



# U5 Búsqueda y Ordenamiento

**Algoritmos y Estructuras de Datos**

Violeta Ocegueda

1



## U5.1 Métodos de búsqueda

**Algoritmos y Estructuras de Datos**

Violeta Ocegueda

2

## Búsqueda lineal



- Compara secuencialmente cada elemento hasta que da con el valor buscado o llega al final del arreglo.



- Pros
  - No requiere que el arreglo esté ordenado.
- Contra
  - Tiene el peor tiempo de ejecución de los algoritmos de búsqueda.

7 4 1 8 3 6

Si buscamos el valor 3:

7 4 1 8 3 6 3 == 7? **✗**

7 4 1 8 3 6 3 == 4? **✗**

7 4 1 8 3 6 3 == 1? **✗**

7 4 1 8 3 6 3 == 8? **✗**

7 4 1 8 3 6 3 == 3? **✓**

7 4 1 8 3 6 **Encontrado!**

3

## Búsqueda lineal



7 4 1 8 3 6

Si buscamos el valor 0:

7 4 1 8 3 6 0 == 7? **✗**

7 4 1 8 3 6 0 == 4? **✗**

7 4 1 8 3 6 0 == 1? **✗**

7 4 1 8 3 6 0 == 8? **✗**

7 4 1 8 3 6 0 == 3? **✗**

7 4 1 8 3 6 0 == 6? **✗**

7 4 1 8 3 6 **No encontrado.**

4



## Búsqueda binaria

- Compara si el valor buscado está en el centro del arreglo, si no está descarta un extremo del arreglo y vuelve a buscar en el centro del arreglo restante.
- Pro
  - Es fácil de implementar.
- Contra
  - Requiere un arreglo ordenado, preferentemente uniformemente distribuido.

5



## Búsqueda binaria

0 1 2 3 4 5 6 7 8 9

Si buscamos el valor 3:

El primer elemento comparado es el que está en la posición  $n/2$ , donde  $n$  es el tamaño del arreglo:  $10/2 = 5$

0 1 2 3 4 5 6 7 8 9

3 == 5? ✗

3 < 5 ✓

0 1 2 3 4 5 6 7 8 9

$$(5-1)/2 = 2$$

0 1 2 3 4 5 6 7 8 9

3 == 2? ✗    3 < 2 ✗

0 1 2 3 4 5 6 7 8 9

$$[(2+1) + 4]/2 = 3$$

0 1 2 3 4 5 6 7 8 9

3 == 3? ✓

6



# U5.2 Métodos de ordenamiento

## Algoritmos y Estructuras de Datos

Violeta Ocegueda

7



## Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
0	1000	-1

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
0	12	0

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
1	9	1

8



## Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
2	3	2

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
3	3	2

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
4	3	2

Algoritmos y Estructuras de Datos

9

9



## Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	3	2

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
6	3	2

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Ya que recorrimos todo el arreglo, intercambiamos la posición del valor menor con la primera posición del arreglo.

Algoritmos y Estructuras de Datos

10

10



## Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
1	1000	-1

En la segunda corrida, estos son los valores iniciales

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
1	9	1

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
2	9	1

Algoritmos y Estructuras de Datos

11

11



## Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
3	7	3

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
4	7	3

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	7	3

Algoritmos y Estructuras de Datos

12

12

# Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
6	7	3

Ahora intercambiamos el valor menor por la segunda posición del arreglo.

3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
2	1000	-1

En la tercera corrida, estos son los valores iniciales

3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Algoritmos y Estructuras de Datos 13

13

# Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
2	12	2

3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
3	9	3


3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
4	9	3

Algoritmos y Estructuras de Datos 14

14

# Selección



Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	9	3

3	7	12	9	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
6	8	6


Ahora intercambiamos el valor menor por la tercera posición del arreglo.

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Algoritmos y Estructuras de Datos 15

15

# Selección



Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
3	1000	-1

En la cuarta corrida, estos son los valores iniciales

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
3	9	3

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]


i	Menor	Posición
4	9	3

Algoritmos y Estructuras de Datos 16

16



# Selección



Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	9	3

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
6	9	3


3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Ahora intercambiamos el valor menor por la cuarta posición del arreglo.

Algoritmos y Estructuras de Datos
17

17

# Selección



Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
4	1000	-1

En la quinta corrida, estos son los valores iniciales

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
4	14	4

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	11	5

Algoritmos y Estructuras de Datos
18

18

# Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	8	9	14	11	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
6	11	5

Ahora intercambiamos el valor menor por la quinta posición del arreglo.

3	7	8	9	11	14	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	1000	-1

En la sexta corrida, estos son los valores iniciales

3	7	8	9	11	14	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Algoritmos y Estructuras de Datos 19

19

# Selección

Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	8	9	11	14	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
5	14	5

3	7	8	9	11	14	12
[0]	[1]	[2]	[3]	[4]	[5]	[6]

i	Menor	Posición
6	12	6

Ahora intercambiamos el valor menor por la sexta posición del arreglo.

3	7	8	9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Algoritmos y Estructuras de Datos 20

20

## Selección



Busco el elemento más pequeño y lo coloco en la posición más pequeña disponible.

3	7	8	9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

El arreglo ya quedó ordenado.

21

## Inserción



Vector original

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

9	12	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

3	9	12	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

3	7	9	12	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

3	7	9	12	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

22

## Inserción

3	7	9	11	12	14	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

3	7	8	9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

El arreglo ya quedó ordenado.

