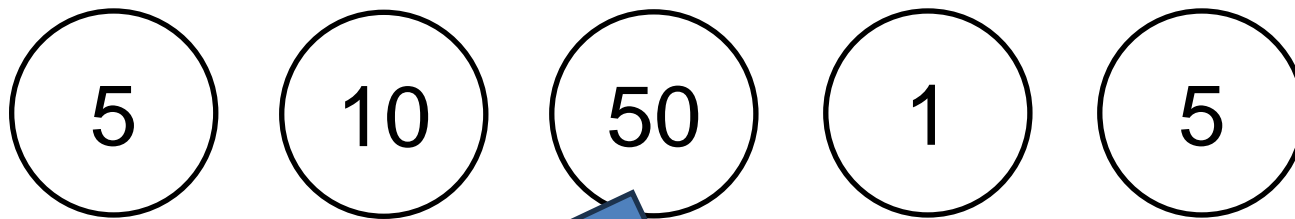


138



Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



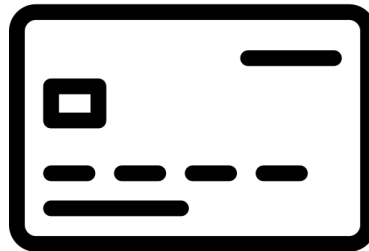
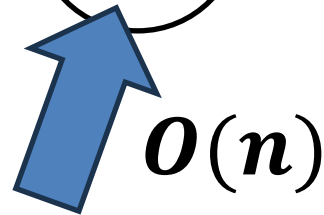
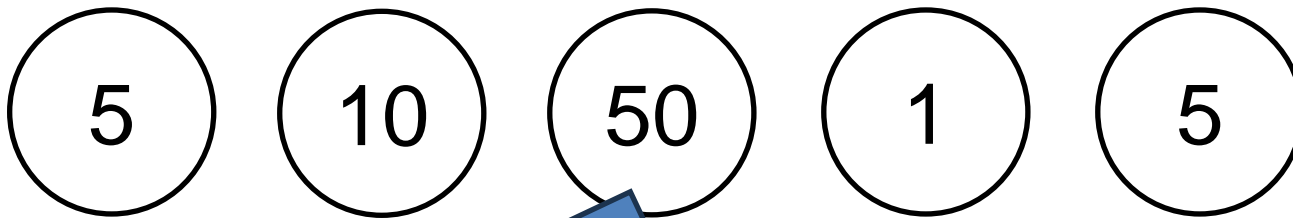
???

138



Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



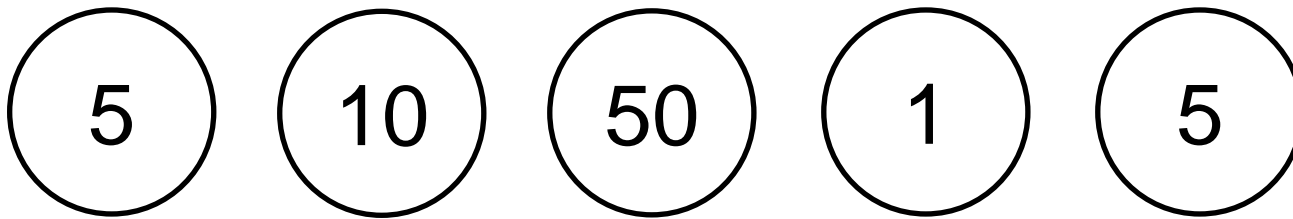
$O(n^2)$

138

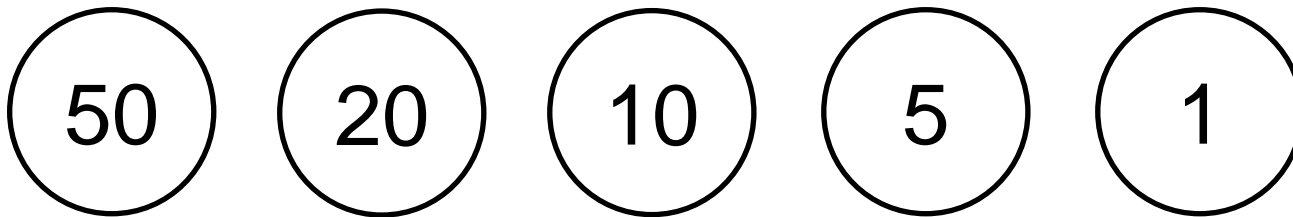


Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?

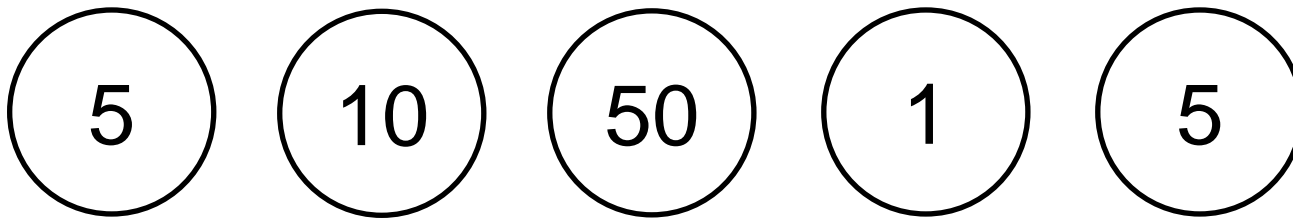


TRUCO: ¡Ordenarlo!



Algoritmos Voraces - Ejemplo

¿Cuál sería la complejidad?



TRUCO: ¡Ordenarlo! $O(n \cdot \log n)$



$$O(n + n \cdot \log n) = O(n \cdot \log n)$$



Algoritmos Voraces - SORT

Cuando buscamos algo en orden, ordenar la entrada puede resultar en una buena idea.

$O(n \cdot \log n)$

```
numeros = [30, 5, 1]
```

```
numeros.sort() #[1, 5, 30]
```

```
numeros.sort(reverse = True) #[30, 5, 1]
```

```
numeros.sort(key = isPar) #[5, 1, 30]
```



Algoritmos Voraces – Planting Trees

<https://open.kattis.com/problems/plantingtrees>

4
2 3 4 3 ➡ 7



 CPU TIME LIMIT
3 seconds

$1 < N < 100.000$

$1 < t < 1.000.000$

Ojo! Si planto un árbol hoy, “empieza” a crecer mañana.



Algoritmos Voraces – Planting Trees

<https://open.kattis.com/problems/plantingtrees>

4
2 3 4 3  7



CPU TIME LIMIT
3 seconds

$1 < N < 100.000$

$1 < t < 1.000.000$

$O(N \cdot \log N)$



Algoritmos Voraces – Planting Trees

<https://open.kattis.com/problems/plantingtrees>

4
2 3 4 3 ➡ 7



Y si... ???

$1 < N < 1.000.000$

$1 < t < 1.000.000$



Algoritmos Voraces – Planting Trees

<https://open.kattis.com/problems/plantingtrees>

4
2 3 4 3 ➡ 7



Y si... ???

$1 < N < 1.000.000$

$1 < t < 1.000.000$

Existe una solución en

$O(N + t)$



Búsqueda Binaria

- Definición
- Dónde aplicarla (Espacio de búsqueda)
- Ejemplos

¡Vuestro nuevo
Mejor Amigo!



Búsqueda Binaria - Definición

Es un **algoritmo de búsqueda** para encontrar un elemento en un espacio de búsqueda **monótono**.



Búsqueda Binaria - Definición

Es un **algoritmo de búsqueda** para encontrar un elemento en un espacio de búsqueda **monótono**.

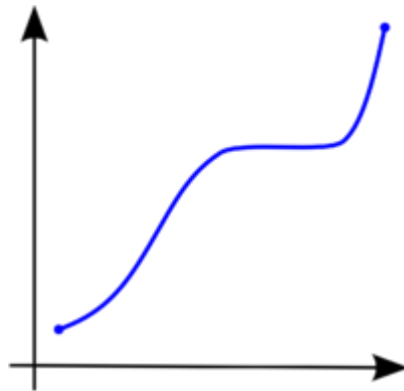
¿Qué significa monótono?



Búsqueda Binaria - Definición

Un **espacio de búsqueda monótono u ordenado** es aquel en el que sus elementos se hallan ordenados bajo un cierto criterio.

En **matemáticas**, por ejemplo, una **función es monótona creciente** si para cualquier x , y con $x < y$ se tiene que $f(x) \leq f(y)$.



Búsqueda Binaria

¡Perfecto!

Búsqueda Binaria → Algoritmo de búsqueda

Pero ¿cómo funciona?



Búsqueda Binaria

¡Perfecto!

Búsqueda Binaria → Algoritmo de búsqueda

Pero ¿cómo funciona?

La **idea** del algoritmo es **ir descartando mitades** donde sabemos que **NO podemos encontrar la solución**



Búsqueda Binaria - Ejemplo

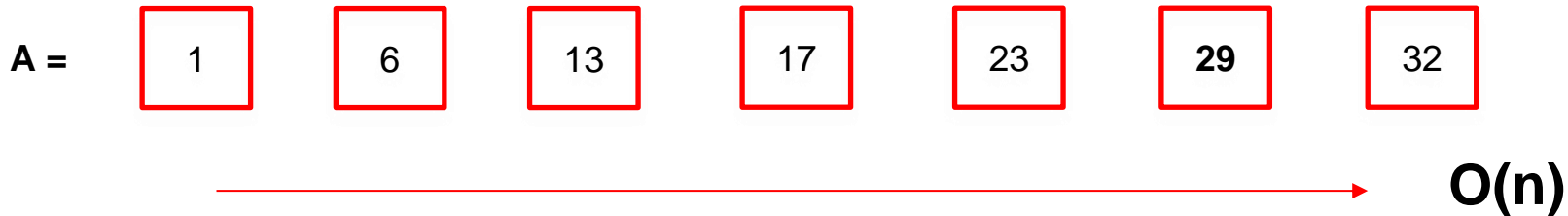
Queremos **encontrar** el elemento **$x = 29$** en una **lista ordenada**



Búsqueda Binaria - Ejemplo

Queremos **encontrar** el elemento $x = 29$ en una **lista ordenada**

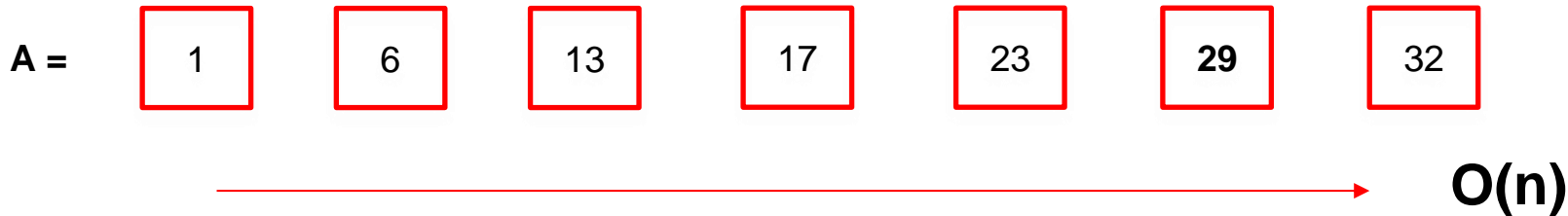
1º Idea → Recorrer toda la lista hasta encontrarlo.



