

How many shortest-length paths are there to get from your house to the doughnut shop?

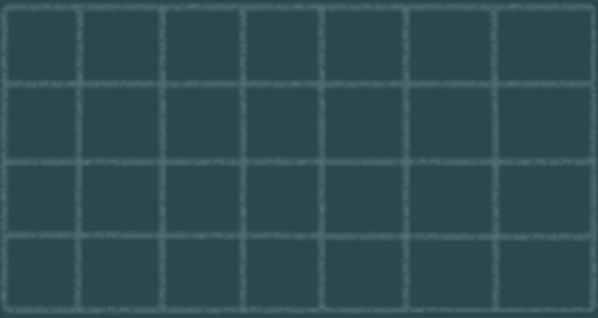


$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

P	Q	R	P V Q	P V R	(P V Q) ^ (P V R)
T	T	T	T	T	T
T	T	F	T	T	T
T	F	T	T	T	T
T	F	F	T	T	T
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	F	T	F
F	F	F	F	F	F

7, 11, 15, 19, 23...

$$\begin{aligned}
 a_1 - a_0 &= 4 \\
 a_2 - a_1 &= 4 \\
 a_3 - a_2 &= 4 \\
 &\vdots \\
 + a_n - a_{n-1} &= 4
 \end{aligned}$$



$$\binom{11}{7} = \binom{11}{4} = 330 \text{ paths}$$

101

101

101000101

B₉

$$e^{i\pi} + 1 = 0$$



One-to-One

Find 7 + 12 + 17 + 22 + ... + 342.

$$S_n = 7 + 12 + 17 + 22 + \dots + 342$$

$$+ S_n = 342 + 337 + 332 + 327 + \dots + 7$$

$$2S_n = 349 + 349 + 349 + 349 + \dots + 349$$

$$2S_n = 349 \cdot 68$$



ESTRUCTURAS DE DATOS

LIC. REDES Y SERVICIOS DE CÓMPUTO



There are six dogs to give 13 tacos. use a 'stars and bars' diagram to illustrate the first and sixth dog get 3 tacos, the second dog gets none, the third dog gets 5 and the fourth dog gets one.



$$(A \cup B \cup C) \cup (A \cap B \cap C)$$



$$v - e + f = 2$$

P.I.E. Example:

Original: $\forall x \forall y (x \geq 2y \rightarrow x > y + 1)$

Converse: $\exists x \forall y (x > y + 1 \rightarrow x \geq 2y)$

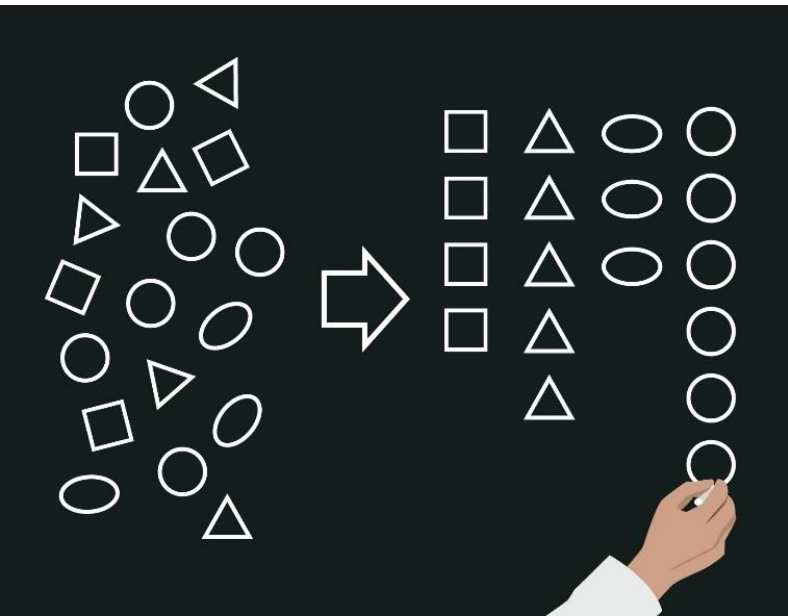
Negation: $\neg [\exists x \forall y (\neg (x \geq 2y) \vee x > y + 1)]$

$\forall x \exists y (x \geq 2y \wedge x \leq y + 1)$

Contrapositive: $\exists x \forall y (x \leq y + 1 \rightarrow x < 2y)$

Clase

- Unidad IX. Métodos de ordenamiento
- Métodos de ordenamiento.
 - Métodos de ordenamiento simples
 - Métodos de ordenamiento complejos.



El ordenamiento de datos (es decir, colocar los datos en cierto orden específico, como ascendente o descendente) es una de las aplicaciones computacionales más importantes.



Ordenar significa reagrupar o reorganizar un conjunto de datos u objetos en una secuencia específica.

Los procesos de ordenación y búsqueda son frecuentes en nuestra vida diaria, en un mundo desarrollado y acelerado en el que la información es de vital importancia.

Y la operación de búsqueda de información generalmente se hace sobre elementos ordenados.

Métodos de ordenamiento

- Encontramos elementos ordenados en cualquier lugar, directorios telefónicos, registros de pacientes, registros de huéspedes, índices de libros.
- La ordenación es una actividad fundamental y relevante.



Métodos de ordenamiento

- Formalmente se define ordenación de la siguiente manera:

Sea A una lista de N elementos

$A_1, A_2, A_3, \dots, A_n$

Ordenar significa permutar estos elementos de tal forma que queden de acuerdo con una distribución preestablecida.

Ascendente: $A_1 \leq A_2 \leq A_3 \dots \leq A_n$

Descendente: $A_1 \geq A_2 \geq A_3 \dots \geq A_n$

Métodos de ordenamiento

Ordenación de arreglos



- También llamada ordenación interna.
- Los elementos se encuentran en la memoria principal de la computadora.

Ordenación de archivos

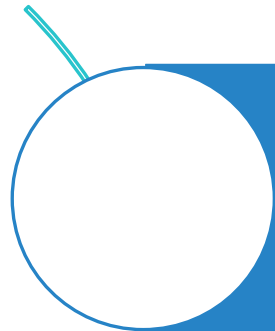


- También llamada ordenación externa.
- Los elementos se encuentran en archivos almacenados en dispositivos de almacenamiento secundario.