23

## Burbuja

Vector original

12	9	3	7	14	11	8
----	---	---	---	----	----	---

9	12	3	7	14	11	8
---	----	---	---	----	----	---

9	3	12	7	14	11	8
---	---	----	---	----	----	---

9	3	7	12	14	11	8
---	---	---	----	----	----	---

9	3	7	12	14	11	8
---	---	---	----	----	----	---

9	3	7	12	11	14	8
---	---	---	----	----	----	---

9	3	7	12	11	8	14
---	---	---	----	----	---	----

9	3	7	12	11	8	14
---	---	---	----	----	---	----

24

# Burbuja



9	3	7	12	11	8	14
---	---	---	----	----	---	----

3	9	7	12	11	8	14
---	---	---	----	----	---	----

3	7	9	12	11	8	14
---	---	---	----	----	---	----

3	7	9	12	11	8	14
---	---	---	----	----	---	----

3	7	9	11	8	12	14
---	---	---	----	---	----	----

3	7	9	11	8	12	14
---	---	---	----	---	----	----

3	7	9	11	8	12	14
---	---	---	----	---	----	----

Algoritmos y Estructuras de Datos

25

25

# Burbuja



3	7	9	11	8	12	14
---	---	---	----	---	----	----

3	7	9	11	8	12	14
---	---	---	----	---	----	----

3	7	9	11	8	12	14
---	---	---	----	---	----	----

3	7	9	11	8	12	14
---	---	---	----	---	----	----

3	7	9	8	11	12	14
---	---	---	---	----	----	----

3	7	9	8	11	12	14
---	---	---	---	----	----	----

Algoritmos y Estructuras de Datos

26

26

# Burbuja

Diagram illustrating the first pass of the Bubble Sort algorithm. The array is [3, 7, 9, 8, 11, 12, 14]. The left side shows the progression of the first pass: 3 and 7 are compared, 7 and 9 are compared, 9 and 8 are compared and swapped, and 8 and 11 are compared. The right side shows the initial state with a green arrow pointing down, indicating the start of the process.