Búsqueda Binaria - Ejemplo

Queremos **encontrar** el elemento $x = 29$ en una **lista ordenada**

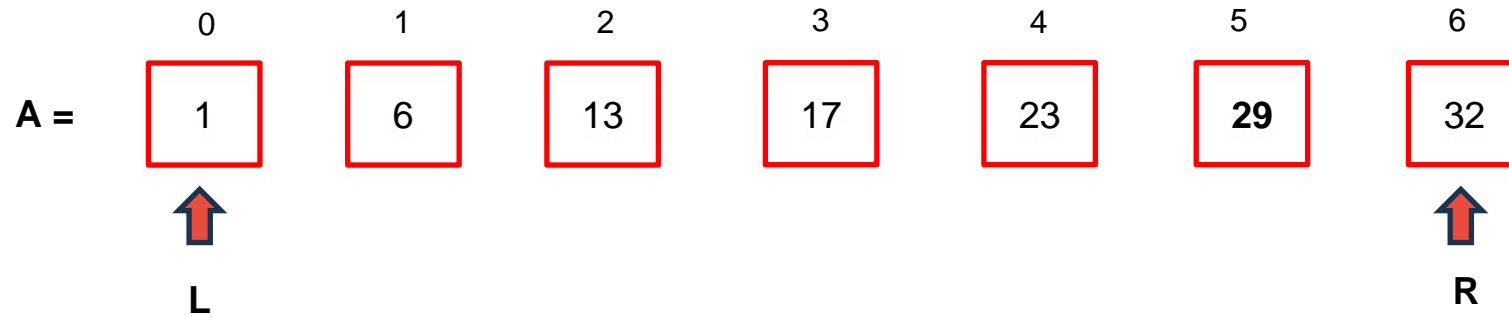
1º Idea → Recorrer toda la lista hasta encontrarlo.



2º Idea → Búsqueda Binaria



Búsqueda Binaria - Ejemplo



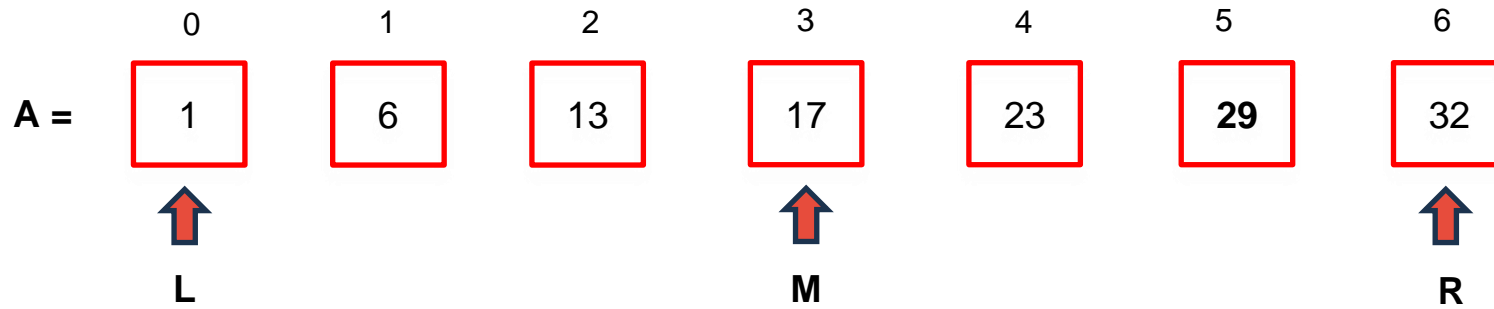
X = 29

L = 0

R = 6



Búsqueda Binaria - Ejemplo



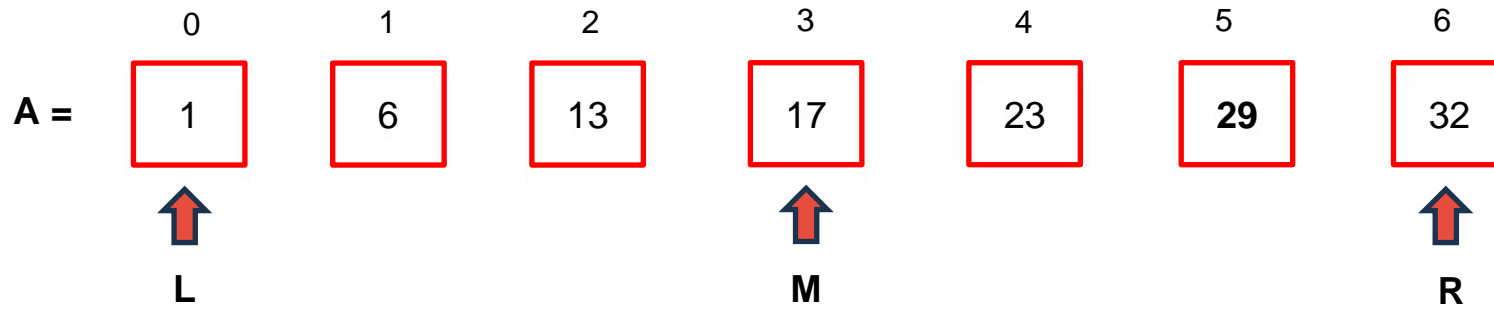
$$M = (L+R) / 2$$

X = 29
L = 0
R = 6

$$M = (0+6)/2 = 3$$



Búsqueda Binaria - Ejemplo



¿Es $A[M] = X$?

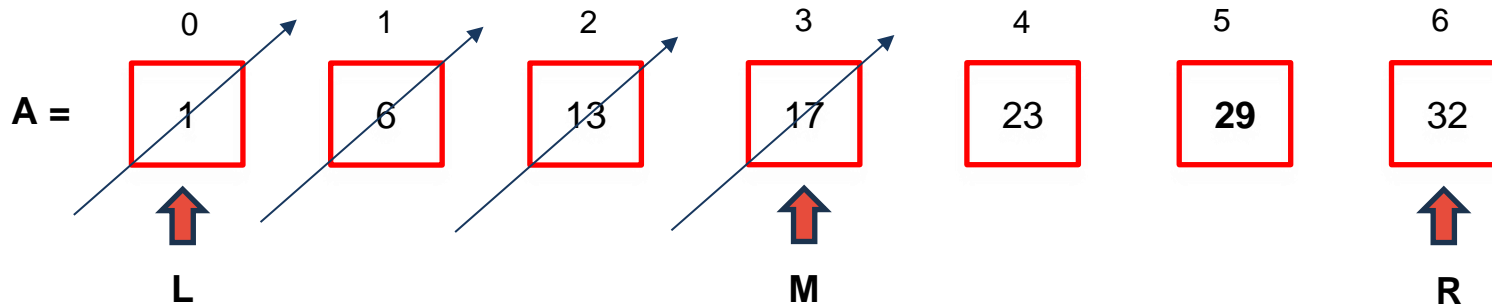
¿Es $A[M] < X$?

¿Es $A[M] > X$?

X = 29
L = 0
R = 6
M = 3



Búsqueda Binaria - Ejemplo



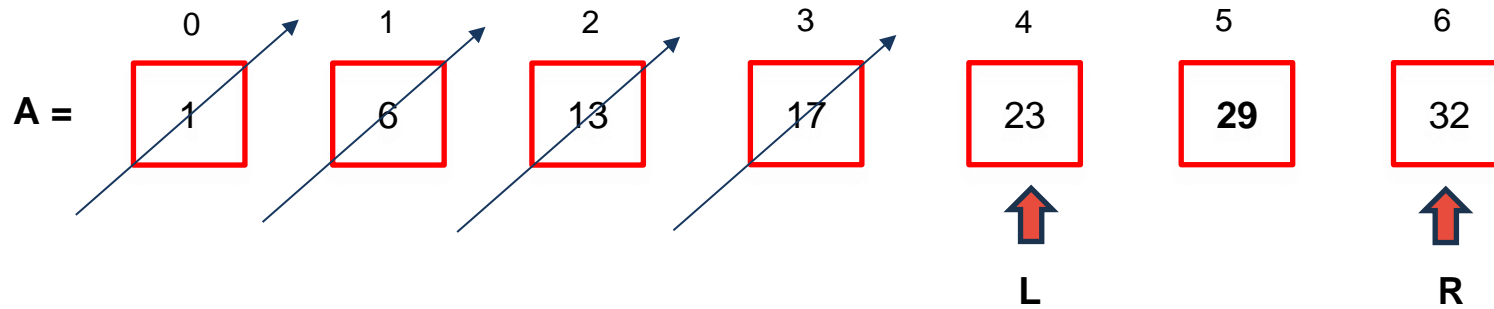
¡ $A[M] < X$!

