
INSERTION SORT PASO A PASO (EJEMPLO CLASE 04 DE NOVIEMBRE 2021)

FACULTAD DE CIENCIAS-UNAM
YESSICA JANETH PABLO MARTÍNEZ

ALGORITMO INSERTION SORT

```
1 for ( i = 0; i < N - 1; i++ )  
2     for ( j = i + 1; j > 0 && arr[ j - 1 ] > arr[ j ]; j-- )  
3         swap ( array, j, j - 1 )
```

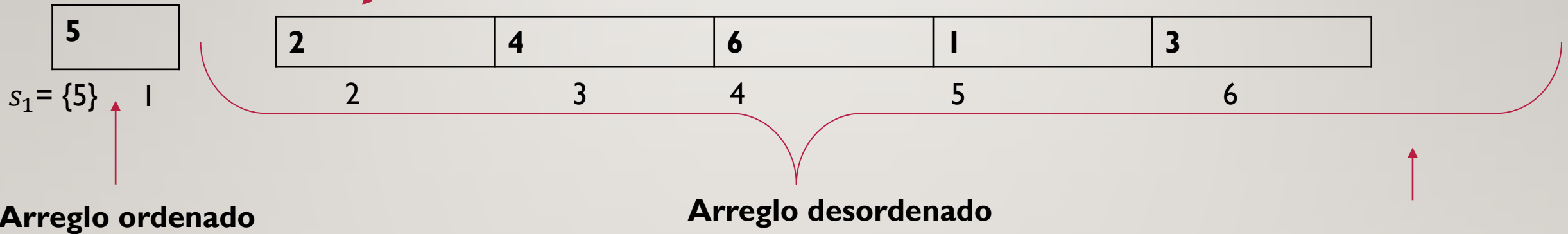
Ejemplo: ordena el siguiente arreglo usando InsertionSort

Tenemos:

S=	5	2	4	6	1	3
índices	1	2	3	4	5	6

Paso 1:

Siguiente elemento a insertar



* Verificamos que: **¿2 > 5? NO** entonces se intercambia . Luego **¿2 < 5? SÍ** , entonces hay intercambio

Paso 2:

Siguiente elemento a insertar

2	5
---	---

↑ 1 2

4	6	1	3
---	---	---	---

3

4

5

6

El elemento 2 se inserta
antes que el elemento 5

Arreglo desordenado

$$s_2 = \{5\},$$

$$s_3 = \{4\}$$

- Verificamos que: ¿5 < 4? **NO** luego ¿4 < 5? **SÍ**. Seguimos preguntando: ¿4 < 2? **NO**, por lo que insertamos a $s_3 = \{4\}$ en la
- posición 2 ¿Por qué? , por la verificación de ¿ $s_3 < s_2$? **SÍ**, es decir, ¿4 < 5?

Paso 3:

2	4	5
---	---	---

1

2

3



El elemento 4 se inserta
antes que el elemento 5

Siguiente elemento a insertar



6	1	3
---	---	---

4

5

6

Arreglo desordenado

$$s_3 = \{5\},$$

$$s_4 = \{6\}$$

- Verificamos que: ¿ $5 < 6$? **SÍ**, luego ¿ $6 < 5$? **NO**. Seguimos preguntando ¿ $6 < 4$? **NO**, ¿ $6 < 2$? **NO**, por lo que insertamos
- a $s_4 = \{6\}$ en la posición 4 **¿Por qué?**, por la verificación de ¿ $s_3 < s_4$? **SÍ**, es decir, ¿ $5 < 6$?

Paso 4:

2	4	5	6
---	---	---	---

1

2

3

4

$s_4 = \{6\}$, $s_5 = \{1\}$

El elemento 6 se inserta
después del elemento 5

Siguiente elemento a insertar

1	3
---	---

5

6

Arreglo desordenado

- Verificamos que: $6 < 1$? **NO**, luego $1 < 6$? **SÍ**, realizamos un cambio, $1 < 5$? **SÍ**, realizamos un cambio, $1 < 4$? **SÍ**,
- realizamos un cambio, $1 < 2$? **SÍ**, realizamos un cambio, por lo que insertamos a $s_5 = \{1\}$ en la
- posición 1 **¿Por qué?**, por la verificación de $s_5 < s_1$? **SÍ**, es decir, $1 < 2$