Algoritmos y Estructuras de Datos 27

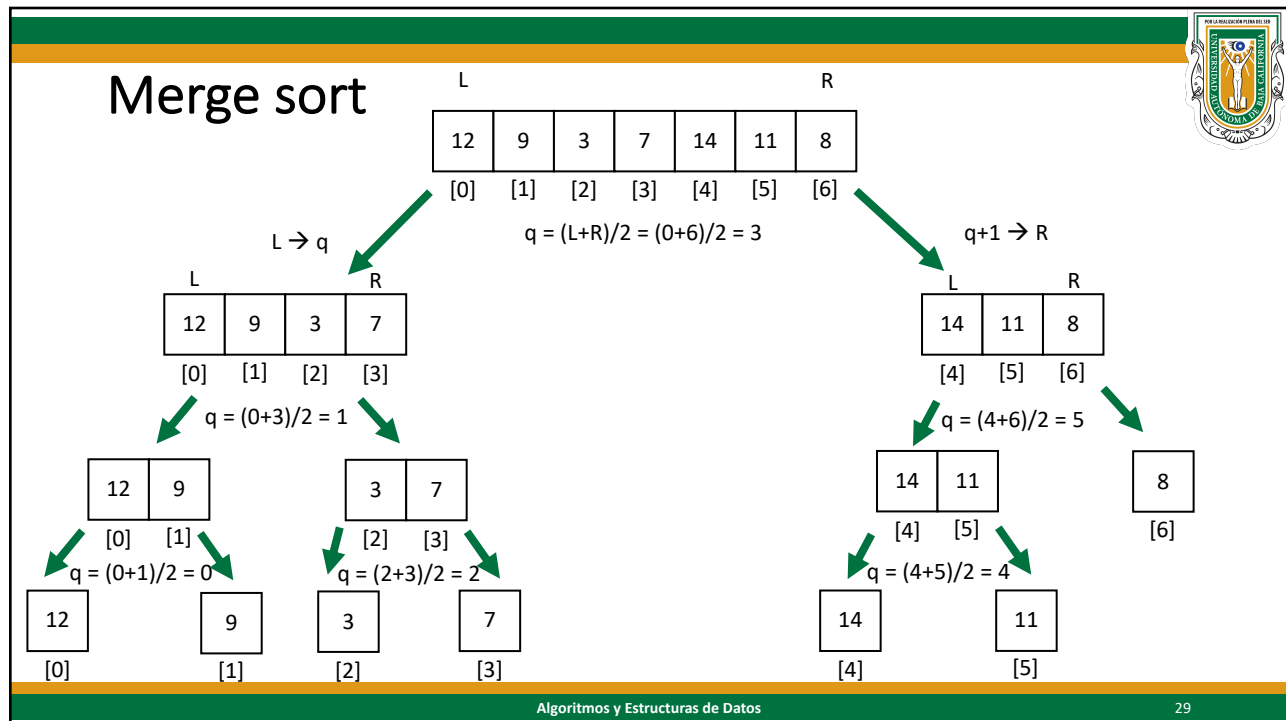
27

# Burbuja

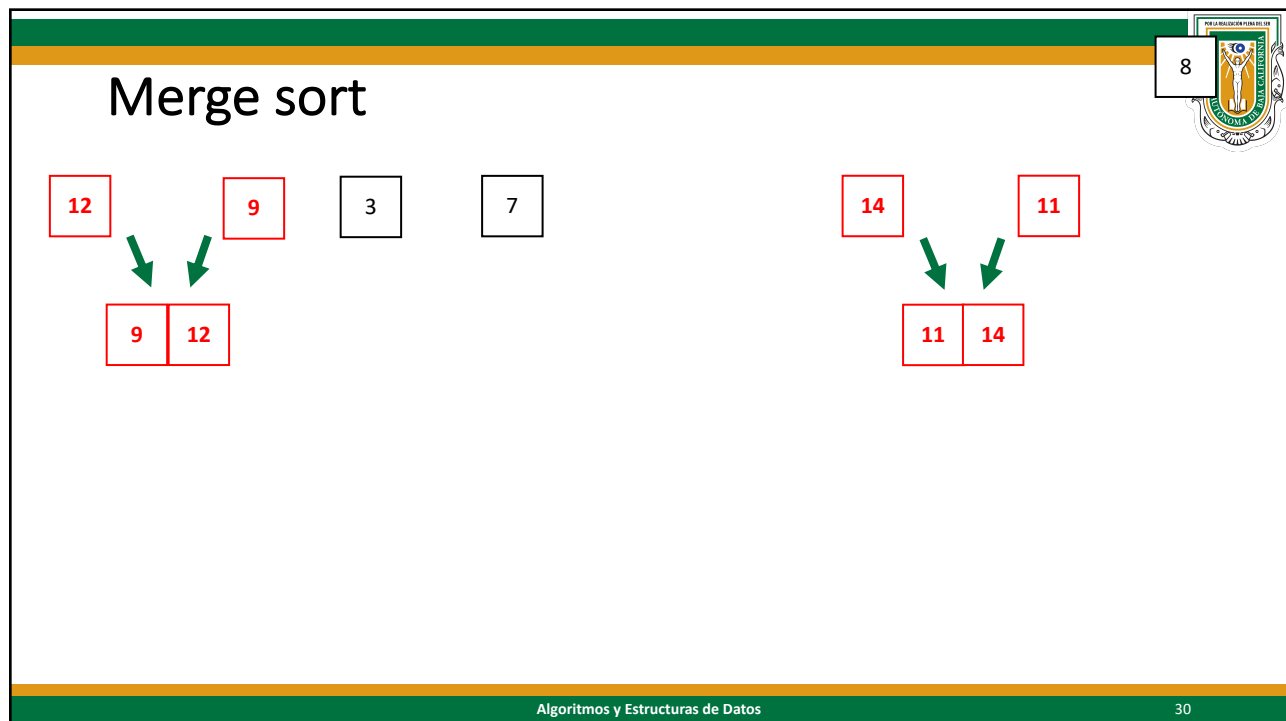
Diagram illustrating the second pass of the Bubble Sort algorithm. The array is [3, 7, 8, 9, 11, 12, 14]. The left side shows the progression of the second pass: 3 and 7 are compared, 7 and 8 are compared, 8 and 9 are compared, and 9 and 11 are compared. The right side shows the initial state with a green arrow pointing down, indicating the start of the process.