Luego todos los elementos con índice menor que M también son menores que X, luego podemos NO considerarlos

X = 29
L = 0
R = 6
M = 3



Búsqueda Binaria - Ejemplo



Entonces $L = M+1$ y **vuelta a empezar**

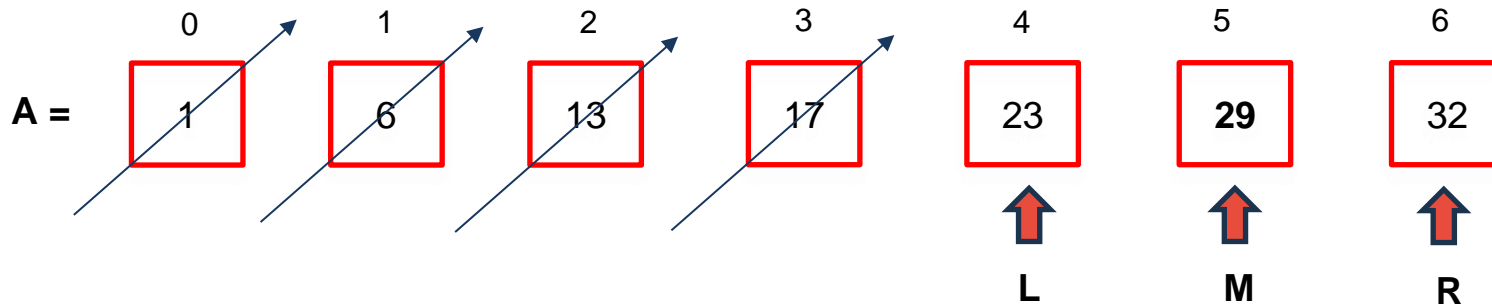
X = 29

L = 4

R = 6



Búsqueda Binaria - Ejemplo

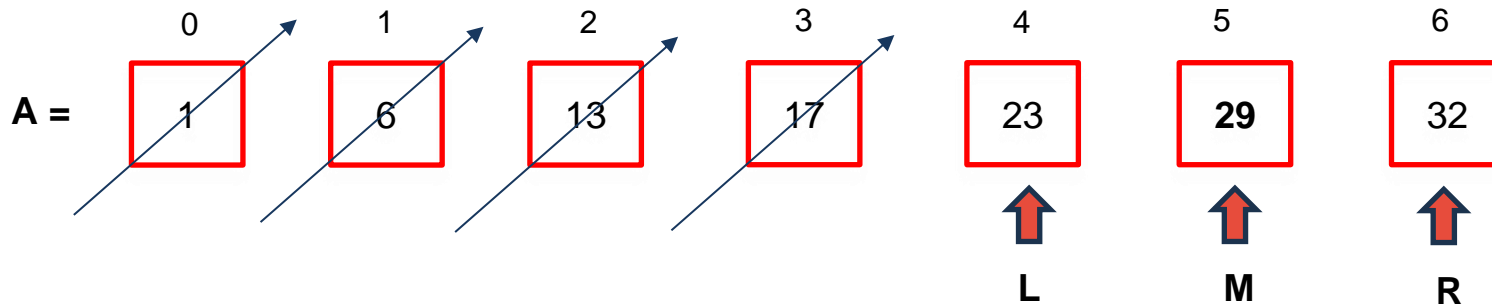


$$M = (L+R) / 2$$
$$M = (4 + 6) / 2$$

X = 29
L = 4
R = 6
M = 5



Búsqueda Binaria - Ejemplo



$$M = (L+R) / 2$$

$$M = (4 + 6) / 2$$

¿Es $A[M] = X$?

¿Es $A[M] < X$?

¿Es $A[M] > X$?

$X = 29$

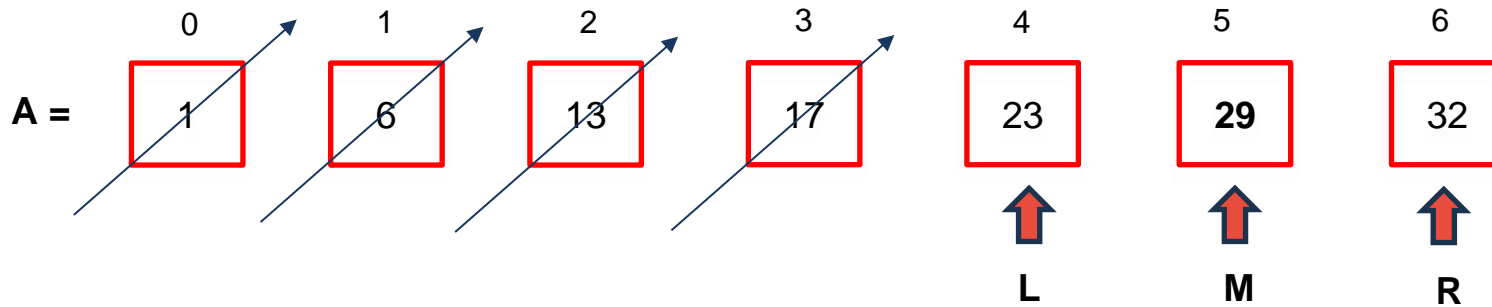
$L = 4$

$R = 6$

$M = 5$



Búsqueda Binaria - Ejemplo



$$M = (L+R) / 2$$
$$M = (4 + 6) / 2$$

¡ A[M] = X !

Luego devolvemos como **solución M**

X = 29
L = 4
R = 6
M = 5



Búsqueda Binaria

¿Complejidad?



Búsqueda Binaria

¿Complejidad?

$O(\log(n))$



Búsqueda Binaria

Pero....

¿Y si el elemento que nos piden buscar **NO** está en la lista?



Búsqueda Binaria

0	1	2	3	4	5	6
1	6	13	17	23	29	32

Encontrar el índice máximo del elemento menor o igual que $X = 8$



Búsqueda Binaria

0	1	2	3	4	5	6
1	6	13	17	23	29	32

Encontrar el índice máximo del elemento menor o igual que $X = 8$



Máximo i tal que $A[i] \leq X$



Búsqueda Binaria

0	1	2	3	4	5	6
1	6	13	17	23	29	32

Encontrar el índice máximo del elemento menor o igual que $X = 8$



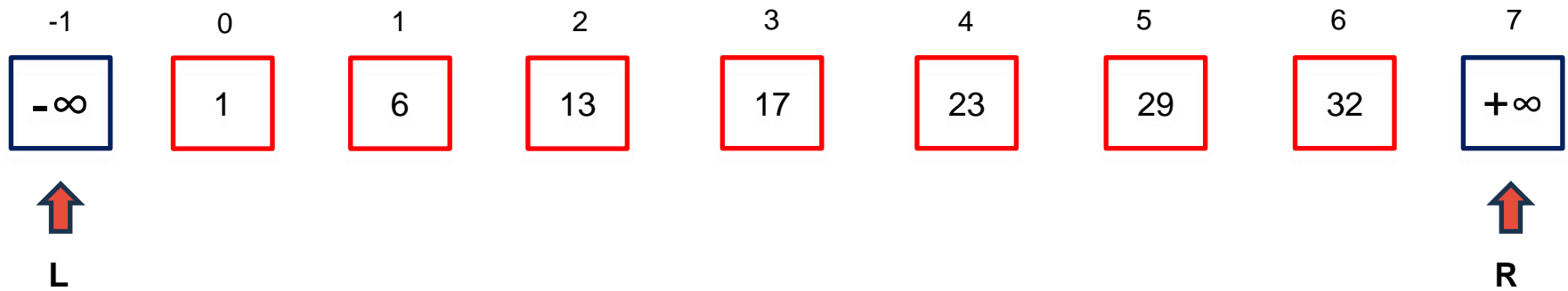
Máximo i tal que $A[i] \leq X$

Se aplica una búsqueda binaria manteniendo que:

$$\begin{aligned} A[L] &\leq X \\ A[R] &> X \end{aligned}$$



Búsqueda Binaria



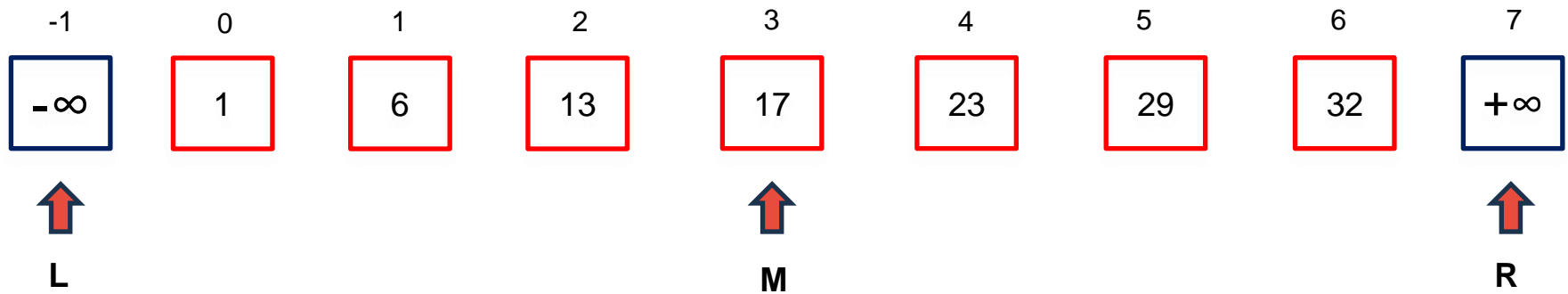
Invariantes:

$A[L] \leq X$

$A[R] > X$



Búsqueda Binaria



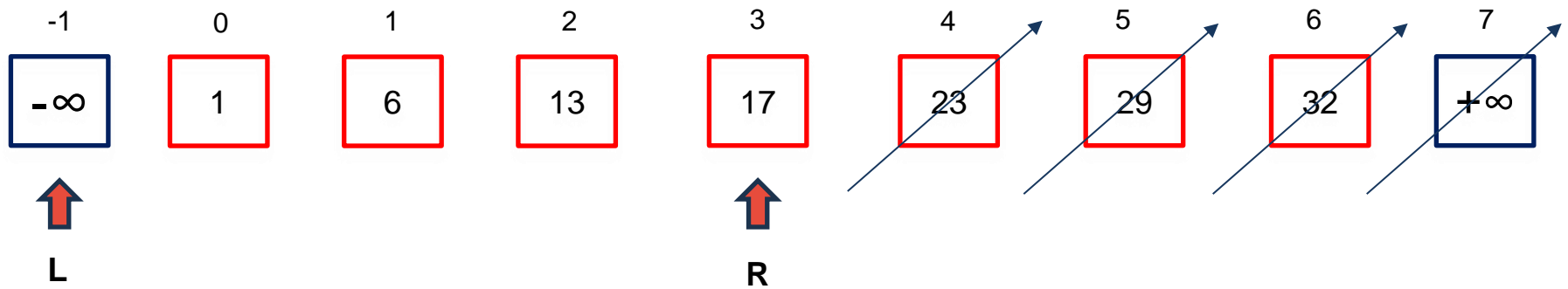
Invariantes:

$A[L] \leq X$

$A[R] > X$



Búsqueda Binaria



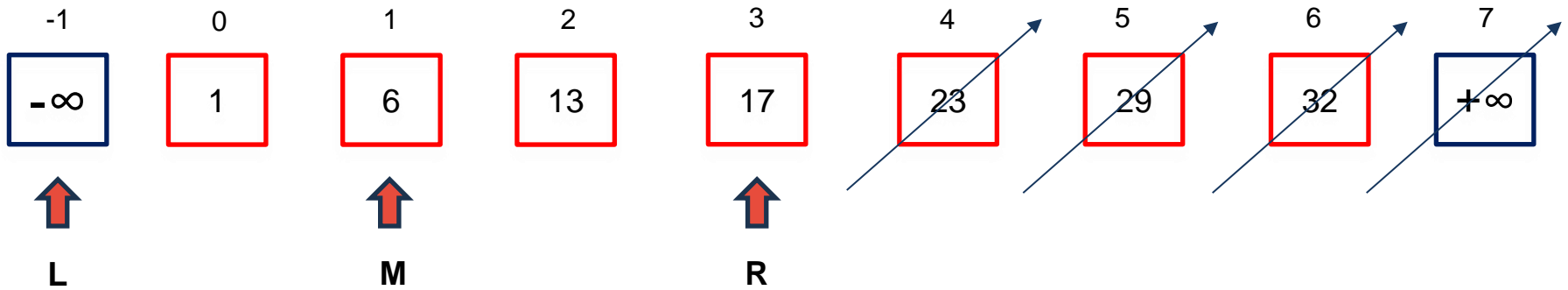
Invariantes:

$$A[L] \leq X$$

$$A[R] > X$$



Búsqueda Binaria



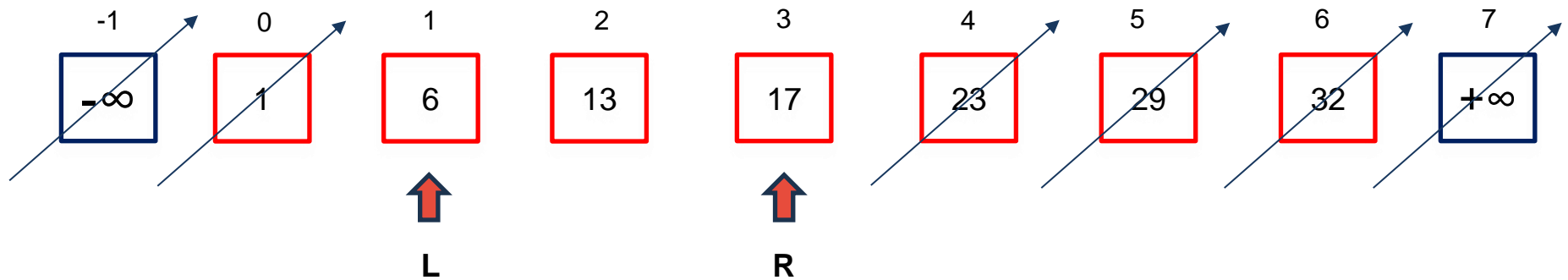
Invariantes:

$$A[L] \leq X$$

$$A[R] > X$$



Búsqueda Binaria



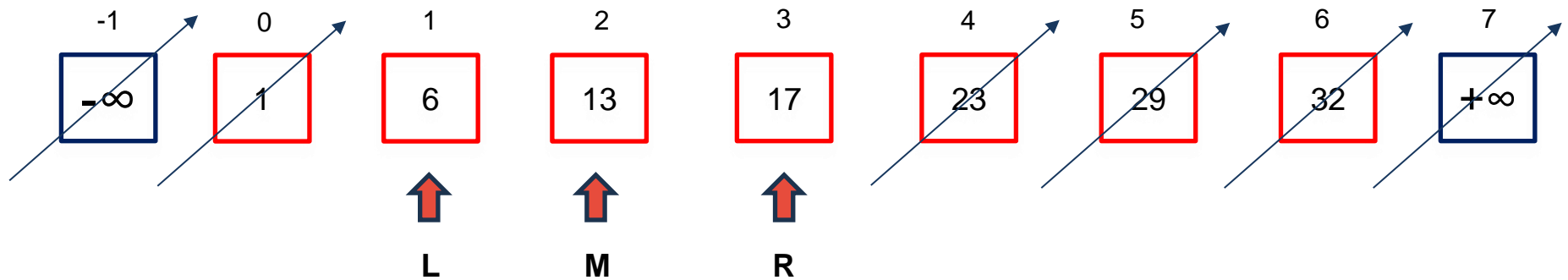
Invariantes:

$$A[L] \leq X$$

$$A[R] > X$$



Búsqueda Binaria



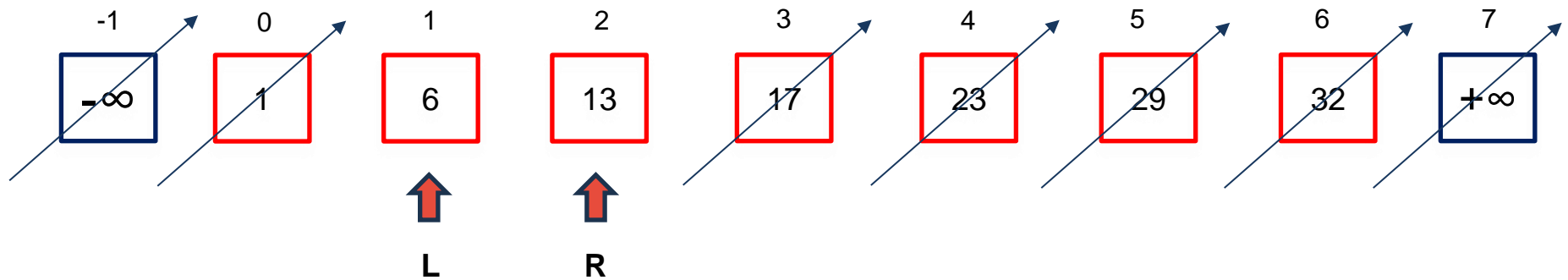
Invariantes:

$$A[L] \leq X$$

$$A[R] > X$$



Búsqueda Binaria



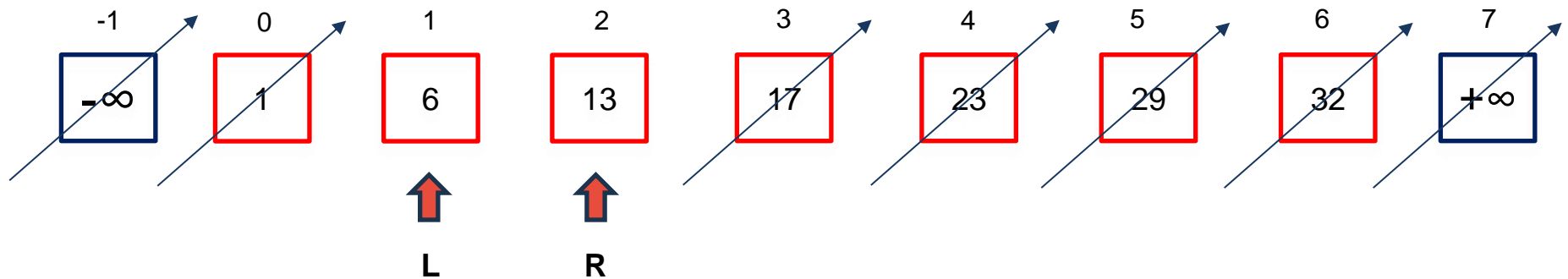
Invariantes:

$$A[L] \leq X$$

$$A[R] > X$$



Búsqueda Binaria



El índice del elemento menor o igual que X se encuentra en L

Luego devolvemos L como solución

Invariantes:

$$A[L] \leq X$$

$$A[R] > X$$



Búsqueda Binaria

Y el caso contrario, ¿cómo sería?

0	1	2	3	4	5	6
1	6	13	17	23	29	32

Encontrar el índice mínimo del elemento mayor o igual que $X = 8$



Búsqueda Binaria

Y el caso contrario, ¿cómo sería?

0	1	2	3	4	5	6
1	6	15	1	23	29	32

Encontrar el índice mínimo del elemento mayor o igual que $X = 8$



Sliding Window

Útil para problemas en los que queremos comprobar una característica de los intervalos de una lista.

Tenemos dos índices para marcar los límites del intervalo. Puede ser de tamaño fijo o variable.

Se evitan cálculos redundantes y se reduce la complejidad.



Sliding Window - Tamaño fijo

- Lista de tamaño n
- Ventana de tamaño $m < n$
- Sin sliding window: complejidad $O(n*m)$



- Con siliding window: complejidad $O(n)$



Sliding Window - Tamaño fijo

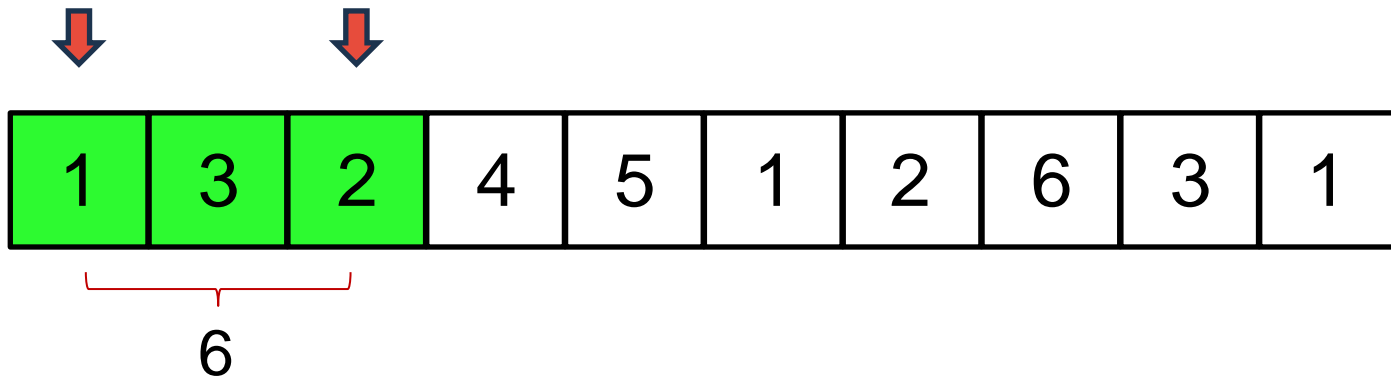
1	3	2	4	5	1	2	6	3	1
---	---	---	---	---	---	---	---	---	---

Vamos a ver un ejemplo con:

- $n=10$
- $m=3$
- Suma máxima en subintervalos de longitud m



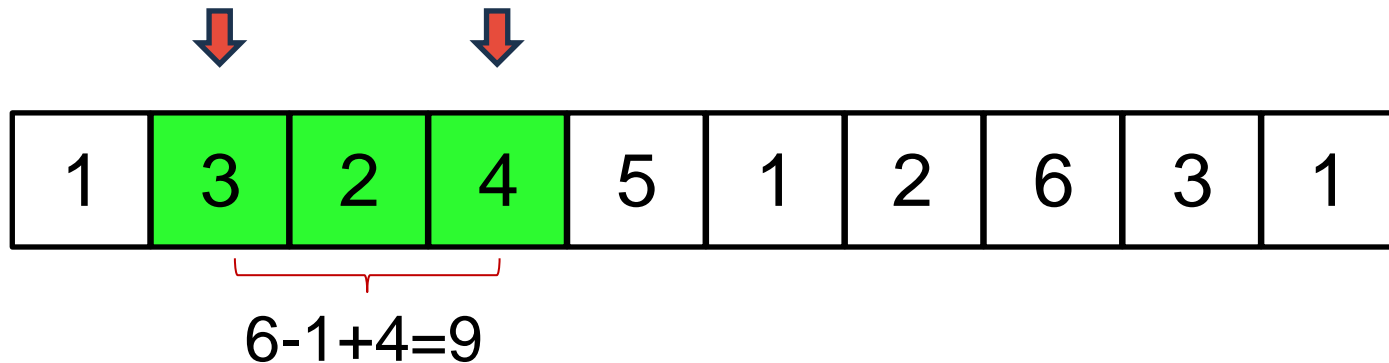
Sliding Window - Tamaño fijo



- $i = 0$
- $j = 2$
- Máximo = 6



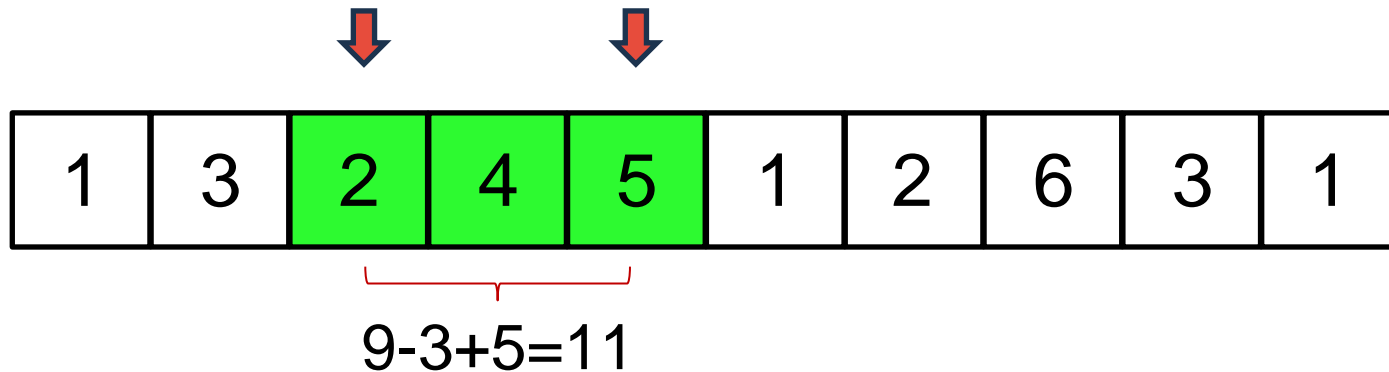
Sliding Window - Tamaño fijo



- $i = 1$
- $j = 3$
- Máximo = 9



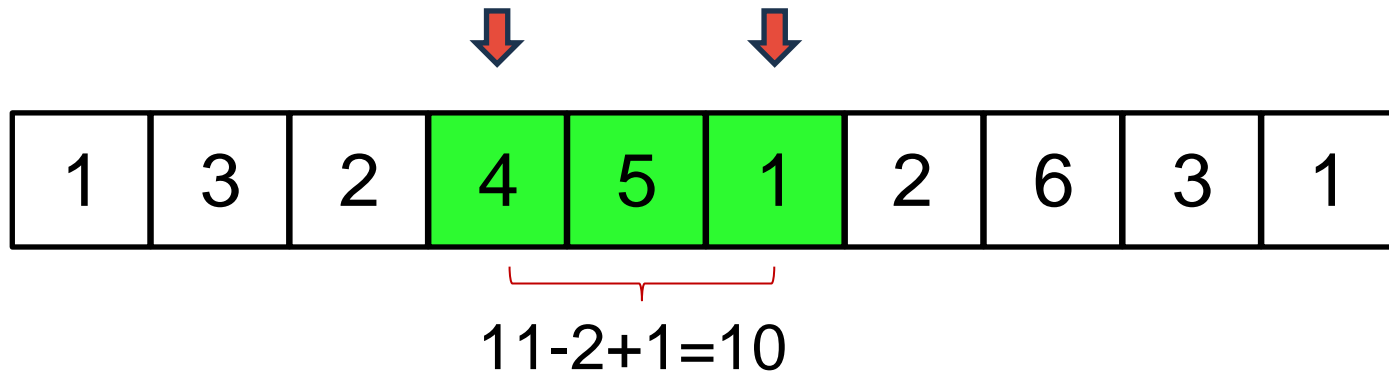
Sliding Window - Tamaño fijo



- $i = 2$
- $j = 4$
- Máximo = 11



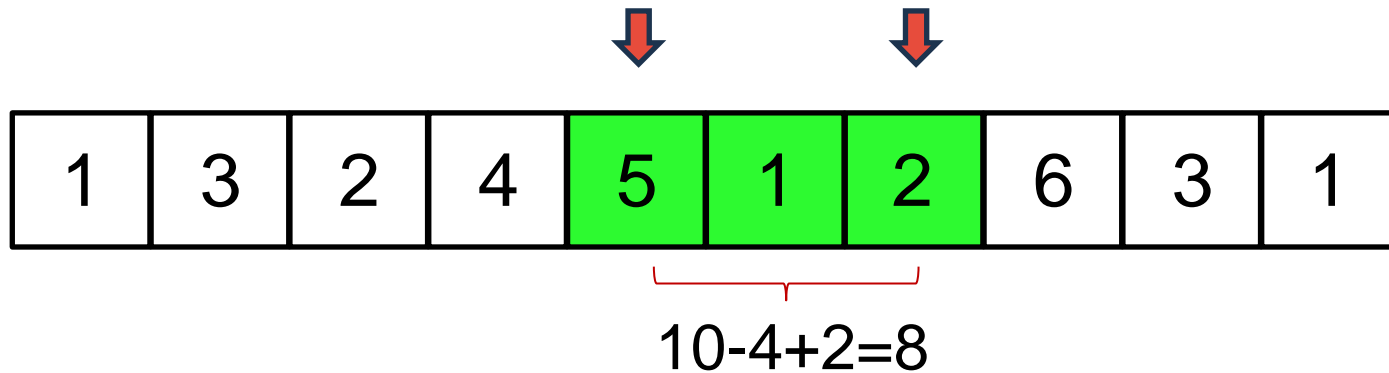
Sliding Window - Tamaño fijo



- $i = 2$
- $j = 4$
- Máximo = 11



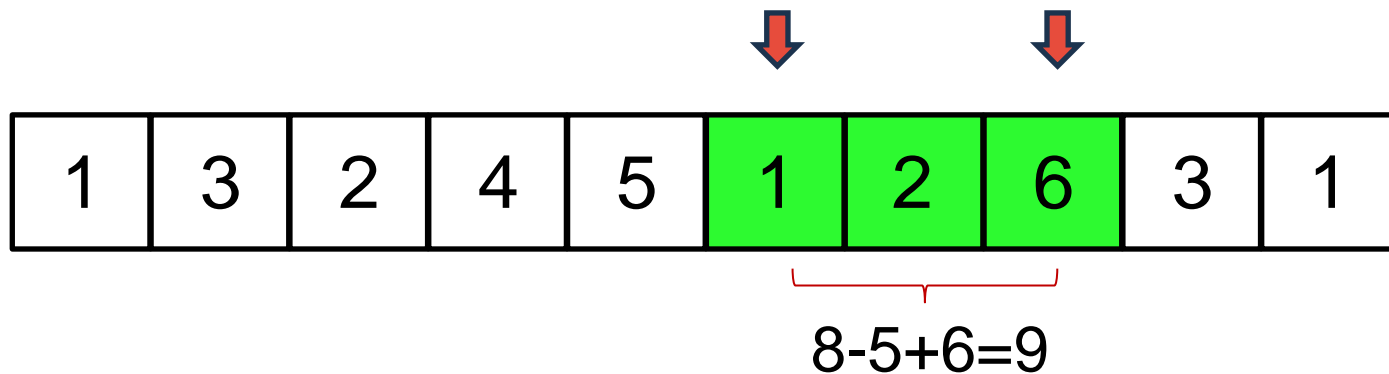
Sliding Window - Tamaño fijo



- $i = 4$
- $j = 6$
- Máximo = 11



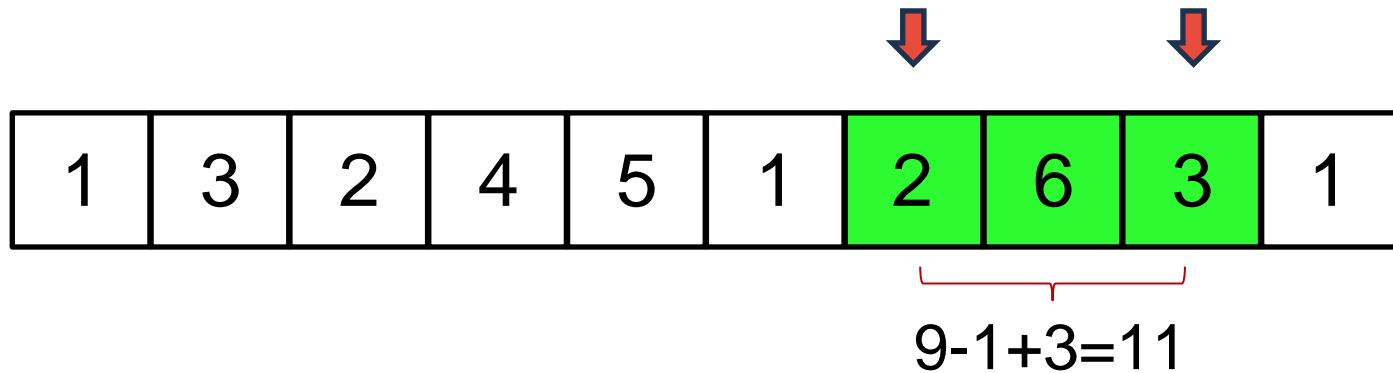
Sliding Window - Tamaño fijo



- $i = 5$
- $j = 7$
- Máximo = 11



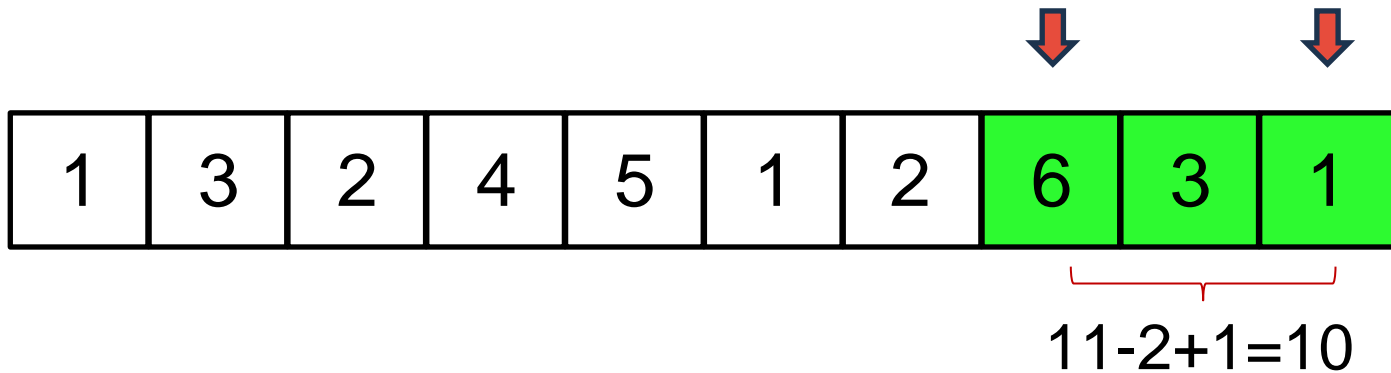
Sliding Window - Tamaño fijo



- $i = 6$
- $j = 8$
- Máximo = 11



Sliding Window - Tamaño fijo



- $i = 7$
- $j = 9$
- Máximo = 11



Sliding Window - Tamaño variable

- Lista de tamaño n
- Dos índices:
 - i ➡ inicio del subintervalo
 - j ➡ final del subintervalo
- Si el intervalo no cumple la condición:
 - Aumentamos su tamaño ➡ incrementamos j
- Si el intervalo supera la condición:
 - Reducimos su tamaño ➡ incrementamos i



Sliding Window - Tamaño variable



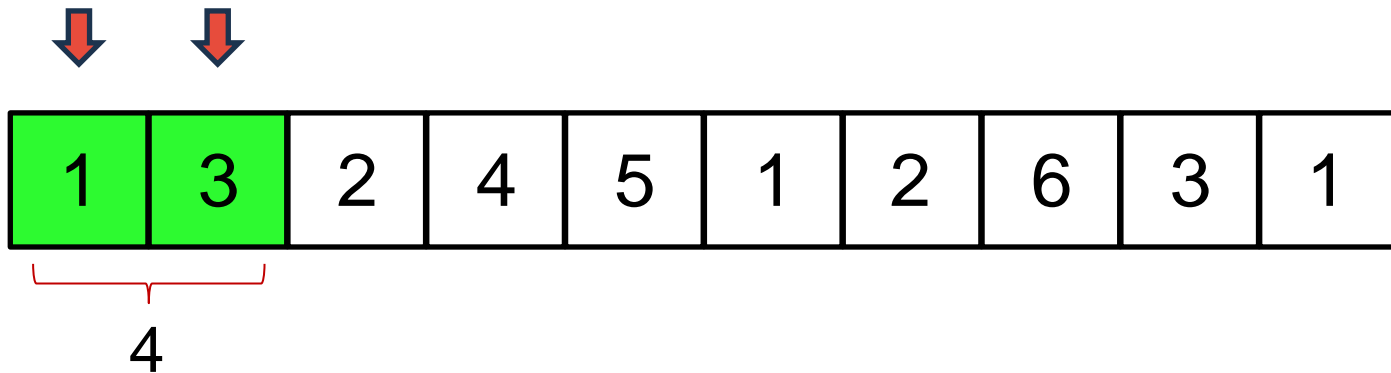
1	3	2	4	5	1	2	6	3	1
---	---	---	---	---	---	---	---	---	---

