



Fundamentos de Programación 2

Ing. Marco Aedo López

Arreglos (Arrays): Búsquedas y Ordenamientos



Capítulo 1



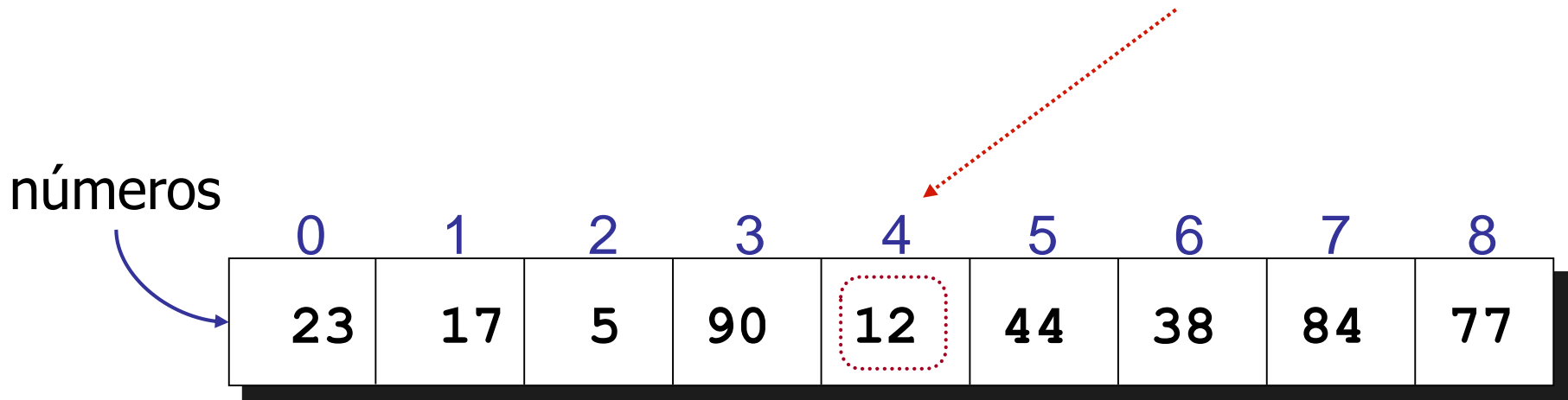
14. Buscando en un arreglo

- Es frecuente la necesidad de determinar si un arreglo contiene un valor en particular
- Cuando mantenemos una colección de datos, una de las operaciones que necesitaremos es un método de búsqueda para localizar un elemento
- Este es el planteamiento del problema:

Dado un valor **x**, devolver el índice de **x** en el arreglo, si tal **x** existe. Caso contrario, retornar -1 (NO_ENCONTRADO). Asumimos que no existen elementos duplicados en el arreglo

14. Buscando en un arreglo

- Primera aproximación: Búsqueda Lineal (de inicio a fin)



Búsqueda fallida: `búsqueda (45)` \longrightarrow **-1 (NO_ENCONTRADO)**

Búsqueda exitosa: `búsqueda (12)` \longrightarrow **4**



14. Buscando en un arreglo

- Pseudocódigo Búsqueda Lineal:

```
busquedaLineal(lista[], valor)
    para i desde 0 hasta lista.longitud-1 inc 1
        si lista[i]==valor
            retornar i
        finSi
    finCiclo
    retornar -1
FIN
```

Ejercicio 1: Implementar el método en Java



Consideración de Diseño: Cohesión

- Note que el método `busquedaLineal` no muestra el valor encontrado en el arreglo. Tampoco muestra un mensaje si es que se encontró o no
- En vez de eso, devuelve la posición en donde se encontró el elemento ó -1 si es que no fue encontrado
- Esto sigue el principio de **cohesión** - un método sólo debe realizar una sola tarea. En este caso, debe dejarse decidir, a quién llamo al método, si algún mensaje debe ser mostrado



14. Buscando en un arreglo

■ Problema:

- Crear un programa que indique si se encontró o no un `id` en un arreglo de `ids`, también que muestre en qué posición logró encontrarlo
- Escribir un método `encontrarEstudiante` que busca un `id` dentro de un arreglo de `ids`
- El método `encontrarEstudiante` debería recibir el arreglo `estudiantesIds` y un `id` como parámetros y retornar el valor del índice donde encuentre el `id` dentro del arreglo
- Si el `id` no se encuentra, retornar `-1`



14. Buscando en un arreglo

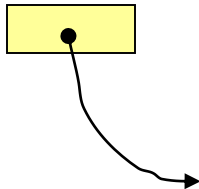
- Problema:
 - ¿En un arreglo ordenado?