Paso 5:

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

1

2

3

4

5



El elemento 1 se inserta
en la primera posición

$$s_5 = \{6\}, s_5 = \{3\}$$

Siguiente elemento a insertar



3

6



Arreglo desordenado

- Verificamos que: ¿ $6 < 3$? **NO**, luego ¿ $3 < 6$? **SÍ**, realizamos un cambio, ¿ $3 < 5$? **SÍ**, realizamos un cambio, ¿ $3 < 4$? **SÍ**,
- realizamos un cambio, ¿ $3 < 2$? **NO**, por lo que insertamos a $s_6 = \{3\}$ en la
- posición 3 **¿Por qué?**, por la verificación de ¿ $s_6 < s_2$? , es decir, ¿ $3 < 2$?

Como resultado obtenemos al arreglo ordenado usando InsertionSort:

	1	2	3	4	5	6
índices	1	2	3	4	5	6

QUICKSORT PASO A PASO (EJEMPLO CLASE 04 DE NOVIEMBRE 2021)

FACULTAD DE CIENCIAS-UNAM
YESSICA JANETH PABLO MARTÍNEZ

ALGORITMO QUICKSORT

Con ayuda de nuestro método **partition** podemos obtener a nuestro algoritmo quickSort

```
1 partition ( arr [ ], lo, hi ) :  
2     i = lo  
3     j = hi + 1  
4     piv = arr [ lo ]  
5     while ( true ) :  
6         while ( arr [ ++i ] < piv ) if ( i == hi ) break  
7         while ( piv < arr [ --j ] ) if ( j == lo ) break  
8         if ( i ≥ j ) break  
9         swap ( arr, i, j )  
10    swap ( arr, lo, j )  
11    return j
```




```
1 quicksort ( arr [ ], lo, hi ) :  
2     if ( hi ≤ lo ) return  
3     j = partition ( a, lo, hi )  
4     quicksort ( arr, lo, j - 1 )  
5     quicksort ( arr, j + 1, hi )
```

Ejemplo: ordena el siguiente arreglo usando QuickSort

Tenemos:

S=	2	18	-3.2	6	-2.4	-8
índices	0	1	2	3	4	5

Paso estrella (*): mover a **up** al primer valor mayor que el pivote y movemos a **down** al primer valor de derecha a izquierda **menor** que el pivote. (Realizamos este paso en todos los demás pasos)

Paso I:	2	18	-3.2	6	-2.4	-8
	0	1	2	3	4	5
 primero	 pivote					 último

Paso 1:

2	18	-3.2	6	-2.4	-8
---	----	------	---	------	----

0

1

2

3

4

5

primero

pivote

último

up = 18, esto es debido al paso *, tomamos al primer valor de izquierda a derecha **mayor** que el pivote, por lo tanto **¿18 > 2?** **¡Si!**, por lo tanto **up = 18**

down = -8, esto es debido al paso *, tomamos al primer valor de derecha a izquierda **menor** que el pivote, por lo tanto **¿-8 < 2?** **¡Si!**, por lo tanto **down = -8**

2	18	-3.2	6	-2.4	-8
---	----	------	---	------	----

primero

pivote

up = 18
¿18 > 2? ¡Si!

último

down = -8
i-8 < 2? iSi!

Paso 1: como $18 > 2(\text{pivote})$ y $-8 < 2(\text{pivote})$ entonces intercambiamos los valores de up y down

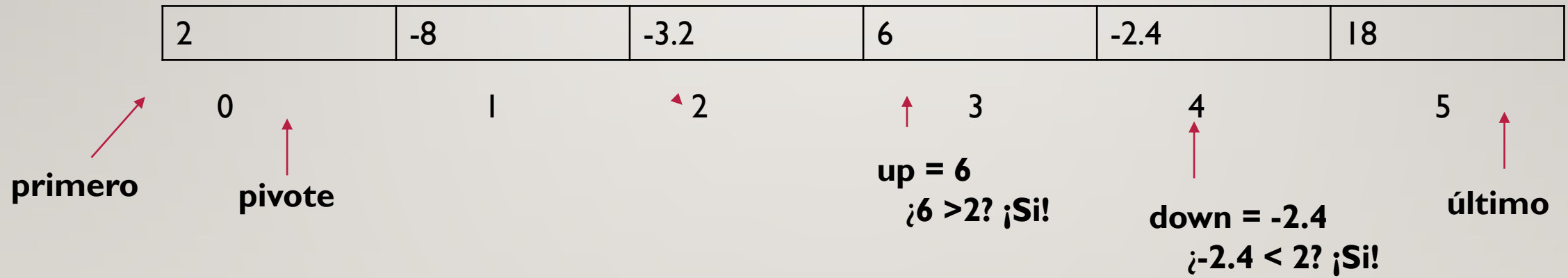
2	-8	-3.2	6	-2.4	18
---	----	------	---	------	----