1

- $i = 0$
- $j = 0$
- $Veces = 0$



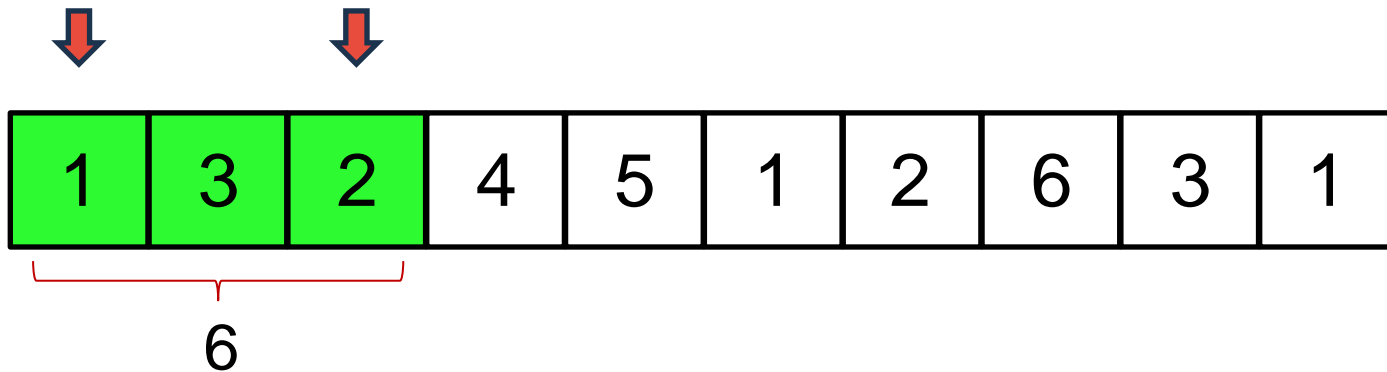
Sliding Window - Tamaño variable



- $i = 0$
- $j = 1$
- Veces = 0



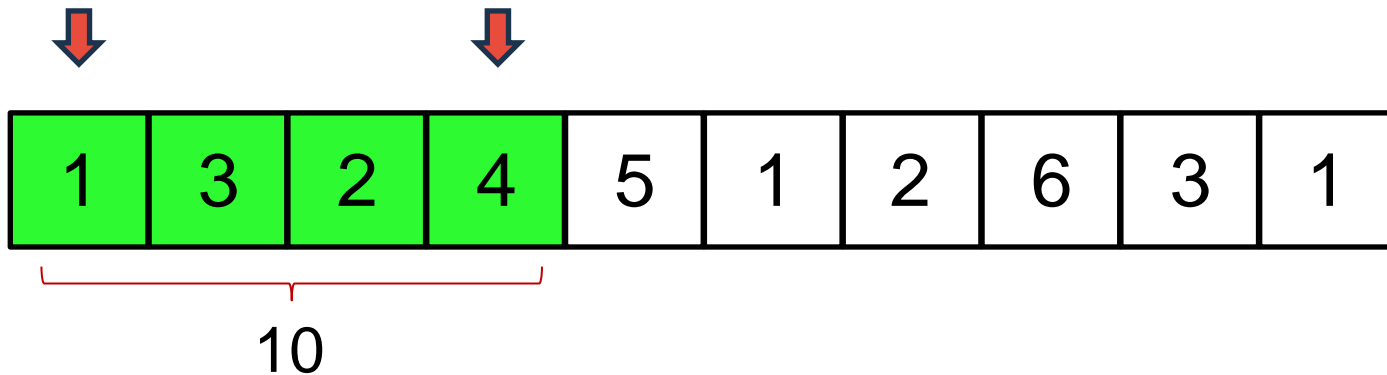
Sliding Window - Tamaño variable



- $i = 0$
- $j = 2$
- $\text{Veces} = 0$



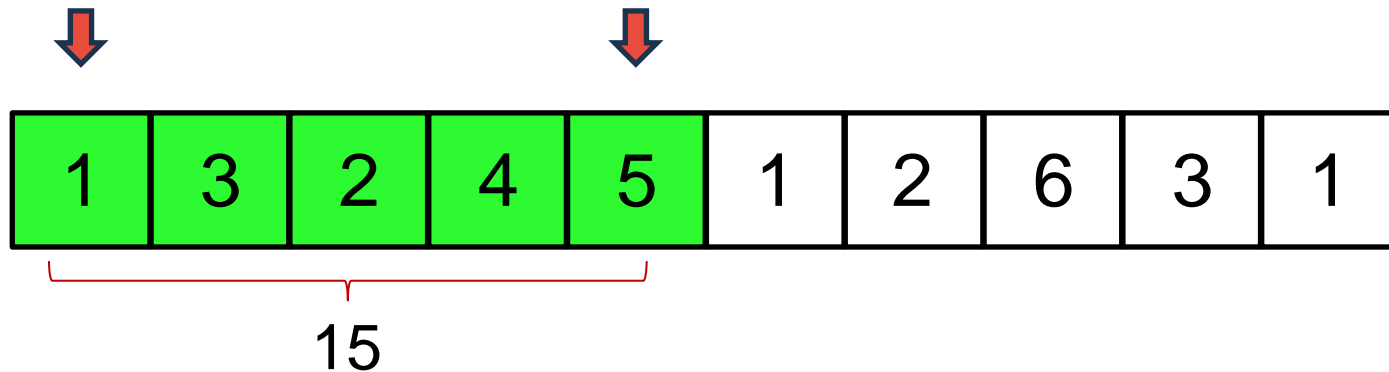
Sliding Window - Tamaño variable



- $i = 0$
- $j = 3$
- Veces = 1



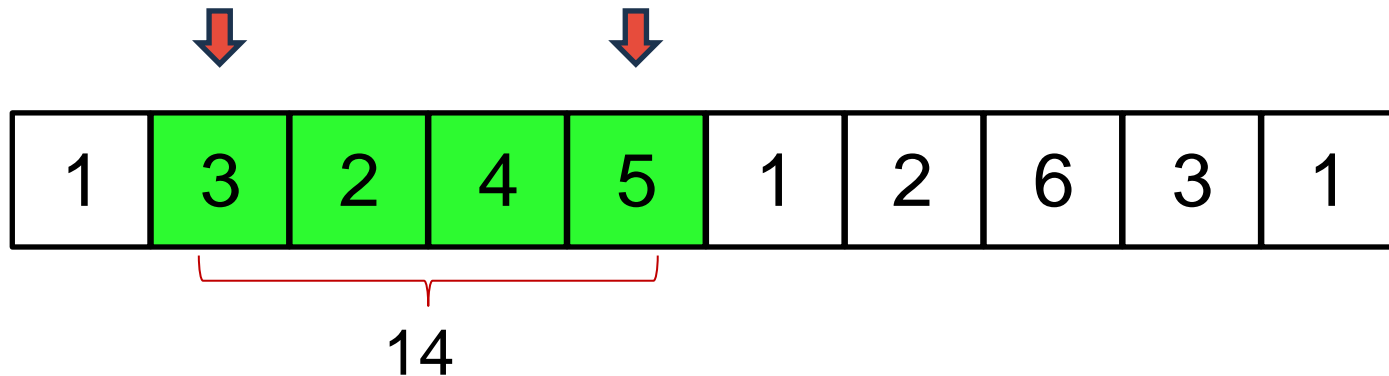
Sliding Window - Tamaño variable



- $i = 0$
- $j = 4$
- $\text{Veces} = 1$



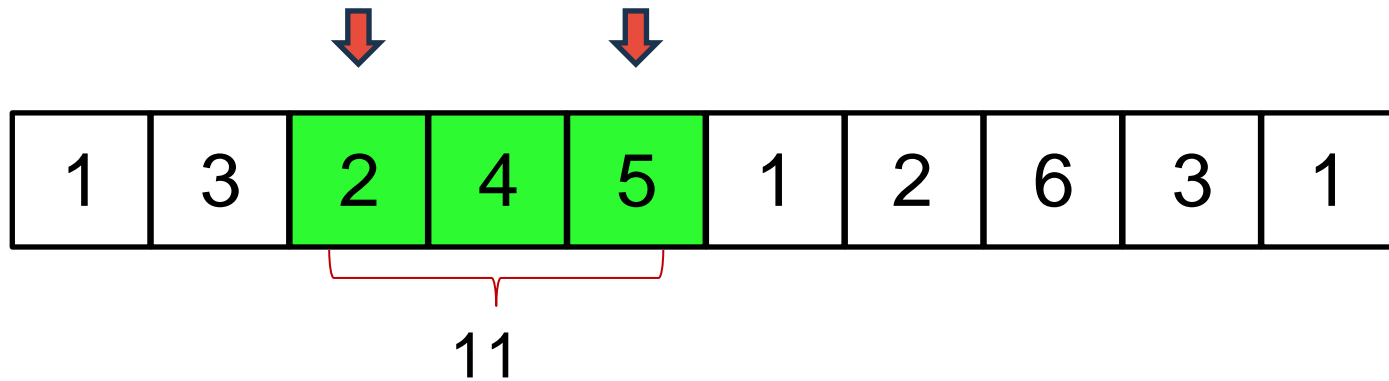
Sliding Window - Tamaño variable



- $i = 1$
- $j = 4$
- Veces = 1



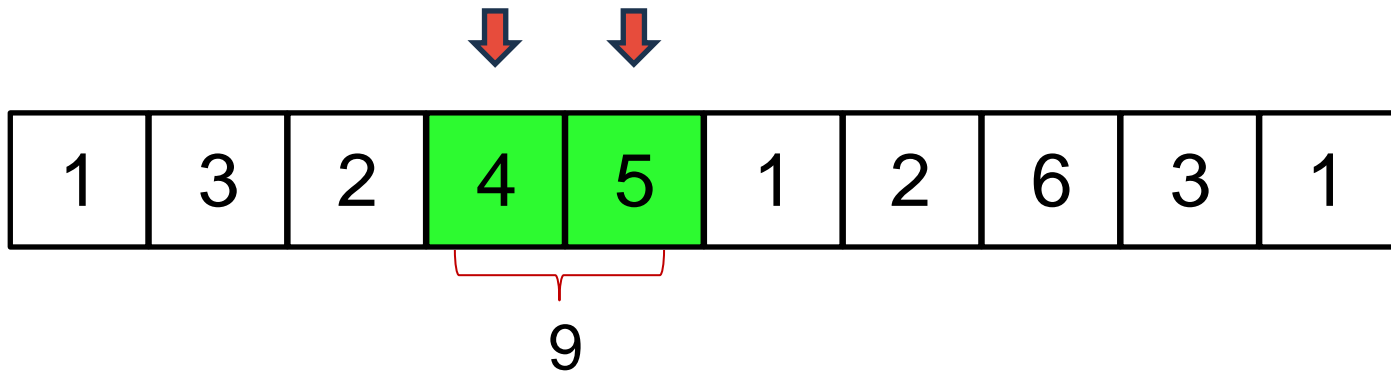
Sliding Window - Tamaño variable



- $i = 2$
- $j = 4$
- Veces = 1