Métodos de ordenamiento

- Métodos directos:

A su vez pueden clasificarse en:



Métodos directos: Su implementación es relativamente sencilla y son fáciles de comprender.

Son ineficientes cuando el número de elementos es mediano o grande.



Métodos logarítmicos: Son más complejos que los directos y su elaboración más sofisticada. Son menos intuitivos y difíciles de entender.

Son más eficientes pues realizan menos comparaciones y movimientos para ordenar los elementos.

 <u>Insertion</u>	 <u>Selection</u>	 <u>Bubble</u>	 <u>Shell</u>	 <u>Merge</u>	 <u>Heap</u>	 <u>Qu</u>
						
						
						
						

Métodos de ordenamiento

- Un punto importante respecto a la ordenación interna es que el resultado final (el vector ordenado) será el mismo, sin importar qué algoritmo se utilice para ordenarlo.
- La elección del algoritmo sólo afecta al tiempo de ejecución y el uso que haga el programa de la memoria.

Métodos de ordenamiento

Los métodos directos más conocidos son:

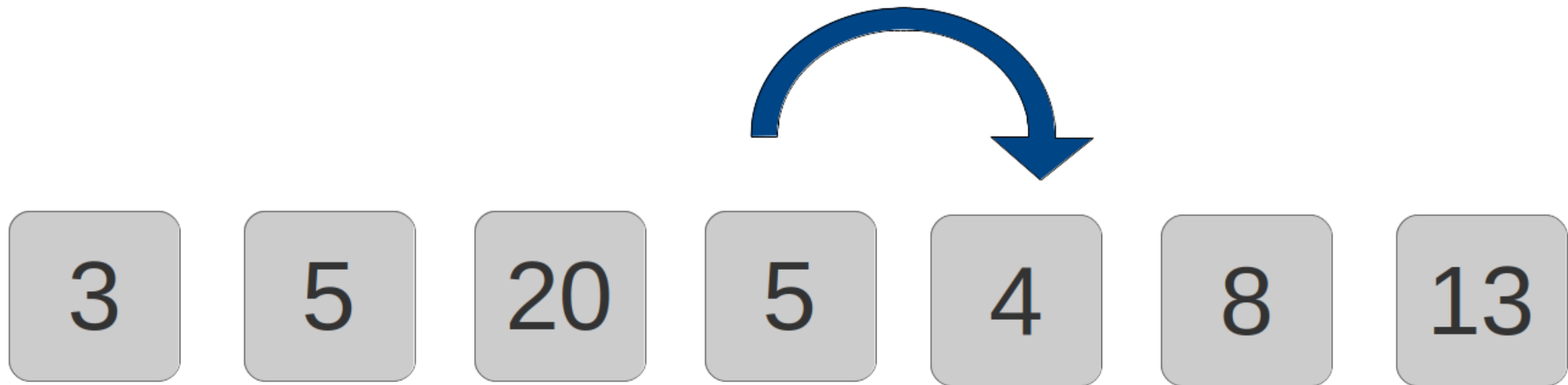
Ordenación por intercambio.

Ordenación por inserción directa.

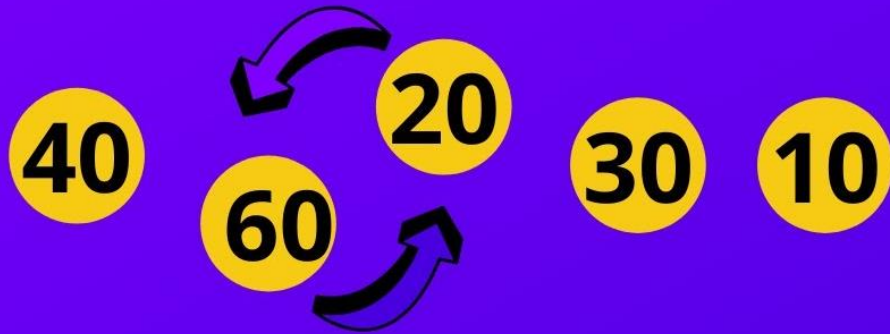
Ordenación por selección.

Ordenamiento de burbuja

Algoritmos de ordenamiento



Algoritmos



Método Burbuja

Métodos de ordenamiento

- Ordenación por intercambio directo.
- También llamado de Burbuja.
- Es el de más fácil comprensión, pero es el más ineficiente.

Métodos de ordenamiento

- Ordenación por intercambio directo.

¿Cómo funciona?

- Puede trabajar de dos maneras, llevando los elementos más pequeños del lado izquierdo o trasladando los elementos más grande de lado derecho.
- La idea básica de este algoritmo consiste en comparar pares de elementos adyacentes e intercambiarlos entre si hasta que todos se encuentren ordenados.
- Se realizan $(n-1)$ pasadas transportando en cada una de ellas el menor o mayor elemento, según sea el caso, a su posición ideal.
- Al final de las pasadas, los elementos estarán ordenados.

Métodos de ordenamiento

- Ordenación por intercambio directo.

Ejemplo

Deseamos ordenar los siguientes elementos de un arreglo unidimensional A, transportando en cada pasada el menor a la parte izquierda.

A: 15 67 08 16 44 27 12 35

Las comparaciones que se realizan son:

Métodos de ordenamiento

- Ordenación por intercambio directo.

A: **08** **15** **67** **12** **16** **44** **27** 35

Primera pasada

$A[7] > A[8]$	$(12 > 35)$	no hay intercambio
$A[6] > A[7]$	$(27 > 12)$	sí hay intercambio
$A[5] > A[6]$	$(44 > 12)$	sí hay intercambio
$A[4] > A[5]$	$(16 > 12)$	sí hay intercambio
$A[3] > A[4]$	$(08 > 12)$	no hay intercambio
$A[2] > A[3]$	$(67 > 08)$	sí hay intercambio
$A[1] > A[2]$	$(15 > 08)$	sí hay intercambio

Luego de la primera pasada el arreglo queda de la siguiente manera:

A: 08 15 67 12 16 44 27 35

Métodos de ordenamiento

- Ordenación por intercambio directo.