primero

pivote

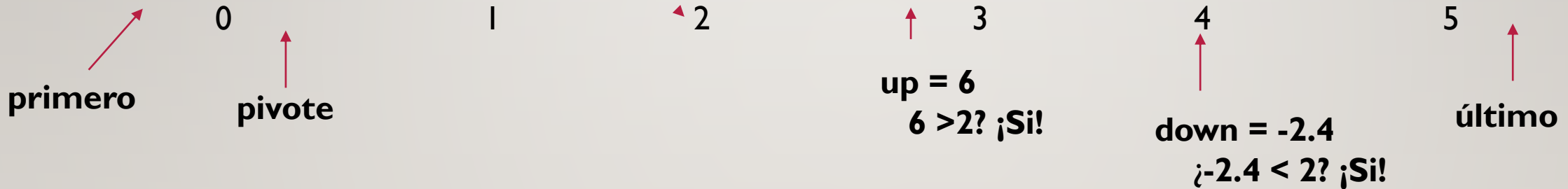
último

Paso 2: movemos a **up** hacia la derecha siguiendo la misma regla del **paso ***, es decir, nos movemos al primer valor **mayor** que el pivote. Luego movemos a **down** hacia la izquierda hasta el primer valor **menor** que el pivote. Por último intercambiamos.



Intercambiamos los valores de los índices 3 y 4

2	-8	-3.2	-2.4	6	18
---	----	------	------	---	----



Paso 2: movemos a **up** hacia la derecha siguiendo la misma regla del **paso ***, es decir, nos movemos al primer valor **mayor** que el pivote. Luego movemos a **down** hacia la izquierda hasta el primer valor **menor** que el pivote. Por último intercambiamos.

up toma el valor del índice 4, es decir, el valor de 6 pues $6 > 2$ (pivote) y **down** toma el valor de -2.4 pues $-2.4 < 2$ (pivote)

2	-8	-3.2	-2.4	6	18
---	----	------	------	---	----



Paso 3: Como **up** y **down** se cruzaron entonces realizamos un intercambio de valores con **down** y el **pivote**

Intercambiamos los valores de los índices 3 y 0

-2.4	-8	-3.2	2	6	18
------	----	------	---	---	----

0

1

2

3

4

5

primero

último

pivIndex

Primero'

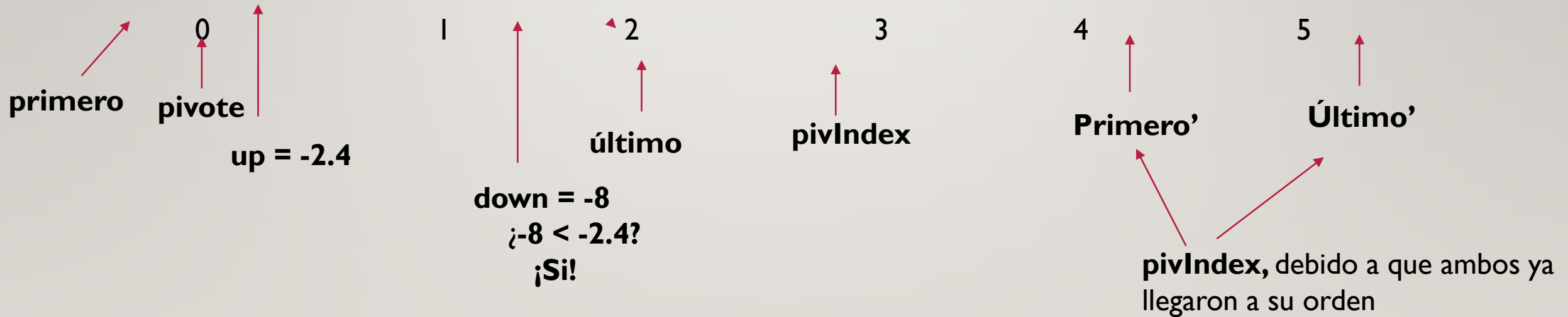
Último'

Ojo: notemos que a partir de nuestro **pivIndex** tendremos de lado izquierdo a valores menores que el y de lado derecho valores mayores a el , por lo tanto **pivIndex** representará al elemento fijo que llegó a su orden.