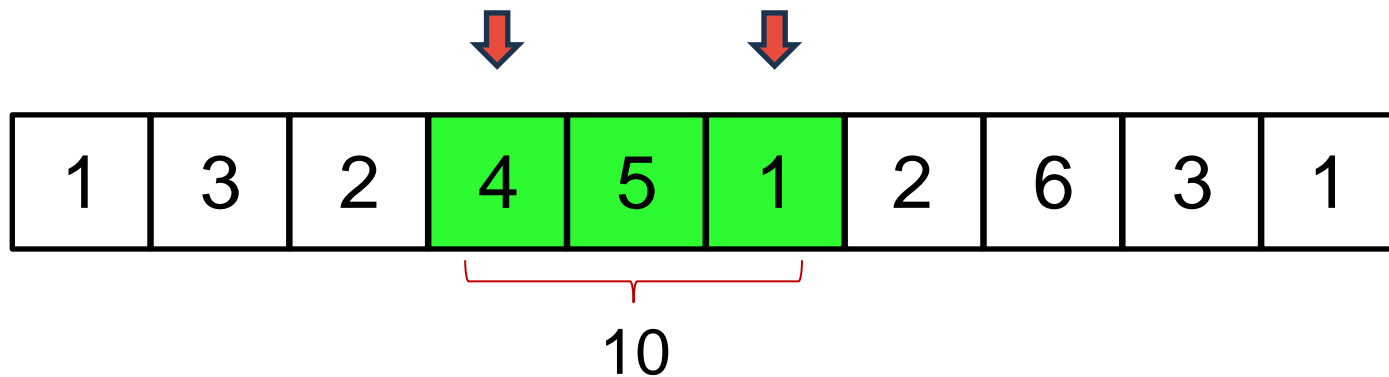
Sliding Window - Tamaño variable



- $i = 3$
- $j = 4$
- Veces = 1



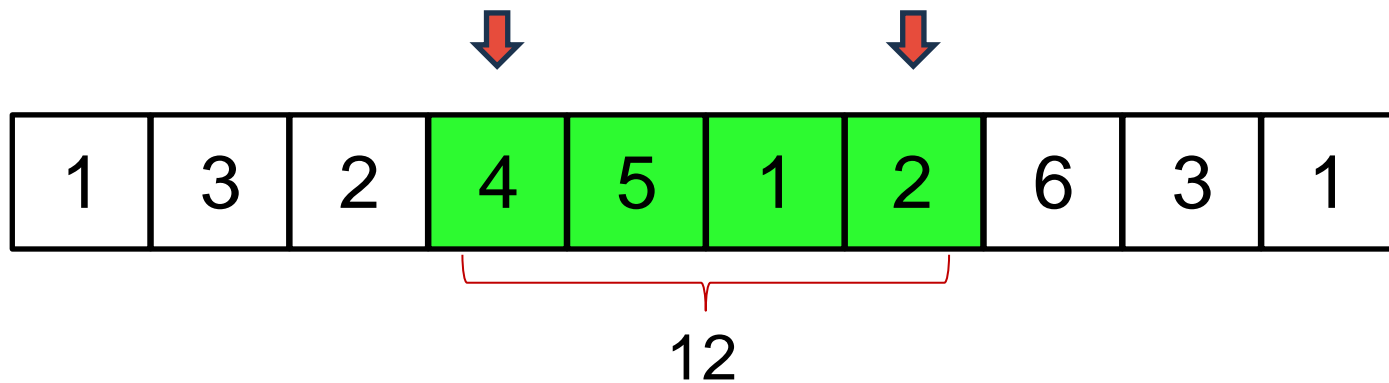
Sliding Window - Tamaño variable



- $i = 3$
- $j = 5$
- Veces = 2



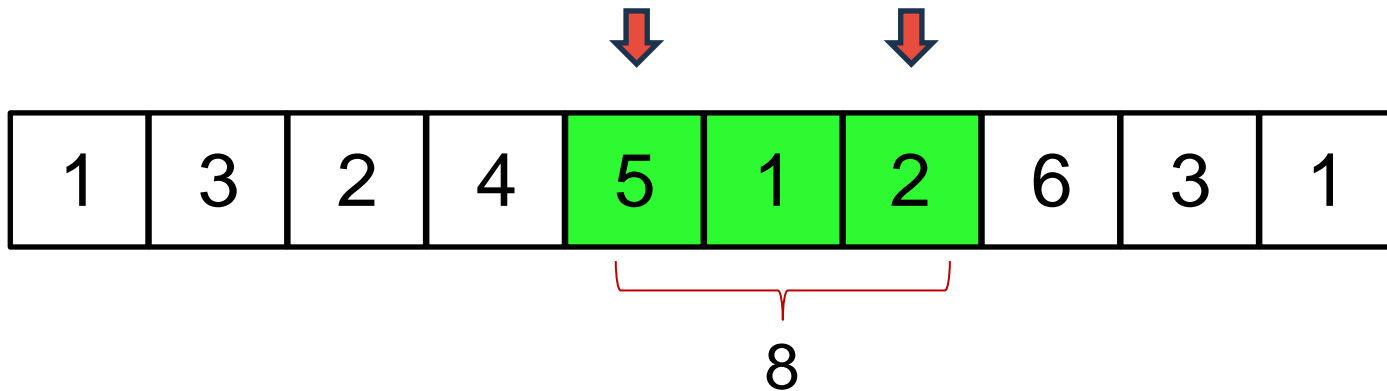
Sliding Window - Tamaño variable



- $i = 3$
- $j = 6$
- $\text{Veces} = 2$



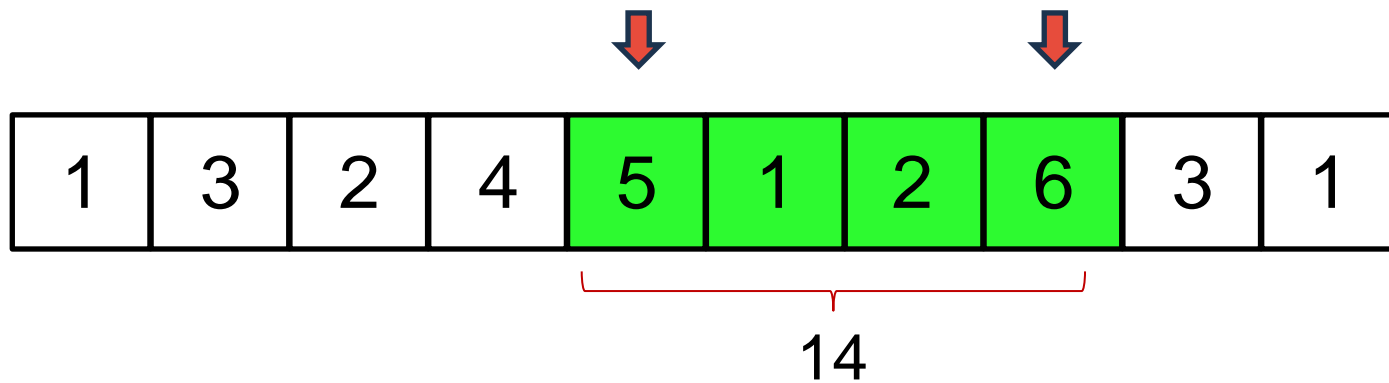
Sliding Window - Tamaño variable



- $i = 4$
- $j = 6$
- $\text{Veces} = 2$



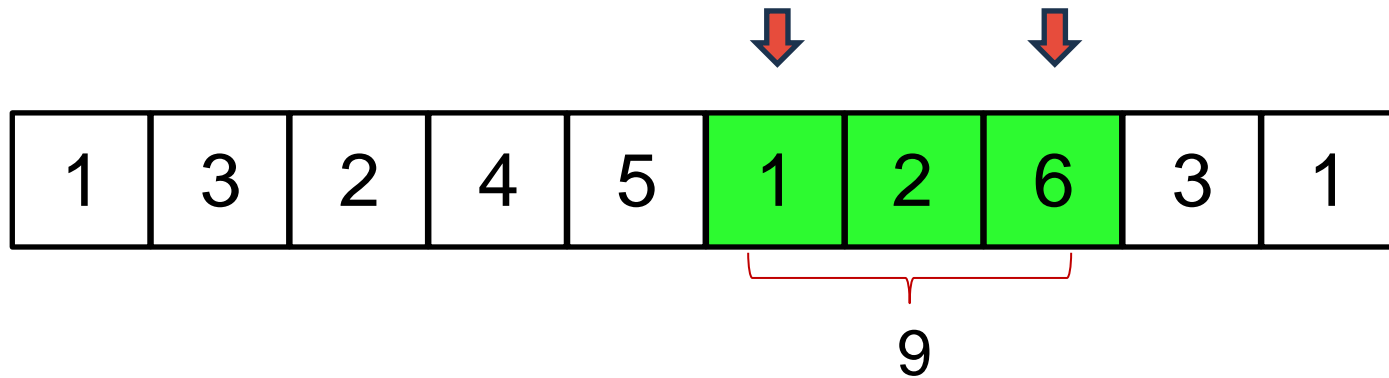
Sliding Window - Tamaño variable



- $i = 4$
- $j = 7$
- $\text{Veces} = 2$



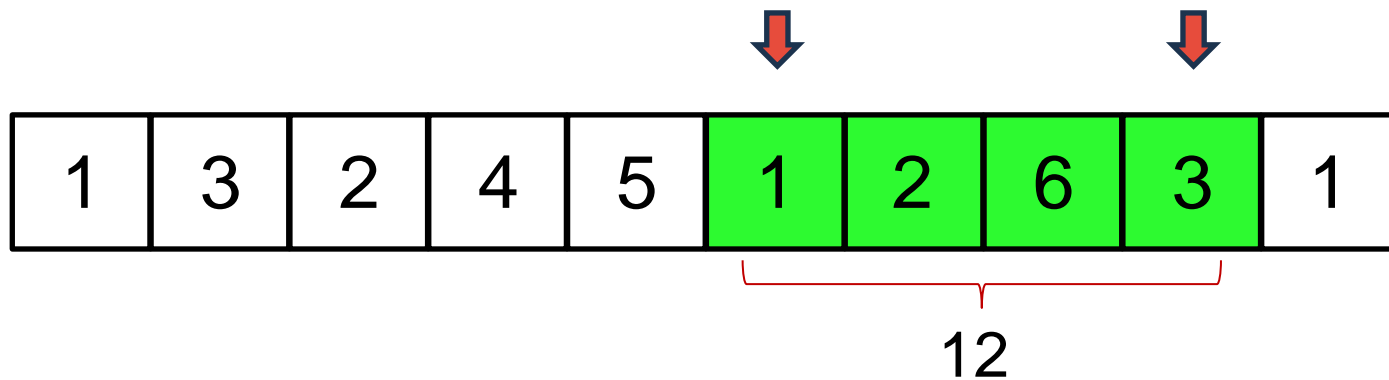
Sliding Window - Tamaño variable



- $i = 5$
- $j = 7$
- $\text{Veces} = 2$



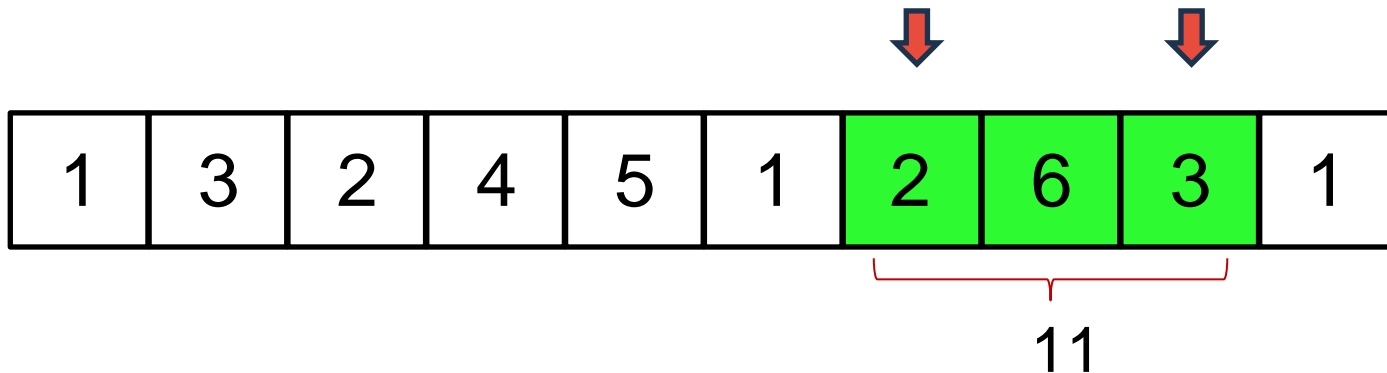