Segunda pasada

A: 08 **12** **15** **67** 16 **27** **44** 35

$A[7] > A[8]$ $(27 > 35)$ no hay intercambio
 $A[6] > A[7]$ $(44 > 27)$ sí hay intercambio
 $A[5] > A[6]$ $(16 > 27)$ no hay intercambio
 $A[4] > A[5]$ $(12 > 16)$ no hay intercambio
 $A[3] > A[4]$ $(67 > 12)$ sí hay intercambio
 $A[2] > A[3]$ $(15 > 12)$ sí hay intercambio

A: 08 **12** 15 67 16 27 44 35

Métodos de ordenamiento

- Ordenación por intercambio directo.

A: 08 12 15 67 16 27 44 35

Resultados de las siguientes pasadas

3a. pasada:	08	12	15	16	67	27	35	44
4a. pasada:	08	12	15	16	27	67	35	44
5a. pasada:	08	12	15	16	27	35	67	44
6a. pasada:	08	12	15	16	27	35	44	67
7a. pasada:	08	12	15	16	27	35	44	67

Métodos de ordenamiento

Actividad:

Elabora el algoritmo de ordenación para el método de intercambio directo que transporta en cada pasada el menor elemento hacia la parte izquierda del arreglo A (arreglo unidimensional de N elementos).

Métodos de ordenamiento

- Ordenación por intercambio directo.

Algoritmo Burbuja_Menor. Algoritmo que ordena los elementos de un arreglo unidimensional utilizando el método de la burbuja. Transporta en cada pasada el elemento más pequeño a la izquierda del arreglo. A es un arreglo unidimensional de N elementos.

1. *Repetir con I desde 2 hasta N*
 - 1.1 *Repetir con J desde N hasta I*
 - 1.1.1 *Si $A(J - 1) > A[J]$ entonces*
Hacer $AUX \leftarrow A[J - 1]$, $A[J - 1] \leftarrow A[J]$ y $A[J] \leftarrow AUX$
 - 1.1.2 {Fin del condicional del paso 1.1.1}
 - 1.2 {Fin del ciclo del paso 1.1}
2. {Fin del ciclo del paso 1}

Métodos de ordenamiento

- Ordenación por **Inserción directa**.
- También se conoce como el método de la baraja, pues es el método utilizado por los jugadores de cartas cuando las ordenan.

¿Cómo funciona?

- Consiste en insertar un elemento del arreglo en su parte izquierda, que ya se encuentra ordenada. Este proceso se repite desde el segundo hasta el n -ésimo elemento

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

Ejemplo

Deseamos ordenar los siguientes elementos de un arreglo unidimensional A, por inserción directa.

A: 15 67 08 16 44 27 12 35

Las comparaciones que se realizan son:

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

A: 15 67 08 16 44 27 12 35

PRIMERA PASADA


$A[2] < A[1]$ ($67 < 15$) no hay intercambio

A: 15 67 08 16 44 27 12 35

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

A: 08 15 67 16 44 27 12 35



SEGUNDA PASADA

$A[3] < A[2]$ (08 < 67) sí hay intercambio


$A[2] < A[1]$ (08 < 15) sí hay intercambio

A: 08 15 67 16 44 27 12 35

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

A: 08 15 **16** **67** 44 27 12 35



TERCERA PASADA:

$A[4] < A[3]$ ($16 < 67$) sí hay intercambio
 $A[3] < A[2]$ ($16 < 15$) no hay intercambio

A: **08** **15** **16** **67** 44 27 12 35

Métodos de ordenamiento

- Ordenación por **Inserción directa**.
- Se muestra el arreglo al final de las siguientes pasadas:

4a. pasada:	08	15	16	44	67	27	12	35
5a. pasada:	08	15	16	27	44	67	12	35
6a. pasada:	08	12	15	16	27	44	67	35
7a. pasada:	08	12	15	16	27	35	44	67

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

Algoritmo Inserción_directa. Algoritmo que ordena los elementos de un arreglo haciendo uso del método por inserción directa. A es un arreglo de tipo unidimensional de N elementos.

1. *Repetir* con I desde 2 hasta N
Hacer $AUX \leftarrow A[I]$ y $K \leftarrow I - 1$
 - 1.1 Mientras $((K \geq 1) \text{ y } (AUX < A[K]))$ *Repetir*
Hacer $A[K + 1] \leftarrow A[K]$ y $K \leftarrow K - 1$
 - 1.2 {Fin del ciclo del paso 1.1}
Hacer $A[K + 1] \leftarrow AUX$
2. {Fin del ciclo del paso 1}

Métodos de ordenamiento

- Ordenación por **Inserción directa.**

A: 15, 67, 08, 16, 44, 27, 12, 35

1. *Repetir con I desde 2 hasta N*
Hacer $AUX \leftarrow A[I]$ y $K \leftarrow I - 1$
 - 1.1 Mientras $((K \geq 1) \text{ y } (AUX < A[K]))$ *Repetir*
Hacer $A[K + 1] \leftarrow A[K]$ y $K \leftarrow K - 1$
 - 1.2 {Fin del ciclo del paso 1.1}
Hacer $A[K + 1] \leftarrow AUX$
2. {Fin del ciclo del paso 1}

$i=2$

Desde $i=2$ hasta 8

$AUX = 67$

$K = 1$

Mientras $1 \geq 1$ y $67 < 15$

$A[2]=67$

A: 15, 67, 08, 16, 44, 27, 12, 35

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

A: 15, 67, 08, 16, 44, 27, 12, 35

1. *Repetir con I desde 2 hasta N*

Hacer $AUX \leftarrow A[I]$ y $K \leftarrow I - 1$

1.1 Mientras $((K \geq 1)$ y $(AUX < A[K]))$ *Repetir*

Hacer $A[K + 1] \leftarrow A[K]$ y $K \leftarrow K - 1$

1.2 {Fin del ciclo del paso 1.1}

Hacer $A[K + 1] \leftarrow AUX$

2. {Fin del ciclo del paso 1}

$i=3$

Desde $i=2$ hasta 8

$AUX = 08$

$K = 2$

Mientras $2 \geq 1$ y $08 < 67$

$A[3]=67$

$K=1$

A: 15, 67, 67, 16, 44, 27, 12, 35

Métodos de ordenamiento

- Ordenación por **Inserción directa.**

A: 15, 67, 08, 16, 44, 27, 12, 35

1. *Repetir con I desde 2 hasta N*
Hacer $AUX \leftarrow A[I]$ y $K \leftarrow I - 1$
 - 1.1 Mientras $((K \geq 1) \text{ y } (AUX < A[K]))$ *Repetir*
Hacer $A[K + 1] \leftarrow A[K]$ y $K \leftarrow K - 1$
 - 1.2 { Fin del ciclo del paso 1.1 }
2. { Fin del ciclo del paso 1 }