14. Buscando en un arreglo

- Problema:

- ¿Y si el arreglo está ordenado?
- Aplicamos la búsqueda binaria

números



0	1	2	3	4	5	6	7	8
5	12	17	23	38	45	77	84	90

- Primero buscamos el valor en la posición media del arreglo. Si coincide con el elemento buscado **x**, se FINALIZA. Si el valor buscado **x** es menor al valor en la posición media, se busca en la mitad izquierda del arreglo. Si el valor buscado **x** es mayor al valor en la posición media, se busca en la mitad derecha del arreglo. Y ASI SE CONTINUA. -1 si no se encuentra

Secuencia de búsqueda exitosa (1/3)

#1 baja alta media

 0 8 4

búsqueda(45)

$$media = \left\lfloor \frac{baja + alta}{2} \right\rfloor$$

0	1	2	3	4	5	6	7	8
5	12	17	23	38	45	77	84	90

↑
baja

↑
media

↑
alta

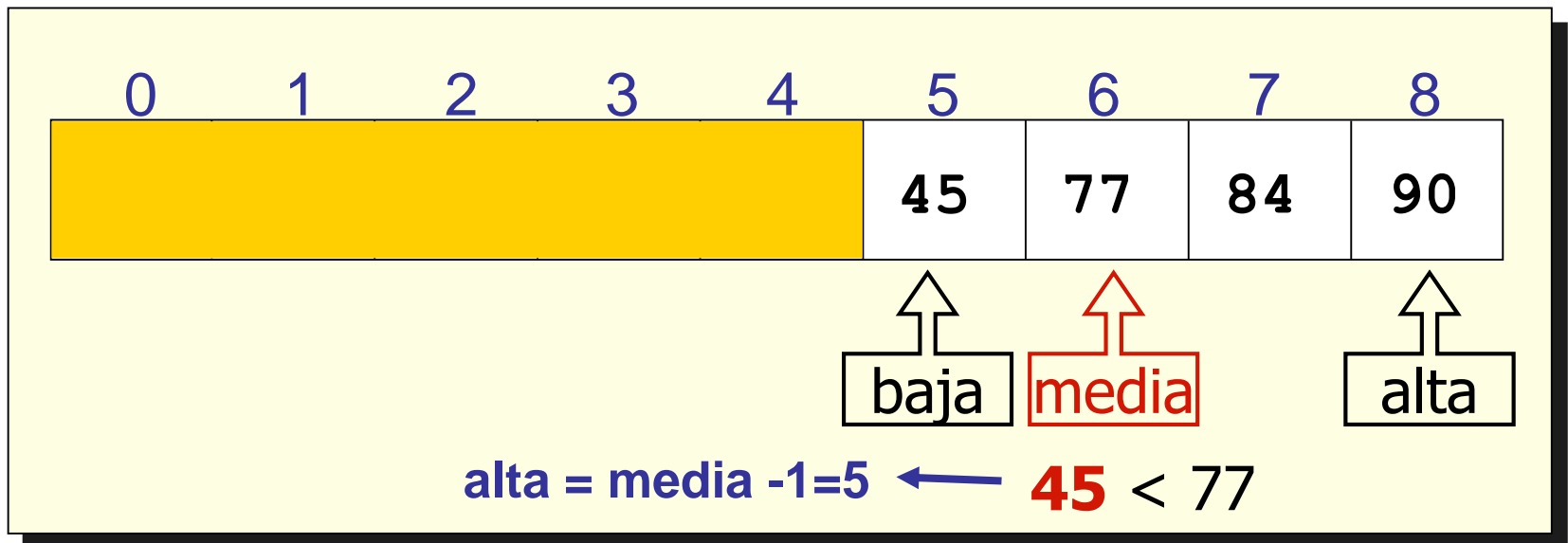
45 > 38 → baja = media+1 = 5

Secuencia de búsqueda exitosa(2/3)

	<u>baja</u>	<u>alta</u>	<u>media</u>
#1	0	8	4
#2	5	8	6

búsqueda(45)

$$media = \left\lfloor \frac{baja + alta}{2} \right\rfloor$$



Secuencia de búsqueda exitosa(3/3)

	baja	alta	media
#1	0	8	4
#2	5	8	6
#3	5	5	5

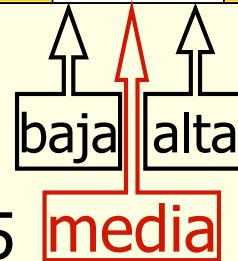
búsqueda(45)

$$media = \left\lfloor \frac{baja + alta}{2} \right\rfloor$$

0 1 2 3 4 5 6 7 8



Búsqueda exitosa!!