- .- Declaramos a las nuevas variables **Primero'** y **Último'** para el nuevo subarreglo a ordenar .
- .- Por último volvemos a realizar los pasos para ordenar

Intercambiamos los valores de los índices 3 y 0

-2.4	-8	-3.2	2	6	18
------	----	------	---	---	----



Por lo tanto realizamos un intercambio

Intercambiamos los valores de los índices 1 y 0

-8	-2.4	-3.2	2	6	18
----	------	------	---	---	----

0

pivIndex
Ya llegó a su
orden

1

**primero
pivote**

up = -2.4

2

último

3

pivIndex

4

pivIndex

5

pivIndex







down = -3.2

¿-3.2 < -2.4?

¡Si!

Por lo tanto realizamos un intercambio

Intercambiamos los valores de los índices 1 y 2

-8	-3.2	-2.4	2	6	18
0	1	2	3	4	5
					
pivIndex	pivIndex	pivIndex	pivIndex	pivIndex	pivIndex

¡Por lo tanto el arreglo queda ordenado mediante QuickSort !

SELECTIONSORT PASO A PASO (EJEMPLO CLASE 08 DE NOVIEMBRE 2021)

FACULTAD DE CIENCIAS-UNAM
YESSICA JANETH PABLO MARTÍNEZ

ALGORITMO SELECTIONSORT

```
1 for ( i = N - 1; i > 0; i-- )
2     max = 0
3     for ( j = 1; j ≤ i; j++ )
4         if ( array[ j ] > array[ max ] )
5             max = j
6     swap ( array, max, i )
```

Ejemplo: ordena el siguiente arreglo usando selectionSort

Tenemos:

S=	3	-0.2	5	-2	-1.5	4
índices	1	2	3	4	5	6

Paso 1:

Se intercambia con el elemento mínimo

3	-0.2	5	-2	-1.5	4
1	2	3	4	5	6

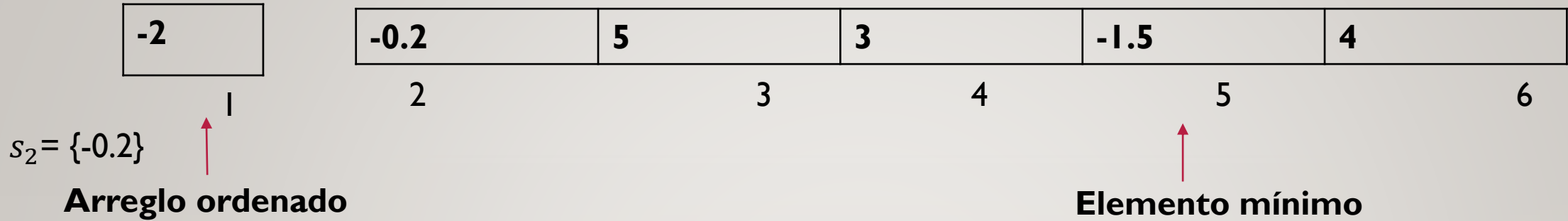
$s_1 = \{3\}$

Elemento mínimo

Paso * : Lo primero que hacemos es posicionarnos en el primer elemento del arreglo y lo recorremos hasta encontrar al elemento mínimo. Si lo encontramos intercambiamos al elemento mínimo con el de la posición $i, i = 1, 2, 3, 4, 5, 6$