$i=3$

Desde $i=2$ hasta 8

$AUX = 08$

$K = 2$

Mientras $1 \geq 1$ y $08 < 15$

$A[2]=15$

$K=0$

$A[1]=08$

A: 08, 15, 67, 16, 44, 27, 12, 35

Métodos de ordenamiento

- Ordenación por **Inserción directa**.

A: 08, 15, 67, 16, 44, 27, 12, 35

1. Repetir con I desde 2 hasta N

Hacer $AUX \leftarrow A[I]$ y $K \leftarrow I - 1$

1.1 Mientras $((K \geq 1) \text{ y } (AUX < A[K]))$ Repetir

Hacer $A[K + 1] \leftarrow A[K]$ y $K \leftarrow K - 1$

1.2 { Fin del ciclo del paso 1.1 }

Hacer $A[K + 1] \leftarrow AUX$

2. { Fin del ciclo del paso 1 }

$i=4$

Desde $i=2$ hasta 8

$AUX = 16$

$K = 3$

Mientras $3 \geq 1$ y $16 < 67$

$A[4]=67$

$K=2$

$A[3]=16$

A: 08, 15, 16, 67, 44, 27, 12, 35

Métodos de ordenamiento

- Ordenación por **Selección directa**.
- Es el más eficiente.
- No se recomienda utilizarlo cuando el número de elementos es mediano o grande.

¿Cómo funciona?

- Busca el menor elemento del arreglo y lo coloca en la primera posición.
- Posteriormente busca el segundo elemento más pequeño y lo coloca en la segunda posición.
- El proceso continúa hasta que todos los elementos del arreglo se encuentren ordenados.

Métodos de ordenamiento

Ordenación por **Selección directa**. Principios

Seleccionar el menor elemento del arreglo.

Intercambiar dicho elemento con el primero.

Repetir los pasos anteriores con los $(n-1)$, $(n-2)$ elementos y así sucesivamente hasta que solo quede el elemento mayor.

Métodos de ordenamiento

- Ordenación por **Selección directa**.

Ejemplo

Deseamos ordenar los siguientes elementos de un arreglo unidimensional A, por el método Selección directa.

A: 15 67 08 16 44 27 12 35

Las comparaciones que se realizan son:

Métodos de ordenamiento

- Ordenación por **Selección directa**.

A: 15 67 08 16 44 27 12 35

PRIMERA PASADA

Se realiza la asignación: $MENOR \leftarrow A[1]$ (15)

($MENOR < A[2]$) (15 < 67) sí se cumple la condición

($MENOR < A[3]$) (15 < 08) no se cumple la condición

$MENOR \leftarrow A[3]$ (8)

($MENOR < A[4]$) (08 < 16) sí se cumple la condición


($MENOR < A[5]$) (08 < 44) sí se cumple la condición

($MENOR < A[6]$) (08 < 27) sí se cumple la condición

($MENOR < A[7]$) (08 < 12) sí se cumple la condición

($MENOR < A[8]$) (08 < 35) sí se cumple la condición

El menor elemento $A[3]$, el 8, se cambió por el primero elemento $A[1]$ el 15.

 A: 08 67 15 16 44 27 12 35

Métodos de ordenamiento

- Ordenación por **Selección directa**.

A: 08 67 15 16 44 27 12 35

SEGUNDA PASADA

Se realiza la siguiente asignación: $MENOR \leftarrow A[2](67)$

($MENOR < A[3]$) ($67 < 15$) no se cumple la condición
 $MENOR \leftarrow A[3](15)$

($MENOR < A[4]$) ($15 < 16$) sí se cumple la condición


($MENOR < A[5]$) ($15 < 44$) sí se cumple la condición

($MENOR < A[6]$) ($15 < 27$) sí se cumple la condición

($MENOR < A[7]$) ($15 < 12$) no se cumple la condición
 $MENOR \leftarrow A[7](12)$

($MENOR < A[8]$) ($12 < 35$) sí se cumple la condición

El segundo elemento menor $A[7]$, el 12, se cambió por el segundo elemento $A[2]$ el 67.

 A: 08 12 15 16 44 27 67 35

Métodos de ordenamiento

- Ordenación por **Selección directa**.

A: 08 12 15 16 44 27 67 35

RESULTADOS DE LAS PASADAS RESTANTES

3a. pasada:	08	12	15	16	44	27	67	35
4a. pasada:	08	12	15	16	44	27	67	35
5a. pasada:	08	12	15	16	27	44	67	35
6a. pasada:	08	12	15	16	27	35	67	44
7a. pasada:	08	12	15	16	27	35	44	67

Métodos de ordenamiento

- Ordenación por **Selección directa**.

Algoritmo `seleccion_directa`. Algoritmo que ordena los elementos de un arreglo haciendo uso del método por selección directa. A es un arreglo de tipo unidimensional de N elementos.