45 == 45

$$media = \left[\frac{baja + alta}{2} \right]$$

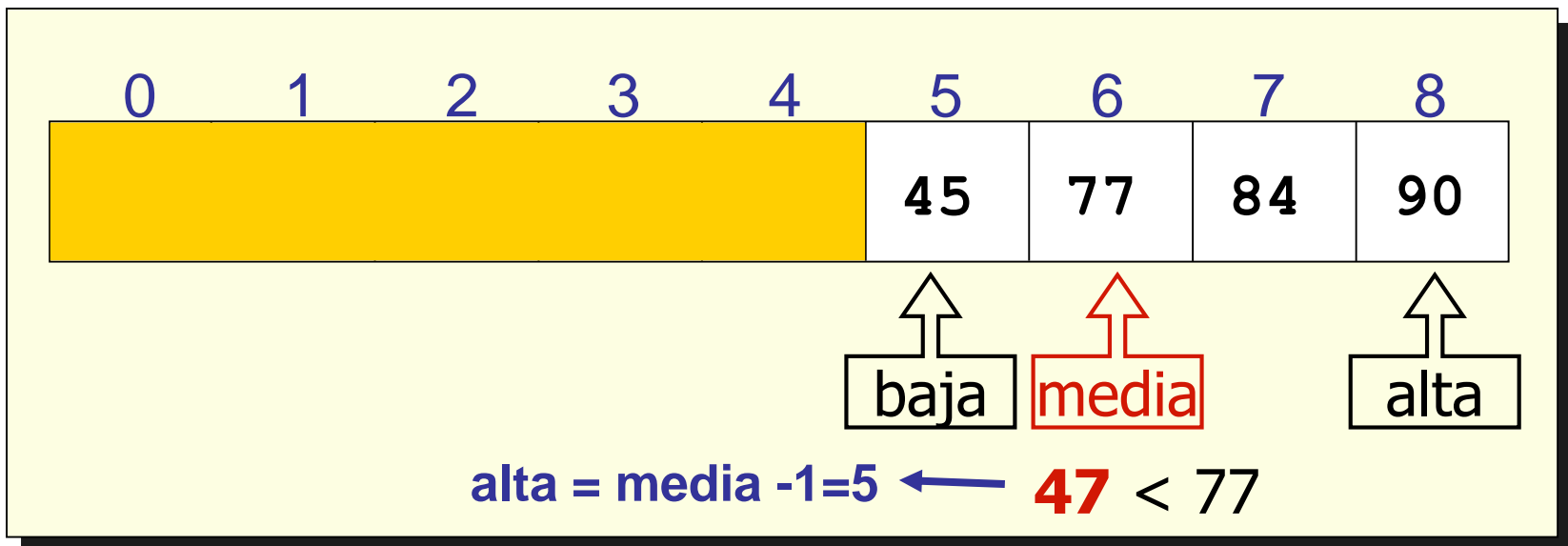
47 > 38 \rightarrow **baja = media+1 = 5**

Secuencia de búsqueda fallida(2/4)

	<u>baja</u>	<u>alta</u>	<u>media</u>
#1	0	8	4
#2	5	8	6

búsqueda(47)