Mínimo = -2, por lo tanto intercambio

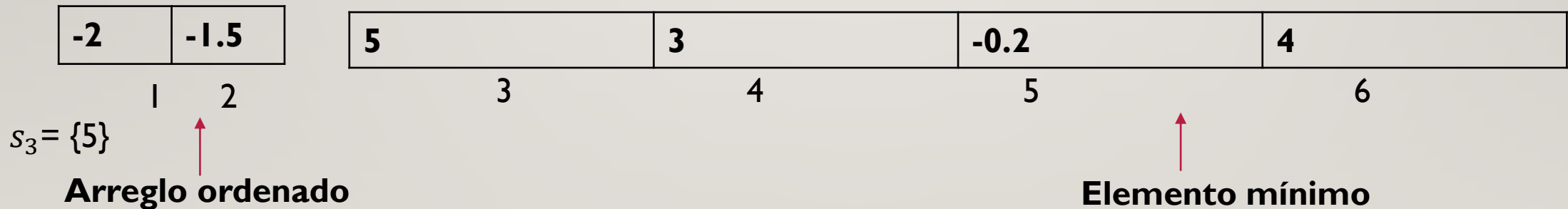
Se intercambia con el elemento mínimo



Paso 2: realizamos el **paso *** y si encontramos al elemento mínimo lo intercambio con mi s_2

Mínimo = -1.5, por lo tanto intercambio

Se intercambia con el elemento mínimo



Paso 3: realizamos el **paso *** y si encontramos al elemento mínimo lo intercambiamos con mi s_3

Mínimo = 3, por lo tanto intercambio

-2	-1.5	-0.2
----	------	------

4

5

6

$s_4 = \{6\}$

Arreglo ordenado

Se intercambia con el elemento mínimo

6	1	3
---	---	---

4

5

6

Elemento mínimo

Paso 4: realizamos el **paso *** y si encontramos al elemento mínimo lo intercambiamos con mi s_4

Mínimo = 1, por lo tanto intercambio

-2	-1.5	-0.2	1
----	------	------	---

1

2

3

4

$s_5 = \{6\}$

Arreglo ordenado

Se intercambia con el elemento mínimo

6	3
---	---

5

6

Elemento mínimo

Paso 5: realizamos el **paso *** y si encontramos al elemento mínimo lo intercambio con mi s_5 .

Mínimo = -0.2, por lo tanto intercambio

-2	-1.5	-0.2	1	3
1	2	3	4	5

$s_6 = \{6\}$
Arreglo ordenado

6
6

Elemento mínimo

Paso 6: realizamos el **paso *** y si encontramos al elemento mínimo lo intercambio con mi s_6 . Como el elemento mínimo de Todo el arreglo es 6 (el mismo) entonces no se hace ningún intercambio y se agrega al arreglo ordenado.

Paso Final: obtenemos a nuestro arreglo ordenado con selectionSort

-2	-1.5	-0.2	1	3	6
1	2	3	4	5	6

MERGESORT PASO A PASO (EJEMPLO CLASE II DE NOVIEMBRE 2021)

FACULTAD DE CIENCIAS-UNAM
YESSICA JANETH PABLO MARTÍNEZ

ALGORITMO MERGESORT

```
1 mergesort ( arr [ ] ) :  
2   mergesort ( arr, 0, N - 1 )
```

```
1 merge ( arr [ ], lo, mid, hi ) :  
2   i = lo  
3   j = mid + 1  
4   aux = arr.copia()  
5   for ( k = lo; k ≤ hi; k++ ) :  
6       if ( i > mid )                a[ k ] = aux[ j++ ]  
7       else if ( j > hi )            a[ k ] = aux[ i++ ]  
8       else if ( aux[ j ] < aux[ i ] ) a[ k ] = aux[ j++ ]  
9       else                          a[ k ] = aux[ i++ ]
```


Ejemplo: ordena el siguiente arreglo usando mergeSort

Tenemos:

S=	3	-0.2	5	-2	-1.5	4
índices	0	1	2	3	4	5

Paso 1: dividimos por la mitad ayudándonos de **mitad** $\lfloor (0+5)/2 \rfloor = 2$

3	-0.2	5
0	1	2

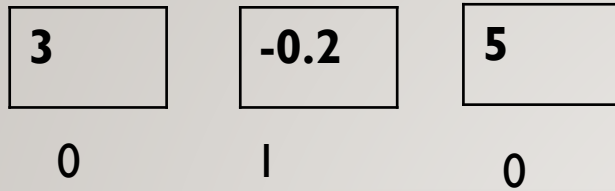
-2	-1.5	4
0	1	2

Paso 2: dividimos por la mitad ayudándonos de **mitad** $\lfloor (0+2)/2 \rfloor = 1$