Algoritmos y Estructuras de Datos 28

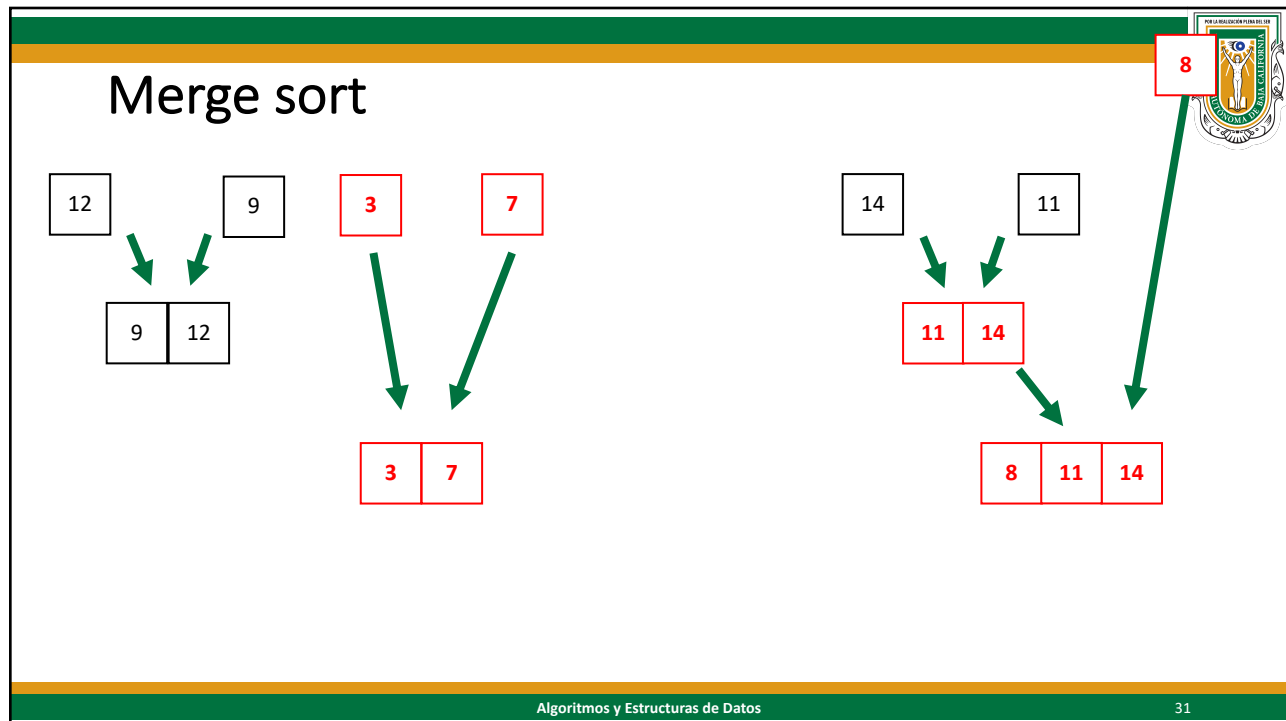
28



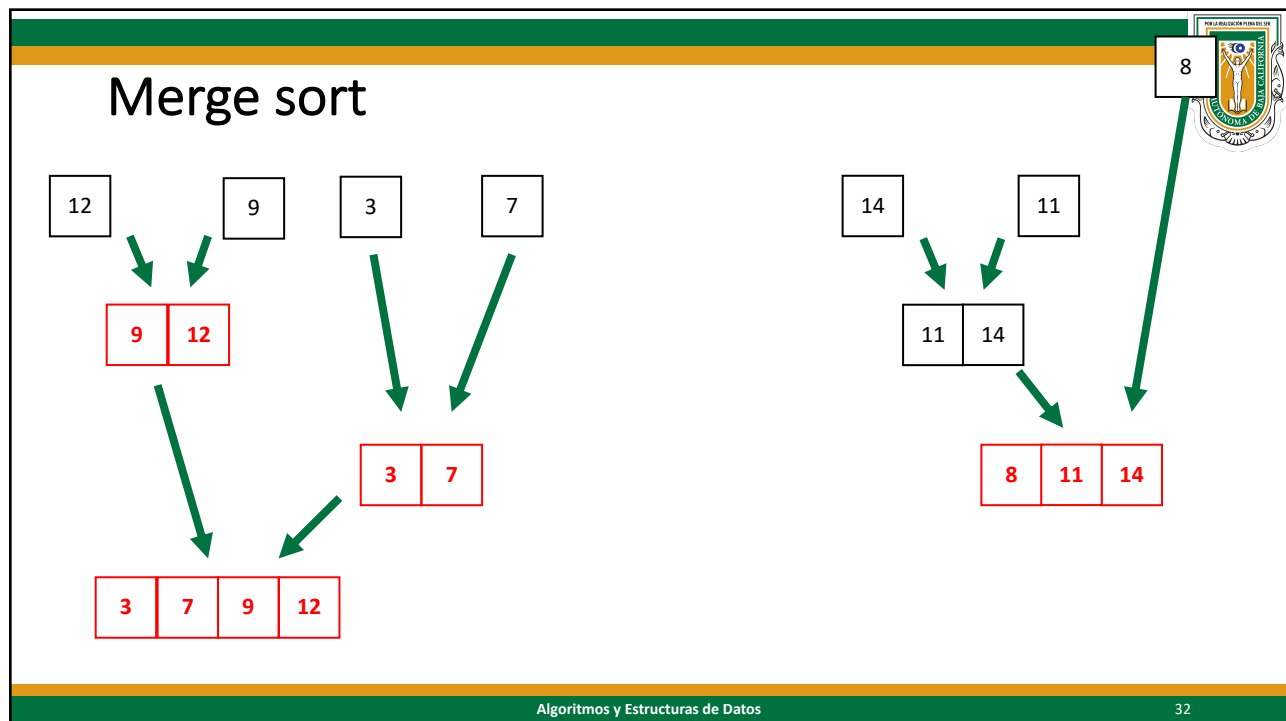
29



30



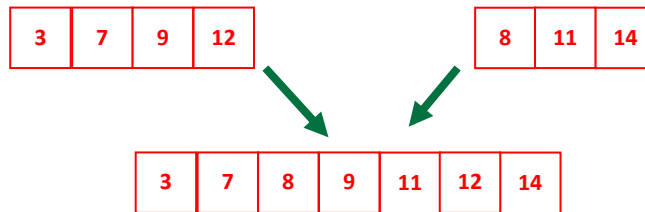
31



32



## Merge sort



Algoritmos y Estructuras de Datos

33

33

## Quick sort

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.


8
---

Algoritmos y Estructuras de Datos

34

34

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.


8

12

Algoritmos y Estructuras de Datos 35

35

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.


8

12 9

Algoritmos y Estructuras de Datos 36

36

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---


Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	8	12	9
---	---	----	---

Algoritmos y Estructuras de Datos 37

37

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---


Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7	8	12	9
---	---	---	----	---

Algoritmos y Estructuras de Datos 38

38

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---


Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7	8	12	9	14
---	---	---	----	---	----

Algoritmos y Estructuras de Datos
39

39

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---


Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7	8	12	9	14	11
---	---	---	----	---	----	----

Algoritmos y Estructuras de Datos
40

40

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Ahora elijo el pivote para el arreglo que quedo a la izquierda, y otro pivote para el arreglo que quedo a la derecha del pivote anterior.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

7
---


8
---

11
----

Algoritmos y Estructuras de Datos
41

41

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Ahora elijo el pivote para el arreglo que quedo a la izquierda, y otro pivote para el arreglo que quedo a la derecha del pivote anterior.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3
---

7
---

8
---


11
----

12
----

Algoritmos y Estructuras de Datos
42

42

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Ahora elijo el pivote para el arreglo que quedo a la izquierda, y otro pivote para el arreglo que quedo a la derecha del pivote anterior.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7
---	---

8
---

9
---


11
----

12
----

Algoritmos y Estructuras de Datos
43

43

## Quick sort



12	9	3	7	14	11	8
----	---	---	---	----	----	---

Primero elijo un pivote, puede ser cualquier elemento del arreglo.

12	9	3	7	14	11	8
----	---	---	---	----	----	---

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Ahora elijo el pivote para el arreglo que quedo a la izquierda, y otro pivote para el arreglo que quedo a la derecha del pivote anterior.

3	7
---	---

8
---

12	9	14	11
----	---	----	----

Acomodo los elementos menores que el pivote a su izquierda y los mayores que el pivote a su derecha.

3	7
---	---

8
---

9
---

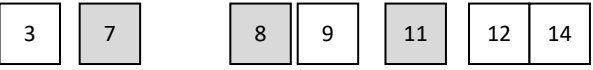
11
----

12	14
----	----

Algoritmos y Estructuras de Datos
44

44

## Quick sort

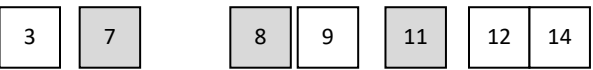


3 7 8 9 11 12 14

Algoritmos y Estructuras de Datos 45

45

## Quick sort



3 7 8 9 11 12 14


Elegimos nuevos pivotes en los segmentos del arreglo que no está ordenado.

3 7 8 9 11 12 14

Algoritmos y Estructuras de Datos 46

46

## Quick sort



3 7 8 9 11 12 14

Elegimos nuevos pivotes en los segmentos del arreglo que no está ordenado.

3 7 8 9 11 12 14

Cuando en el segmento del arreglo en el que se encuentra el pivote sólo está el pivote, consideramos que éste ya está ordenado.