$$media = \left\lfloor \frac{baja + alta}{2} \right\rfloor$$

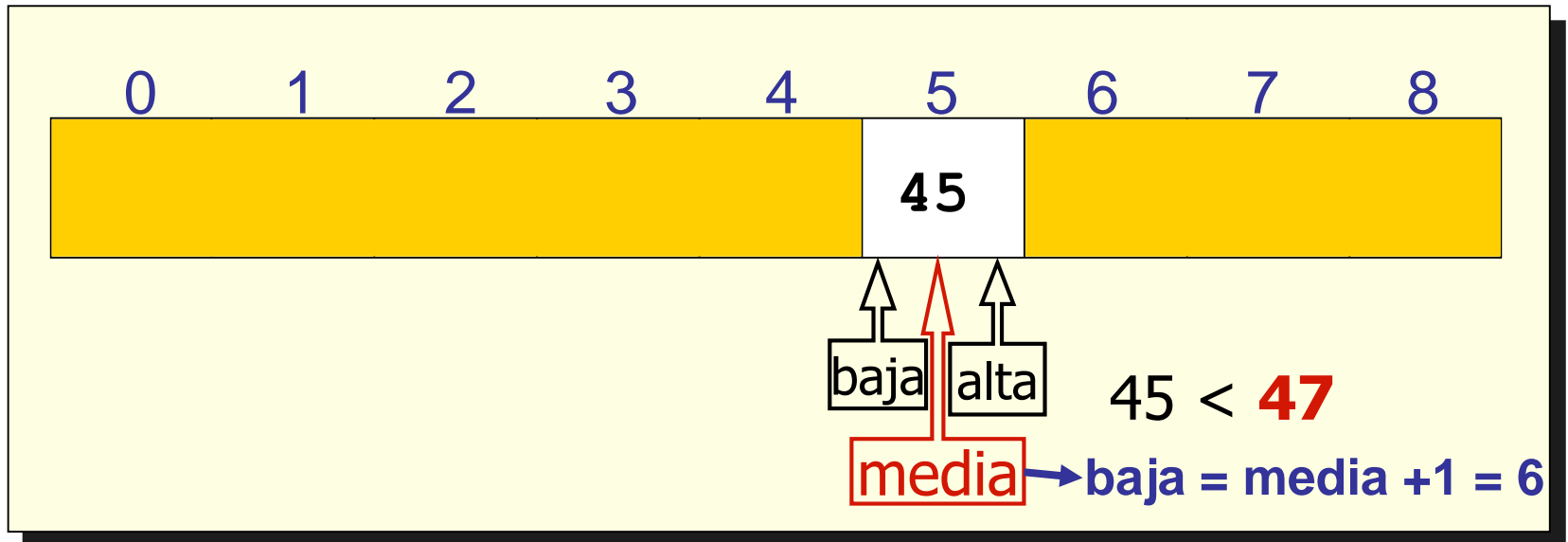


Secuencia de búsqueda fallida(3/4)

	baja	alta	media
#1	0	8	4
#2	5	8	6
#3	5	5	5

búsqueda(47)

$$media = \left\lfloor \frac{baja + alta}{2} \right\rfloor$$



Secuencia de búsqueda fallida(4/4)

	baja	alta	media
#1	0	8	4
#2	5	8	6
#3	5	5	5
#4	6	5	

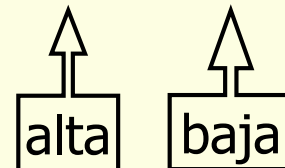
búsqueda(47)

$$media = \left\lfloor \frac{baja + alta}{2} \right\rfloor$$

0 1 2 3 4 5 6 7 8



Búsqueda fallida



no hay mas elementos a buscar ← baja > alta



14. Buscando en un arreglo

```
busquedaBinaria(lista[], valor)
    alta,baja,media:ENTERO
    baja=0
    alta=lista.longitud-1
    mientras (baja<=alta)
        media=(alta+baja)/2
        si lista[media]==valor
            retornar media
        sino si valor<lista[media]
            alta=media-1
        sino
            baja=media+1
        finSi
    finSi
finCiclo
retornar -1
```

FIN

Ejercicio 2:
Implementar el
algoritmo en Java



15. Ordenando un arreglo

- El ordenamiento es una tarea común en la computación
- Cuando tenemos una colección de datos, muchas aplicaciones necesitan ordenar los datos en cierto orden. Por ejemplo ordenar la información de una persona en forma ascendente por edad
- Ejemplos:
 - Ordenar mails en su correo por fecha, por remitente
 - Ordenar canciones por título, por autor
 - Ordenar alumnos por su CUI, por su promedio



15. Ordenando un arreglo

- Aquí vemos la definición del problema:

Dado un arreglo de N valores enteros,
ordenar sus valores en orden ascendente.