1. *Repetir con I desde 1 hasta $N - 1$*
 - Hacer $MENOR \leftarrow A[I]$ y $K \leftarrow I$
 - 1.1 *Repetir con J desde $I + 1$ hasta N*
 - 1.1.1 *Si $(A[J] < MENOR)$ entonces*
 - Hacer $MENOR \leftarrow A[J]$ y $K \leftarrow J$
 - 1.1.2 {Fin del condicional del paso 1.1.1}
 - 1.2 {Fin del ciclo del paso 1.1}
 - Hacer $A[K] \leftarrow A[I]$ y $A[I] \leftarrow MENOR$
2. {Fin del ciclo del paso 1}

Métodos de ordenamiento

- Actividad:

Elabora la prueba de escritorio para el siguiente algoritmo, que corresponde al método de ordenación por **selección directa**, con el siguiente arreglo:

1. *Repetir* con I desde 1 hasta $N - 1$

A:15, 67, 08, 16, 44, 27, 12, 35

Hacer $MENOR \leftarrow A[I]$ y $K \leftarrow I$

1.1 *Repetir* con J desde $I + 1$ hasta N

1.1.1 Si $(A[J] < MENOR)$ entonces

Hacer $MENOR \leftarrow A[J]$ y $K \leftarrow J$

1.1.2 {Fin del condicional del paso 1.1.1}

1.2 {Fin del ciclo del paso 1.1}

Hacer $A[K] \leftarrow A[I]$ y $A[I] \leftarrow MENOR$

2. {Fin del ciclo del paso 1}

Métodos de ordenamiento

**Los métodos logarítmicos que se abordarán
son:**

Quicksort.

HeapSort.

MergeSort.

Métodos de ordenamiento

- Ordenación Quicksort.
- Es el más eficiente de los métodos de ordenación interna
- También conocido como método rápido u ordenación por partición.
- Es una mejora sustancial al método de intercambio directo.
- Su autor C. A. Hoare lo llamó así por la velocidad con la que ordena los elementos.

Métodos de ordenamiento

- Ordenación Quicksort.

¿Cómo funciona?

1. Se toma un elemento x de cualquier posición en el arreglo.
2. Se trata de ubicar a x en la posición correcta del arreglo, de tal forma que todos los elementos a su izquierda sean menores o iguales a x y todos los de la derecha, mayores o iguales a x .
3. Se repiten los pasos anteriores pero ahora con los conjuntos de datos que se encuentran a la izquierda y a la derecha de la posición en el arreglo de x .
4. El proceso termina cuando todos los elementos se encuentran en su posición correcta en el arreglo.

Métodos de ordenamiento

- Ordenación Quicksort.

Ejemplo:

El siguiente arreglo se ordenará por el método Quicksort:

A: 15 67 08 16 44 27 12 35

Se selecciona $A[1]$, es decir $x=15$. Y se realizan las siguientes comparaciones:

Métodos de ordenamiento

- Ordenación Quicksort.

A: 15 67 08 16 44 27 12 35

PRIMERA PASADA

x=15

Recorrido de derecha a izquierda

$A[8] \geq X$	$(35 \geq 15)$	no hay intercambio
$A[7] \geq X$	$(12 \geq 15)$	sí hay intercambio

A: 12 67 08 16 44 27 15 35

Recorrido de izquierda a derecha

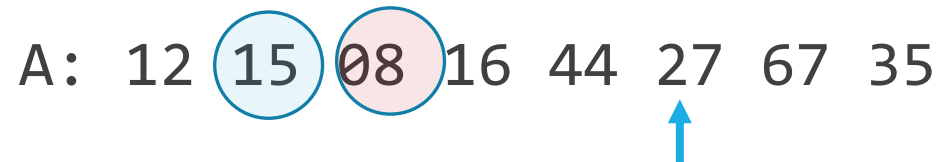
$A[2] \leq X$	$(67 \leq 15)$	sí hay intercambio
---------------	----------------	--------------------

A: 12 15 08 16 44 27 67 35

Métodos de ordenamiento

- Ordenación Quicksort.

A: 12 15 08 16 44 27 67 35



SEGUNDA PASADA

Recorrido de derecha a izquierda

$A[6] \geq X$	$(27 \geq 15)$	no hay intercambio
$A[5] \geq X$	$(44 \geq 15)$	no hay intercambio
$A[4] \geq X$	$(16 \geq 15)$	no hay intercambio
$A[3] \geq X$	$(08 \geq 15)$	sí hay intercambio

A: 12 08 15 16 44 27 67 35

Métodos de ordenamiento

- Ordenación Quicksort.

A: 12 08 15 16 44 27 67 35

El recorrido de izquierda a derecha debería iniciar en la misma posición en la que se encuentra x, el proceso termina, ya que se detecta que x se encuentra en la posición correcta

A: 12 08 15 16 44 27 67 35

1er. conjunto 2o. conjunto

Métodos de ordenamiento

- Ordenación Quicksort.

A: 12 08 15 16 44 27 67 35

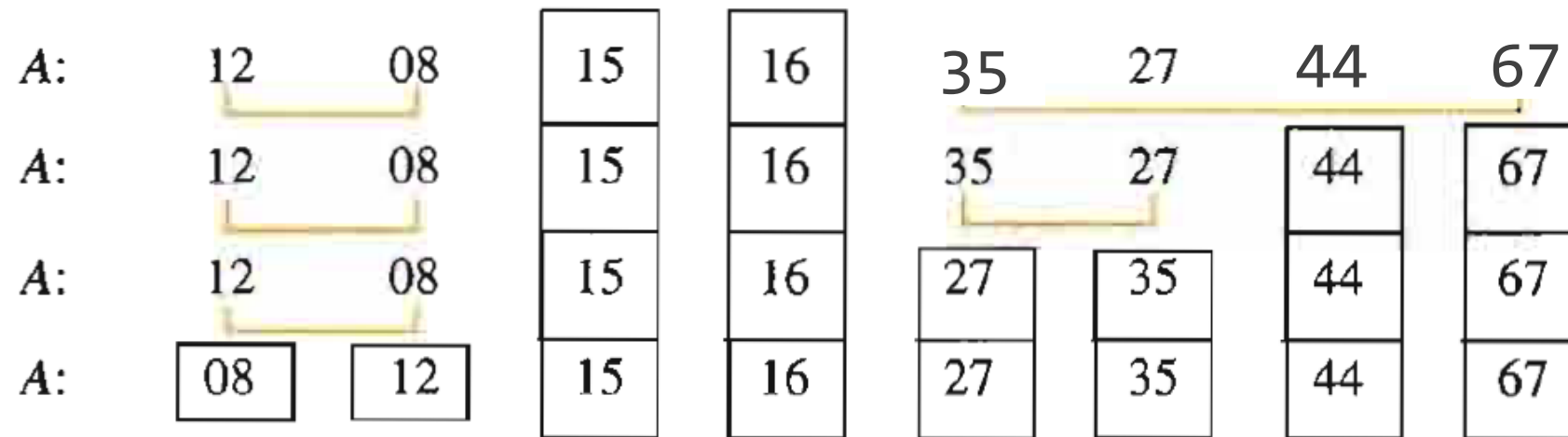
1er. conjunto 2o. conjunto

Este proceso de particionamiento que se aplica para encontrar la posición ideal de x en el arreglo, se repite cada vez que queden conjuntos formados por dos o más elementos.