3 7 8 9 11 12 14

Algoritmos y Estructuras de Datos 47

47

## Quick sort



3 7 8 9 11 12 14

Elegimos nuevos pivotes en los segmentos del arreglo que no está ordenado.

3 7 8 9 11 12 14

Cuando en el segmento del arreglo en el que se encuentra el pivote sólo está el pivote, consideramos que éste ya está ordenado.

3 7 8 9 11 12 14

Continuamos con el segmento de arreglo que todavía tiene elementos por ordenar.

3 7 8 9 11 12 14

Algoritmos y Estructuras de Datos 48

48



## Quick sort




Diagram illustrating the Quick Sort algorithm steps:

Initial array: [3, 7, 8, 9, 11, 12, 14]

Step 1: Partitioning around pivot 7. The array is split into [3] and [8, 9, 11, 12, 14].

Step 2: Partitioning around pivot 9. The array [8, 9, 11, 12, 14] is split into [8] and [11, 12, 14].

Step 3: Partitioning around pivot 12. The array [11, 12, 14] is split into [11] and [14].

Final sorted array: [3, 7, 8, 9, 11, 12, 14]

Algoritmos y Estructuras de Datos 49

49

## Quick sort




Diagram illustrating the Quick Sort algorithm steps:

Initial array: [3, 7, 8, 9, 11, 12, 14]

Step 1: Partitioning around pivot 7. The array is split into [3] and [8, 9, 11, 12, 14].

Step 2: Partitioning around pivot 9. The array [8, 9, 11, 12, 14] is split into [8] and [11, 12, 14].

Step 3: Partitioning around pivot 12. The array [11, 12, 14] is split into [11] and [14].

Final sorted array: [3, 7, 8, 9, 11, 12, 14]

Algoritmos y Estructuras de Datos 50

50

## Quick sort

Elegimos nuevos pivotes en los segmentos del arreglo que no está ordenado.

Cuando en el segmento del arreglo en el que se encuentra el pivote sólo está el pivote, consideramos que éste ya está ordenado.

Continuamos con el segmento de arreglo que todavía tiene elementos por ordenar.

Elegimos nuevos pivotes en los segmentos del arreglo que no está ordenado.

Cuando en el segmento del arreglo en el que se encuentra el pivote sólo está el pivote, consideramos que éste ya está ordenado.

El arreglo ya está ordenado.

Algoritmos y Estructuras de Datos 51

51

## Heap sort

- Dos tipos:
  - **Max Heap**: el nodo padre tiene un valor más alto o igual que sus hijos.
  - **Min Heap**: el nodo padre tiene un valor más pequeño o igual que sus hijos.
- Consiste en:
  - Convertir el arreglo en un heap.
  - Repetir hasta que el heap tenga sólo 1 elemento:
    - Intercambiar la raíz del heap por una posición en el vector:
      - Max Heap: la raíz reemplaza la última posición disponible del vector.
      - Min Heap: la raíz reemplaza la primera posición disponible del vector.
    - Remove el último elemento del heap.
    - Heapify los elementos restantes en el heap.

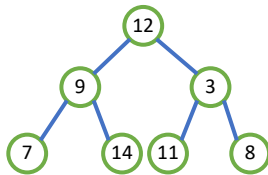
Algoritmos y Estructuras de Datos 52

52

# Heap sort

12	9	3	7	14	11	8
[0]	[1]	[2]	[3]	[4]	[5]	[6]

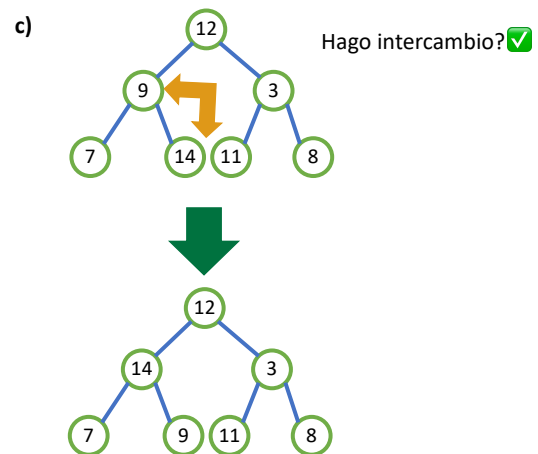
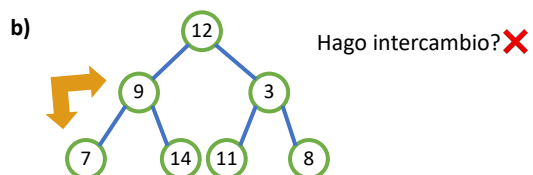
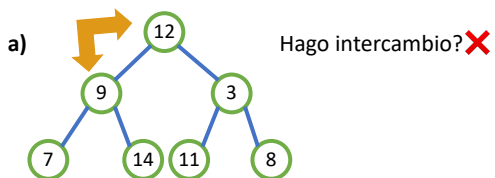
## 1. Construir árbol binario



53

# Heap sort

## 2. Convertir árbol resultante en un Max Heap (heapify)

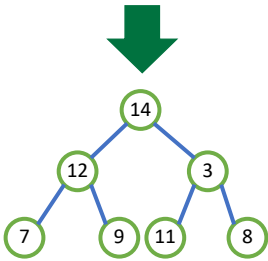
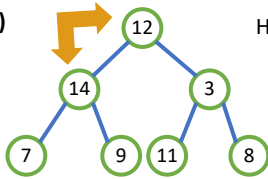