23	17	5	90	12	44	38	84	77
----	----	---	----	----	----	----	----	----

5								
---	--	--	--	--	--	--	--	--



15. Ordenando un arreglo: Burbuja

- Realizar pasadas, en cada pasada se garantiza que el más pesado se hunda al fondo
- Se comparan cada dos elementos, intercambiando cuando el primero es mayor que el segundo

Ordenamiento Burbuja, 1ra pasada

0	1	2	3	4	5	6	7	8
23	17	5	90	12	44	38	84	77

↑ ↑ intercambia

17	23	5	90	12	44	38	84	77
----	----	---	----	----	----	----	----	----

↑ ↑ intercambia

17	5	23	90	12	44	38	84	77
----	---	----	----	----	----	----	----	----

↑ ↑ ok
↑ ↑ intercambia

17	5	23	12	90	44	38	84	77
----	---	----	----	----	----	----	----	----

↑ ↑ intercambia

17	5	23	12	44	90	38	84	77
----	---	----	----	----	----	----	----	----

intercambia ↑ ↑

17	5	23	12	44	38	90	84	77
----	---	----	----	----	----	----	----	----

intercambia ↑ ↑

17	5	23	12	44	38	84	90	77
----	---	----	----	----	----	----	----	----

intercambia ↑ ↑

17	5	23	12	44	38	84	77	90
----	---	----	----	----	----	----	----	----

El valor mayor se encuentra al final del arreglo



15. Ordenando un arreglo: Burbuja

Ejercicio 3: Implementar el algoritmo en Java

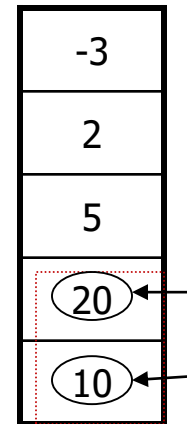
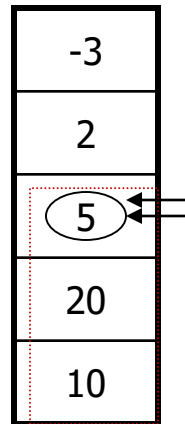
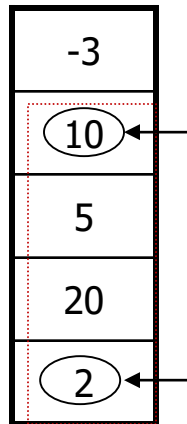
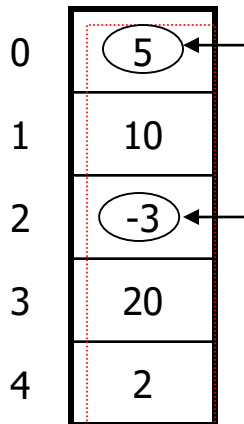
15. Ordenando un arreglo: Selección

- Hay varios algoritmos de ordenamiento con diferentes grados de complejidad y eficiencia
- Pseudocódigo Ordenamiento por Selección:

```
for (i=0; i<arreglo.longitud-1; i++)
```

```
    encuentra el valor más pequeño en el  
    arreglo desde la posición i hasta el final  
    Intercambia el valor encontrado con  
    arreglo[i]
```

arreglo(original)