Métodos de ordenamiento

- Ordenación Quicksort.

Ubicación del resto de los elementos en el arreglo:



Métodos de ordenamiento

Actividad:

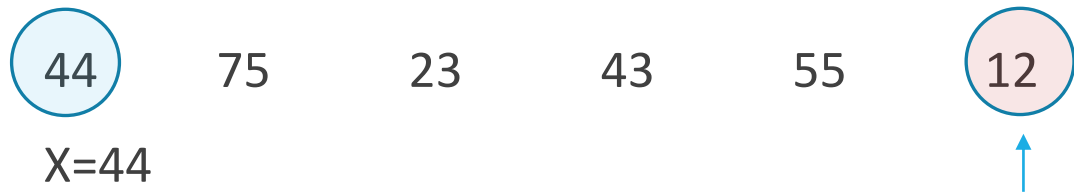
Ordena de manera ascendente los elementos del arreglo unidimensional que se muestran a continuación mediante el método **Quicksort**:

44 75 23 43 55 12

Muestra la organización del arreglo en cada una de las pasadas.

Métodos de ordenamiento

Actividad:

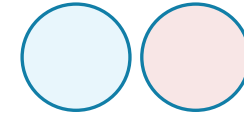


PRIMERA PASADA

RECORRIDO DE DERECHA A IZQUIERDA

$A[6] \geq X$ $12 \geq 44$ Sí hay intercambio

12 75 23 43 55 44



Métodos de ordenamiento

Actividad:

X=44

PRIMERA PASADA

12 75 23 43 55 44
 ↑

RECORRIDO DE IZQUIERDA A DERECHA

$A[2] \leq X$ $75 \leq 44$ Sí hay intercambio

12 44 23 43 55 75

Métodos de ordenamiento

Actividad:

$X=44$

SEGUNDA PASADA

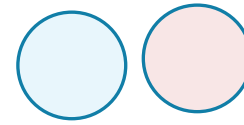
12 44 23 43 55 75

RECORRIDO DE DERECHA A IZQUIERDA

$A[5] \geq X$ $55 \geq 44$ No hay intercambio

$A[4] \geq X$ $43 \geq 44$ Si hay intercambio

12 43 23 44 55 75



Métodos de ordenamiento

Actividad:

$X=44$

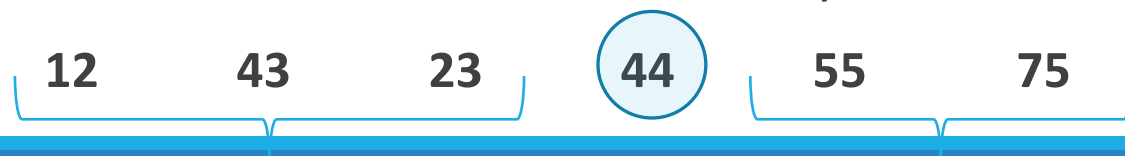
SEGUNDA PASADA

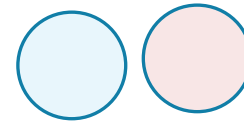
12 43 23 44 55 75
 ↑

RECORRIDO DE IZQUIERDA A DERECHA

$A[3] \leq X$ $23 \geq 44$ No hay intercambio

Al avanzar a la siguiente posición, vemos que llegamos a X y que el subconjunto a su izquierda son los elementos con valor menor y a la derecha con valor mayor.



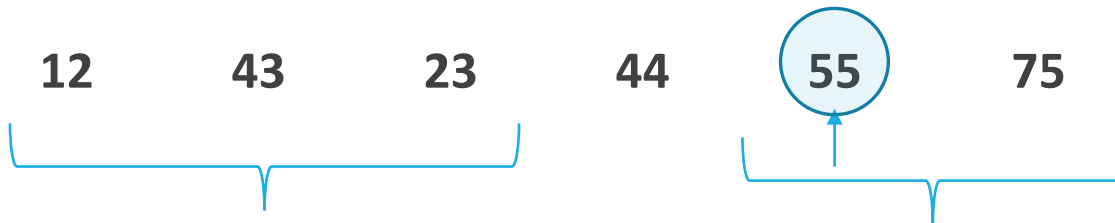


Métodos de ordenamiento

Actividad:

$X=55$

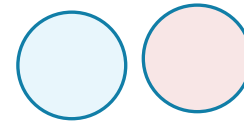
PRIMERA PASADA



RECORRIDO DE DERECHA A IZQUIERDA

$A[6] \geq X$ $75 \geq 55$ No hay intercambio



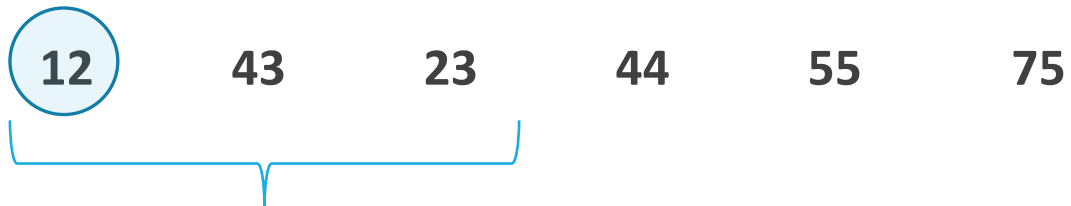


Métodos de ordenamiento

Actividad:

$X=12$

PRIMERA PASADA

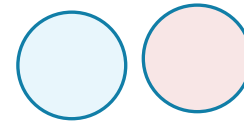


RECORRIDO DE DERECHA A IZQUIERDA

$A[3] \geq X$ $23 \geq 12$ No hay intercambio

$A[2] \geq X$ $43 \geq 12$ No hay intercambio

Llegamos a X por lo que no hay más elementos que ordenar



Métodos de ordenamiento

Actividad:

$X=43$

PRIMERA PASADA

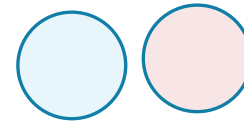
12 43 23 44 55 75

RECORRIDO DE DERECHA A IZQUIERDA

$A[3] \geq X$ $23 \geq 43$ Sí hay intercambio

12 23 43 44 55 75

Llegamos a X por lo que no hay más elementos que ordenar



Métodos de ordenamiento

Actividad:

$X=43$

PRIMERA PASADA

12 23 **43** 44 55 75

RECORRIDO DE IZQUIERDA A DERECHA

12 23 43 44 55 75



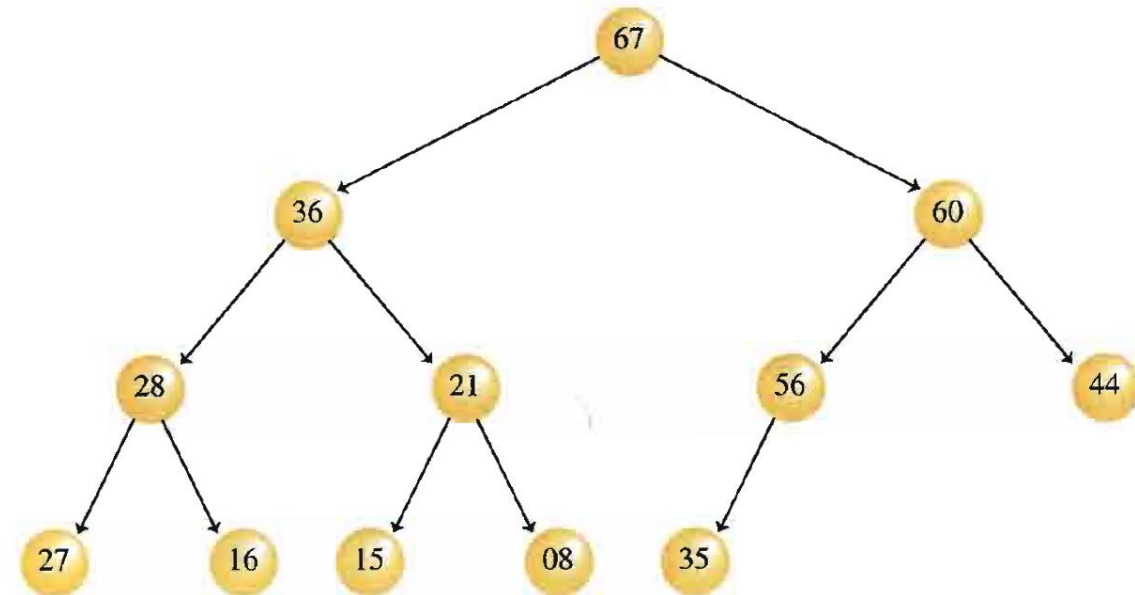
Llegamos a X por lo que no hay más elementos que ordenar

Métodos de ordenamiento

- Ordenación Heapsort.
- También se le conoce como montículo. Nombre dado por su autor J. W. Williams.
- Es el más eficiente de los métodos de ordenación que trabajan con árboles.

Métodos de ordenamiento

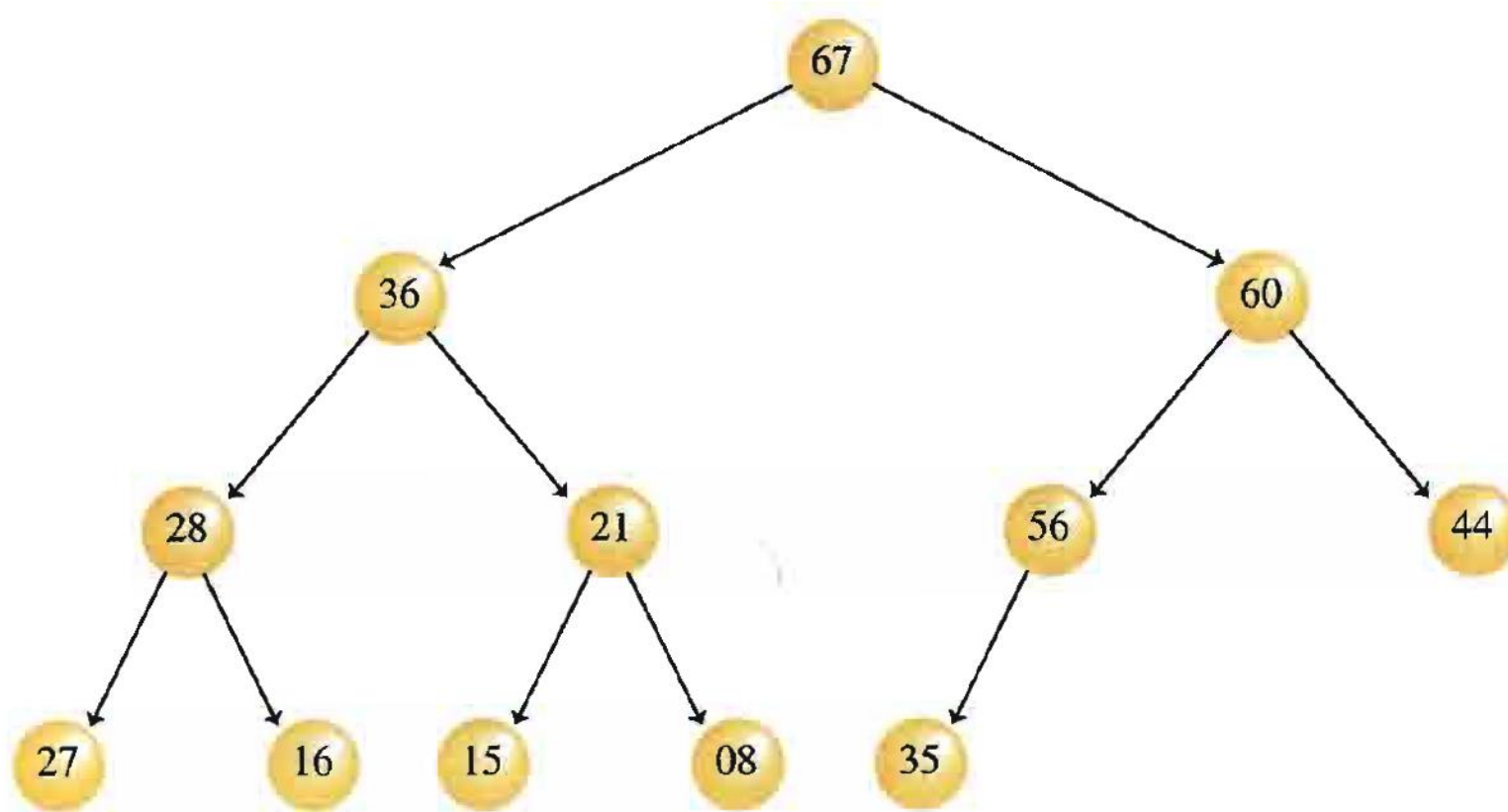
- Ordenación Heapsort.
- **¿Cómo funciona?**
- La idea central se basa en dos operaciones:
 - Construir un montículo.
 - Eliminar la raíz del montículo en forma repetida.
- Y un montículo se define como: Para todo nodo en el árbol se debe cumplir que su valor sea mayor o igual que el valor de cualquiera de sus hijos.