Sliding Window - Tamaño variable



- $i = 5$
- $j = 8$
- $\text{Veces} = 2$



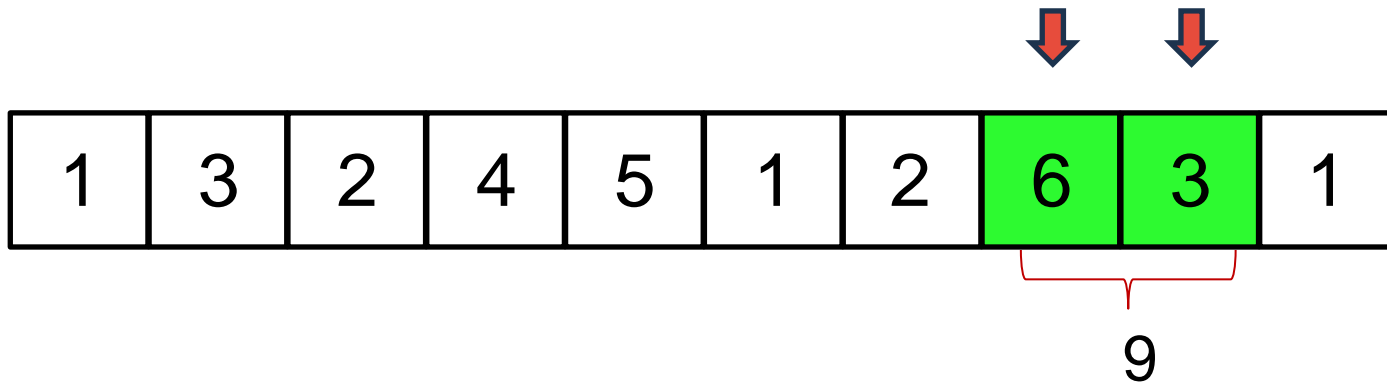
Sliding Window - Tamaño variable



- $i = 6$
- $j = 8$
- Veces = 2



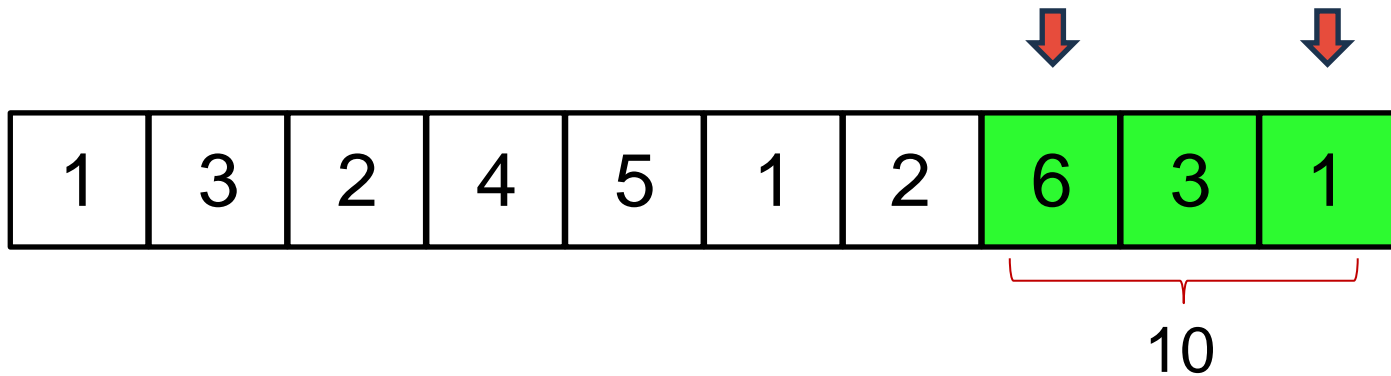
Sliding Window - Tamaño variable



- $i = 7$
- $j = 8$
- Veces = 2



Sliding Window - Tamaño variable



- $i = 7$
- $j = 9$
- $\text{Veces} = 3$



Sliding Window - Ejemplo

<https://open.kattis.com/problems/martiandna>

13 4 3

1 1 3 2 0 1 2 0 0 0 0 3 1

0 2

2 1

1 2

⇒ 7

1 1 3 2 0 1 2 0 0 0 0 3 1



¿PREGUNTAS?



PROBLEMAS
PROPUESTOS

[https://open.kattis.com/
contests/dos5ks](https://open.kattis.com/contests/dos5ks)



HASTA LA SEMANA QUE VIENE!



@URJC_CP



@Dijkstraidos