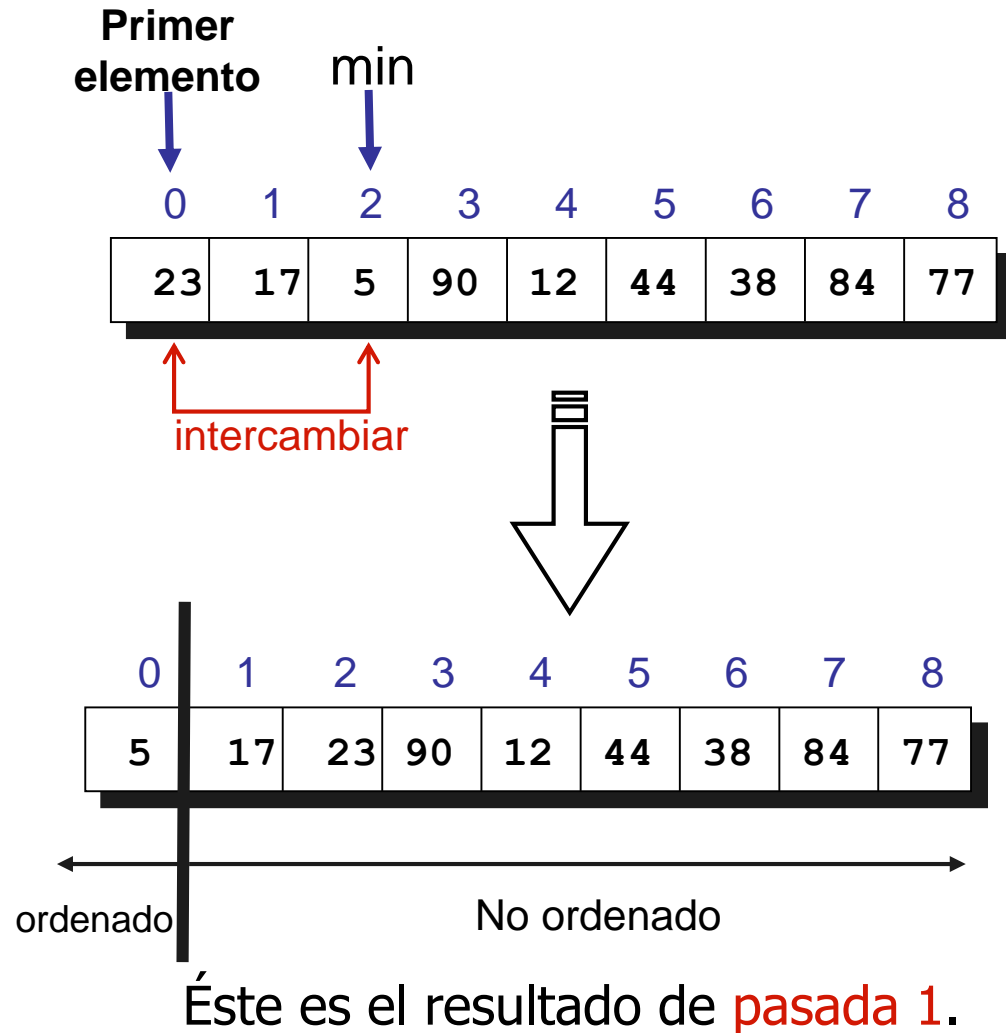


arreglo(ordenado)



Ordenamiento por Selección

1. Encontrar el elemento menor de la lista.
2. Intercambiar el elemento de la primera posición con el elemento menor. Ahora el elemento menor esta en la primera posición.
3. Repetir los Pasos 1 y 2 con el resto de la lista.



Ordenamiento por Selección, pasadas

Pass

	0	1	2	3	4	5	6	7	8
1	23	17	5	90	12	44	38	84	77



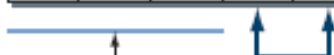
	0	1	2	3	4	5	6	7	8
2	5	17	23	90	12	44	38	84	77



	0	1	2	3	4	5	6	7	8
3	5	12	23	90	17	44	38	84	77



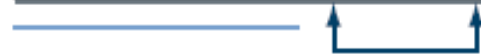
	0	1	2	3	4	5	6	7	8
4	5	12	17	90	23	44	38	84	77



Sorted

Pass

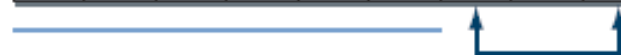
	0	1	2	3	4	5	6	7	8
5	5	12	17	23	90	44	38	84	77



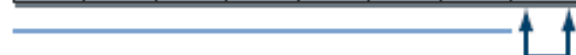
	0	1	2	3	4	5	6	7	8
6	5	12	17	23	38	44	90	84	77



	0	1	2	3	4	5	6	7	8
7	5	12	17	23	38	44	90	84	77



	0	1	2	3	4	5	6	7	8
8	5	12	17	23	38	44	77	84	90



8 iteraciones

Ordenamiento por Seleccion, pasadas

Pasadas #

0 1 2 3 4 5 6 7 8

1

5	17	23	90	12	44	38	84	77
---	----	----	----	----	----	----	----	----

ordenado

2

5	12	23	90	17	44	38	84	77
---	----	----	----	----	----	----	----	----

3

5	12	17	90	23	44	38	84	77
---	----	----	----	----	----	----	----	----

...

7

5	12	17	23	38	44	77	84	90
---	----	----	----	----	----	----	----	----

8

5	12	17	23	38	44	77	84	90
---	----	----	----	----	----	----	----	----

Resultado DESPUÉS
de que cada pasada
se ha completado.