54

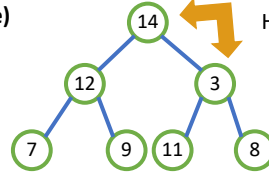
# Heap sort



d) Hago intercambio? ☒



e) Hago intercambio? ☐



Algoritmos y Estructuras de Datos

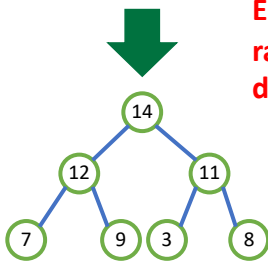
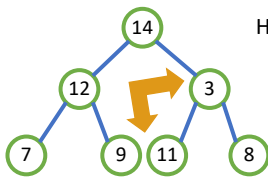
55

55

# Heap sort

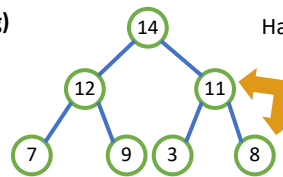


f) Hago intercambio? ☒

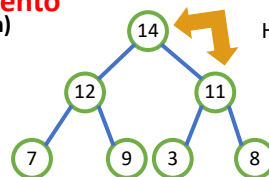


En este momento ya sé que la raíz de mi heap es el elemento de mayor valor.

g) Hago intercambio? ☐



h) Hago intercambio? ☐



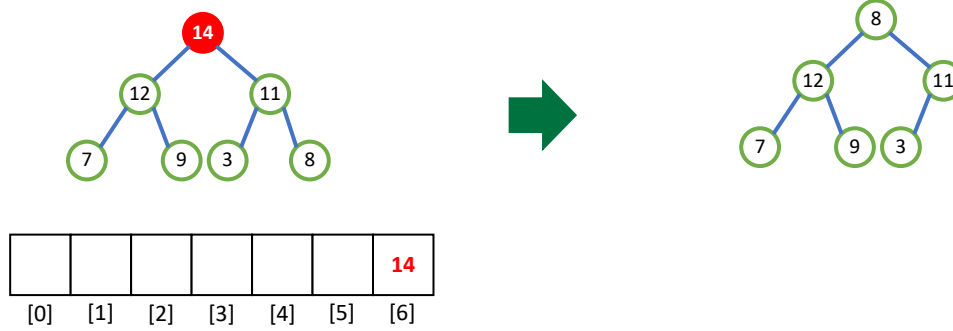
Algoritmos y Estructuras de Datos

56

56

# Heap sort

3. Remove la raíz de heap y sustituirlo por el último elemento del heap.



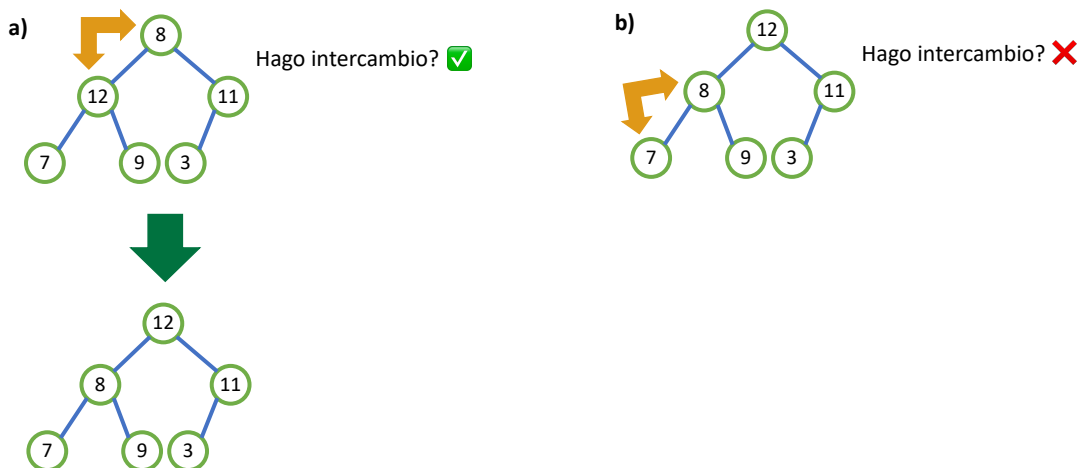
Algoritmos y Estructuras de Datos

57

57

# Heap sort

2. Convertir árbol resultante en un Max Heap (heapify)

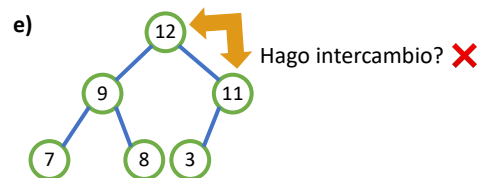
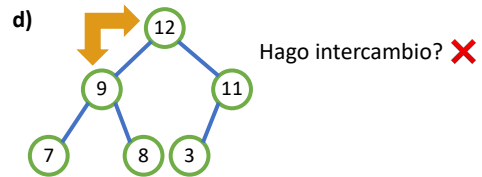
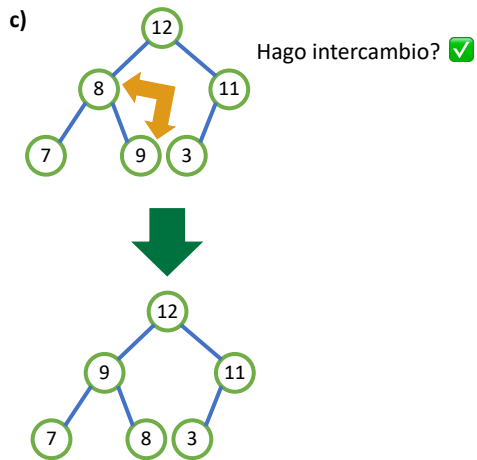