3	-0.2	5
0	1	0

-2	-1.5	4
0	1	0

Paso 3: dividimos por la mitad ayudándonos de **mitad** $\lfloor (0+1)/2 \rfloor = 0$

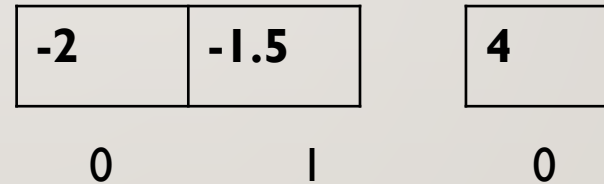
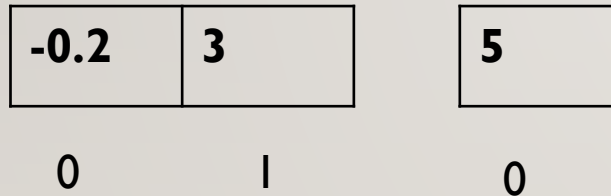


Paso 4: ahora comparamos a los elementos del arreglo, es decir,

¿3 > -0.2? Sí, entonces hay intercambio

¿-2 > -1.5? No, entonces no hay intercambio

A 5 y a 4 no los comparamos aún



Paso 5: mezclamos los elementos y volvemos a comparar los elementos del arreglo, preguntamos:

¿5 < -0.2? No, entonces no hay intercambio

¿5 < 3? No, entonces no hay intercambio

¿4 < -2? No, entonces no hay intercambio

¿4 < -1.5? No, entonces no hay intercambio

Por lo tanto solo mezclamos

-0.2	3	5
-------------	----------	----------

0

1

2

-2	-1.5	4
-----------	-------------	----------

0

1

2

Paso 5: mezclamos los elementos y volvemos a comparar los elementos del arreglo, preguntamos:

¿-2 < 5? Sí, entonces hay intercambio

¿-2 < 3? Sí, entonces hay intercambio

¿-2 < -0.2? Sí, entonces hay intercambio

¿-1.5 < 5? Sí, entonces hay intercambio

¿ $-1.5 < 3$? Sí, entonces hay intercambio
¿ $-1.5 < -0.2$? Sí, entonces hay intercambio

¿ $4 < 5$? Sí, entonces hay intercambio
¿ $4 < 3$? No, entonces no hay intercambio

Por lo tanto solo mezclamos y obtenemos al arreglo ordenado usando mergeSort

S=	-2	-1.5	-0.2	3	4	5
índices						

BÚSQUEDA BINARIA PASO A PASO

FACULTAD DE CIENCIAS-UNAM
YESSICA JANETH PABLO MARTÍNEZ

ALGORITMO PARA BÚSQUEDA BINARIA

```
class BinarySearch {  
    int binarySearch(int arr[], int lo, int hi, int x) {  
        if (hi >= lo && lo < arr.length - 1) {  
            int mid = lo + (hi - lo) / 2;  
            if (arr[mid] == x)  
                return mid;  
            if (arr[mid] > x)  
                return binarySearch(arr, lo, mid - 1, x);  
            return binarySearch(arr, mid + 1, hi, x);  
        }  
        return -1;  
    }  
}
```

Ejemplo: sea el siguiente arreglo S ordenado mediante el uso del ordenamiento mergeSort, busca al elemento **z = -2**

S=	-2	-1.5	-0.2	3	4	5
índices	0	1	2	3	4	5

Paso I: nos basamos en el algoritmo y dividimos por la mitad ayudándonos de **mid**, donde $lo = 0$ y $hi = 5$ por lo tanto **mid** = $\lfloor (0+5)/2 \rfloor = 2$.

Verificamos:

$\text{array}[2] = -0.2 = -2?$, No

$\text{array}[2] = -0.2 < -2?$, No

Por lo tanto $hi = mid - 1$

$$\text{mid} = \lfloor (0+1)/2 \rfloor = 0$$

Verificamos:

¿array[0] = -2 = -2?, sí, por lo tanto devolvemos ¡éxito!

-2	-1.5	-0.2	3	4	5
----	------	------	---	---	---

0 1 2 3 4 5

éxito