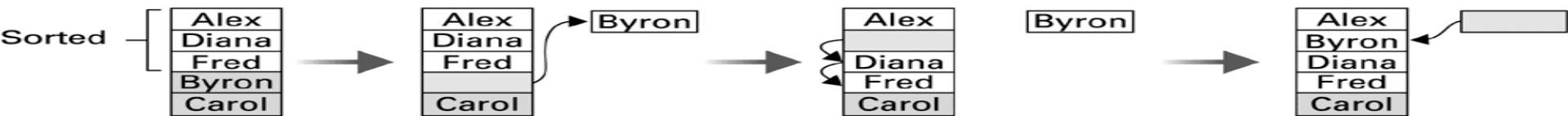
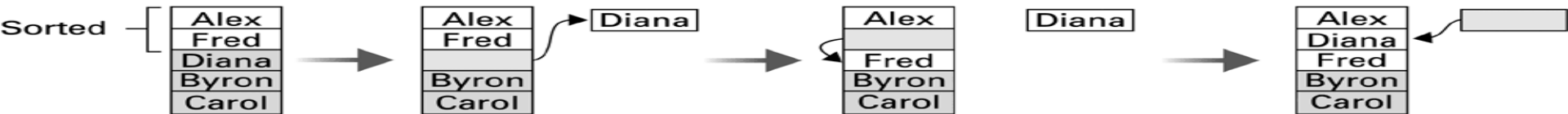
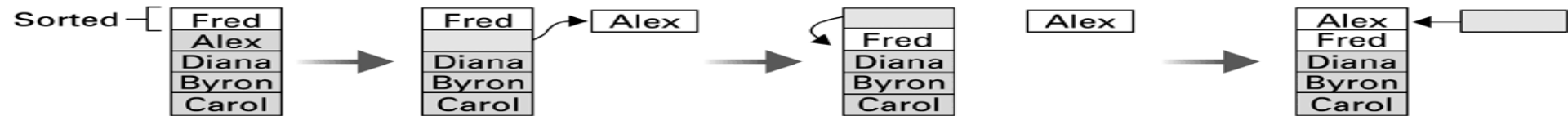
15. Ordenando un arreglo

Ejercicio 4: Implementar el algoritmo en Java

15. Ordenando un arreglo: Inserción

Initial list:

Fred
Alex
Diana
Byron
Carol



Sorted list:

Alex
Byron
Carol
Diana
Fred



Ordenamiento por Inserción (1/5)

- Bases del algoritmo

- Pasada i ,

- Insertar $v[i]$ en la posición correcta en la región ordenada de su izquierda: $v[0] \dots v[i-1]$.

- Ejemplo – pasada 1 (asuma $n = 5$)

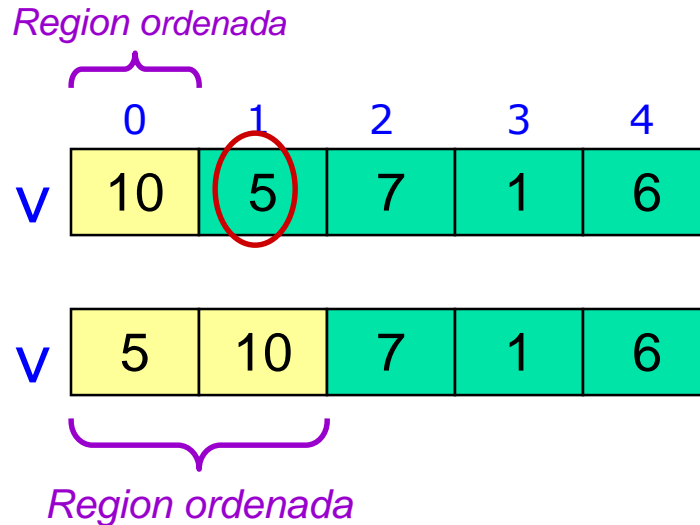
- Compare $v[1]$ con $v[0]$

- Si $v[1] < v[0]$, mover $v[1]$ delante de $v[0]$ (correr los otros valores en caso sea necesario)

- La region ordenada ahora consiste en dos elementos : $v[0]$ y $v[1]$.

Ordenamiento por Inserción (2/5)

Ejemplo – pasada 1



¿En dónde insertar $v[1]$ en la región ordenada de su izquierda?

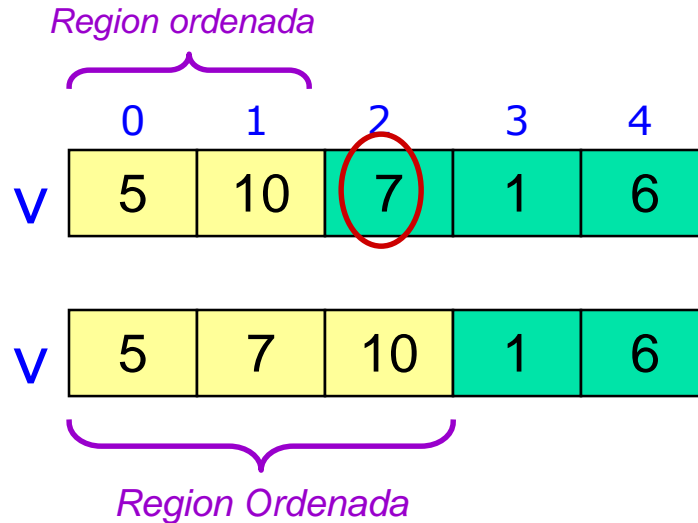
Aquí pueden ver más animaciones

<http://cs.smith.edu/~thiebaut/java/sort/demo.html>

<http://max.cs.kzoo.edu/~abrady/java/sorting/>

Ordenamiento por Inserción (3/5)