Algoritmos y Estructuras de Datos

58

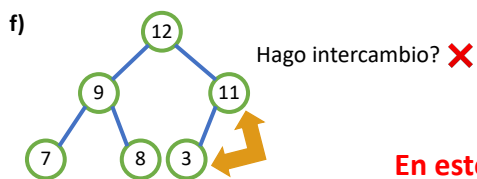
58

# Heap sort



59

# Heap sort

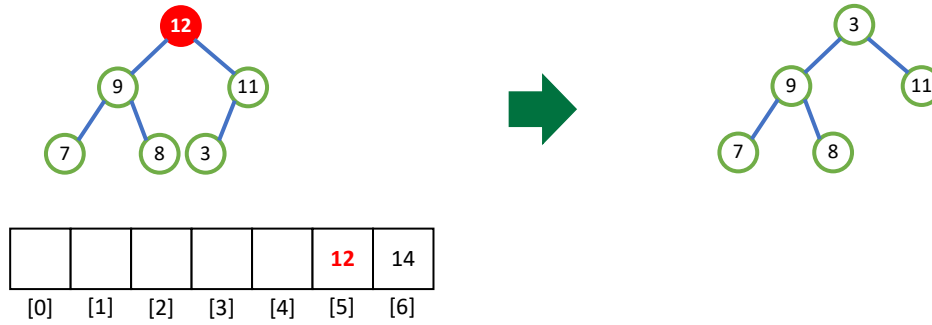


**En este momento ya sé que la raíz de mi heap es el elemento de mayor valor.**

60

# Heap sort

3. Remove la raíz de heap y sustituirlo por el último elemento del heap.



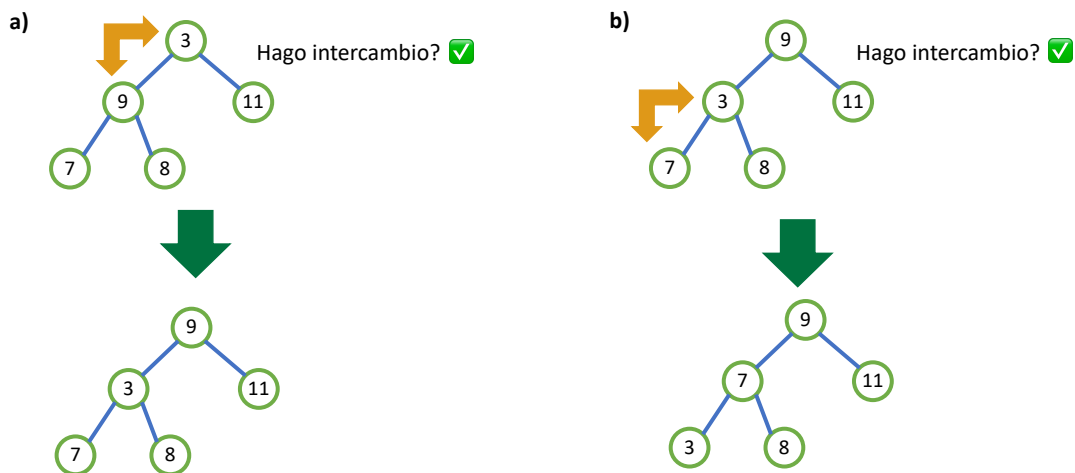
Algoritmos y Estructuras de Datos

61

61

# Heap sort

2. Convertir árbol resultante en un Max Heap (heapify)

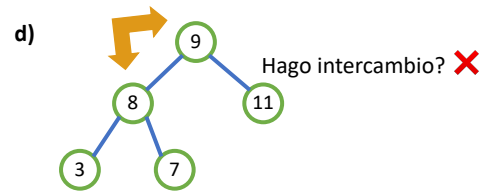
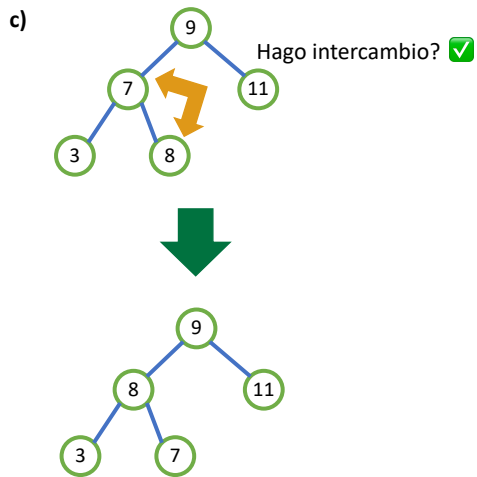


Algoritmos y Estructuras de Datos

62

62

## Heap sort

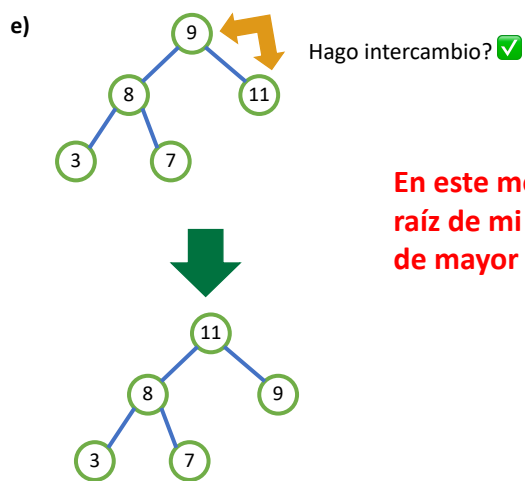


Algoritmos y Estructuras de Datos

63

63

## Heap sort



**En este momento ya sé que la raíz de mi heap es el elemento de mayor valor.**

Algoritmos y Estructuras de Datos

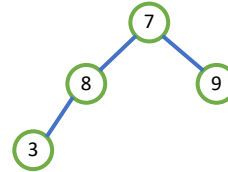
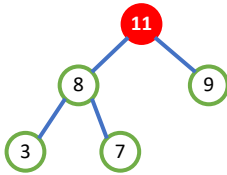
64

64



# Heap sort

3. Remove la raíz de heap y sustituirlo por el último elemento del heap.



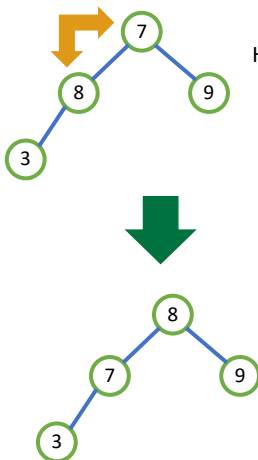
				11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

65

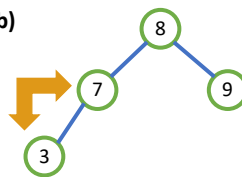
# Heap sort

2. Convertir árbol resultante en un Max Heap (heapify)

a) Hago intercambio? ☒



b) Hago intercambio? ☐



66

# Heap sort

c)

Hago intercambio? ☒

En este momento ya sé que la raíz de mi heap es el elemento de mayor valor.