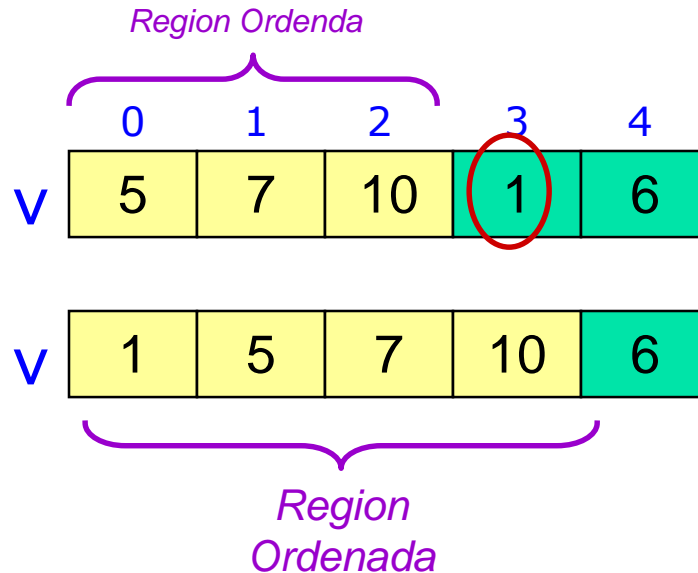
Ejemplo – pasada 2



¿En dónde insertar $v[2]$ en la región ordenada de su izquierda?

Ordenamiento por Inserción (4/5)

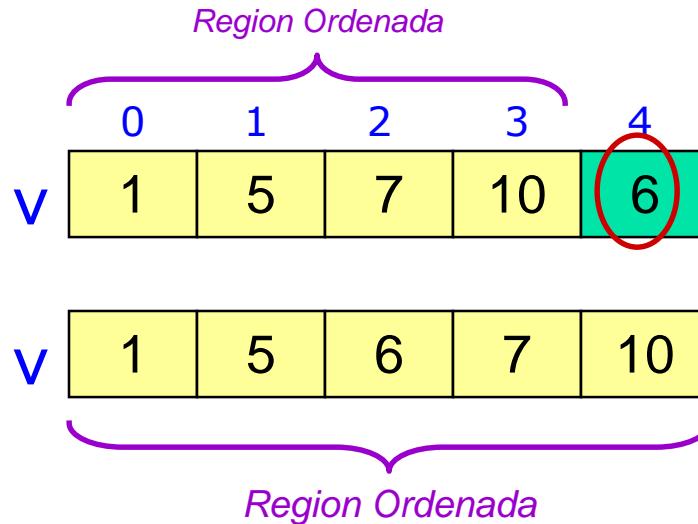
Ejemplo – pasada 3



¿En dónde insertar $v[3]$ en la región ordenada de su izquierda?

Ordenamiento por Inserción (5/5)

Ejemplo – pasada 4



¿En dónde insertar $v[4]$ en la región ordenada de su izquierda?

**Ordenamiento
Completo.**



15. Ordenando un arreglo

Ejercicio 5: Implementar el algoritmo en Java



15. Ordenando un arreglo

- Otros métodos
 - QuickSort
 - MergeSort
 - HeapSort
- Referencias de todos los métodos:
 - <http://www.sorting-algorithms.com>
 - <https://www.youtube.com/watch?v=EdUWyka7kpI>
 - <https://www.youtube.com/watch?v=NiyEqLZmngY>
 - <http://cs.smith.edu/~thiebaut/java/sort/demo.html>
 - <https://www.youtube.com/watch?v=aQiWF4E8flQ>
 - <https://www.youtube.com/watch?v=HsZ-YRQM8sE>
 - <http://youtube.com/watch?v=EreoMaOBTzE>
 - <https://www.youtube.com/watch?v=B7hVxCmfPtM>

Arreglos (Arrays)



Gracias