Algoritmos y Estructuras de Datos 67

67

# Heap sort

3. Remove la raíz de heap y sustituirlo por el último elemento del heap.

			9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

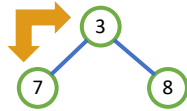
Algoritmos y Estructuras de Datos 68

68

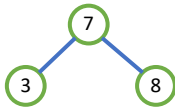
# Heap sort

## 2. Convertir árbol resultante en un Max Heap (heapify)

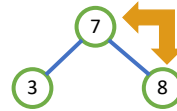
a)



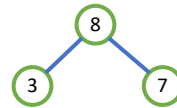
Hago intercambio? ✓



b)



Hago intercambio? ✓



En este momento ya sé que la raíz de mi heap es el elemento de mayor valor.

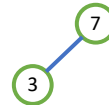
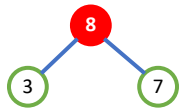
Algoritmos y Estructuras de Datos

69

69

# Heap sort

## 3. Remove la raíz de heap y sustituirlo por el último elemento del heap.



		8	9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

Algoritmos y Estructuras de Datos

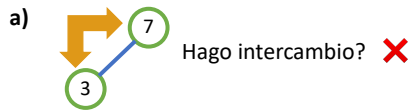
70

70

# Heap sort



## 2. Convertir árbol resultante en un Max Heap (heapify)



En este momento ya sé que la raíz de mi heap es el elemento de mayor valor.

## 3. Remove la raíz de heap y sustituirlo por el último elemento del heap.



	7	8	9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

71

# Heap sort



## 2. Convertir árbol resultante en un Max Heap (heapify)



Pero como ya sólo queda un elemento en el heap, ese elemento va en la última posición disponible del vector.

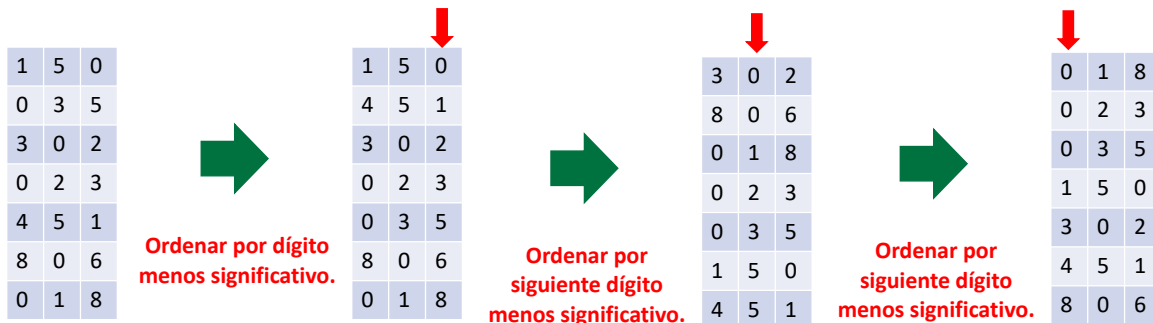
3	7	8	9	11	12	14
[0]	[1]	[2]	[3]	[4]	[5]	[6]

72

## Radix sort



150	35	302	23	451	806	18
[0]	[1]	[2]	[3]	[4]	[5]	[6]



Algoritmos y Estructuras de Datos

73

73

## Shell Sort




- Se comparan los números a una distancia  $K$ , inicialmente
  - $K = n / 2$
  - Donde  $n$  es el tamaño del arreglo.
- Cuando un valor de  $K$  ya no sirve para hacer intercambios, se calcula el siguiente valor de  $k$ :
  - $K = K/2$
- Si tenemos el arreglo:
  - 94 14 25 33 82 25 59 13
  - Entonces  $K = 8 / 2 = 4$

Algoritmos y Estructuras de Datos

74

74



# Shell Sort

$K = 4$

Índices:	0	1	2	3	4	5	6	7
Elementos del vector:	94	14	25	33	82	25	59	13

Valor en posición inicial

Valor en salto

Algoritmos y Estructuras de Datos

75

75

# Shell Sort

K = 4

Índices:	0	1	2	3	4	5	6	7
Elementos del vector:	94	14	25	33	82	25	59	13
	82	14	25	33	94	25	59	13
	82	14	25	33	94	25	59	13
	82	14	25	33	94	25	59	13
	82	14	25	33	94	25	59	13
	82	14	25	13	94	25	59	33
	82	14	25	13	94	25	59	33
	82	14	25	13	94	25	59	33
	82	14	25	13	94	25	59	33
	82	14	25	13	94	25	59	33

Como no hubo intercambio en la vuelta, recalculamos K

$$K = 4 / 2 = 2$$

Algoritmos y Estructuras de Datos

76

76

[illegible][illegible]

# Shell Sort

$K = 1$



Índices:	0	1	2	3	4	5	6	7
Elementos del vector:	13	14	25	25	59	33	82	94
	13	14	25	25	59	33	82	94
	13	14	25	25	59	33	82	94
	13	14	25	25	59	33	82	94
	13	14	25	25	59	33	82	94
	13	14	25	25	33	59	82	94

Va a dar otra vuelta para asegurarse de que ya no hayan intercambios, pero el vector ya está ordenado.