Métodos de ordenamiento

- Ordenación Heapsort.
- **¿Cómo funciona?**
- Para representar un montículo en un arreglo lineal se debe tener en cuenta para todo nodo k lo siguiente:
 - El nodo k se almacena en la posición k correspondiente en el arreglo.
 - El hijo izquierdo del nodo k se almacena en la posición $2 * k$.
 - El hijo derecho del nodo k se almacena en la posición $2 * k + 1$.

- Ordenación Heapsort.
- Representación de un montículo en un arreglo:



A	67	36	60	28	21	56	44	27	16	15	08	35
	1	2	3	4	5	6	7	8	9	10	11	12

Métodos de ordenamiento

- Ordenación Heapsort.
- Por otro lado, es posible encontrar el padre de un nodo no raíz k tomando la parte entera de k entre 2. Por ejemplo, si se desea obtener el padre del nodo $A[11]$, se haría $A[\text{parte entera } (11/2)] = A[5]$.

A	67	36	60	28	21	56	44	27	16	15	08	35
	1	2	3	4	5	6	7	8	9	10	11	12

Métodos de ordenamiento

- Ordenación Heapsort.
- Para insertar un nuevo elemento en el montículo se toman en cuenta los siguientes pasos:
 - Se inserta el elemento en la primera posición disponible.
 - Se verifica si su valor es mayor que el de su padre; si es así, se realiza un intercambio. Si no, el algoritmo se detiene y el elemento queda ubicado en la posición correcta.

Por ejemplo:

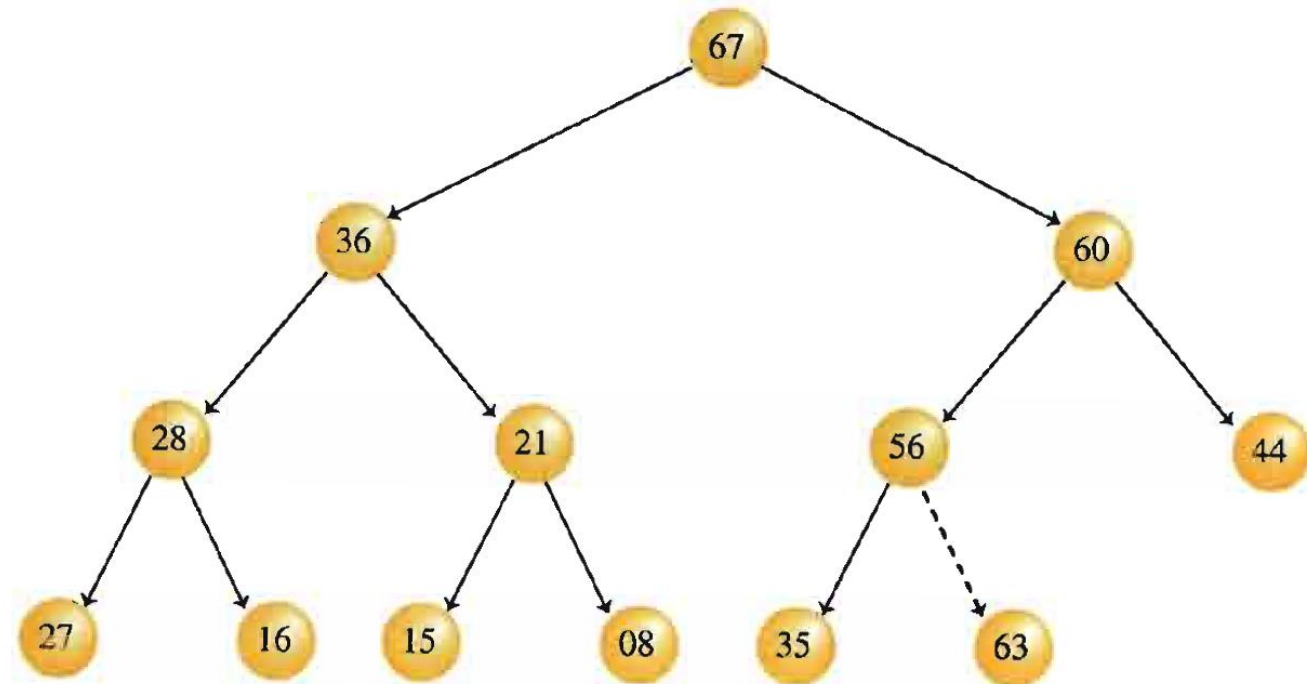
Métodos de ordenamiento

- Ordenación Heapsort.

INSERCIÓN DE ELEMENTOS EN EL ÁRBOL

- Se desea agregar al siguiente montículo el elemento 63:

- $63 > 56$ sí hay intercambio
- $63 > 60$ sí hay intercambio
- $63 > 67$ no hay intercambio

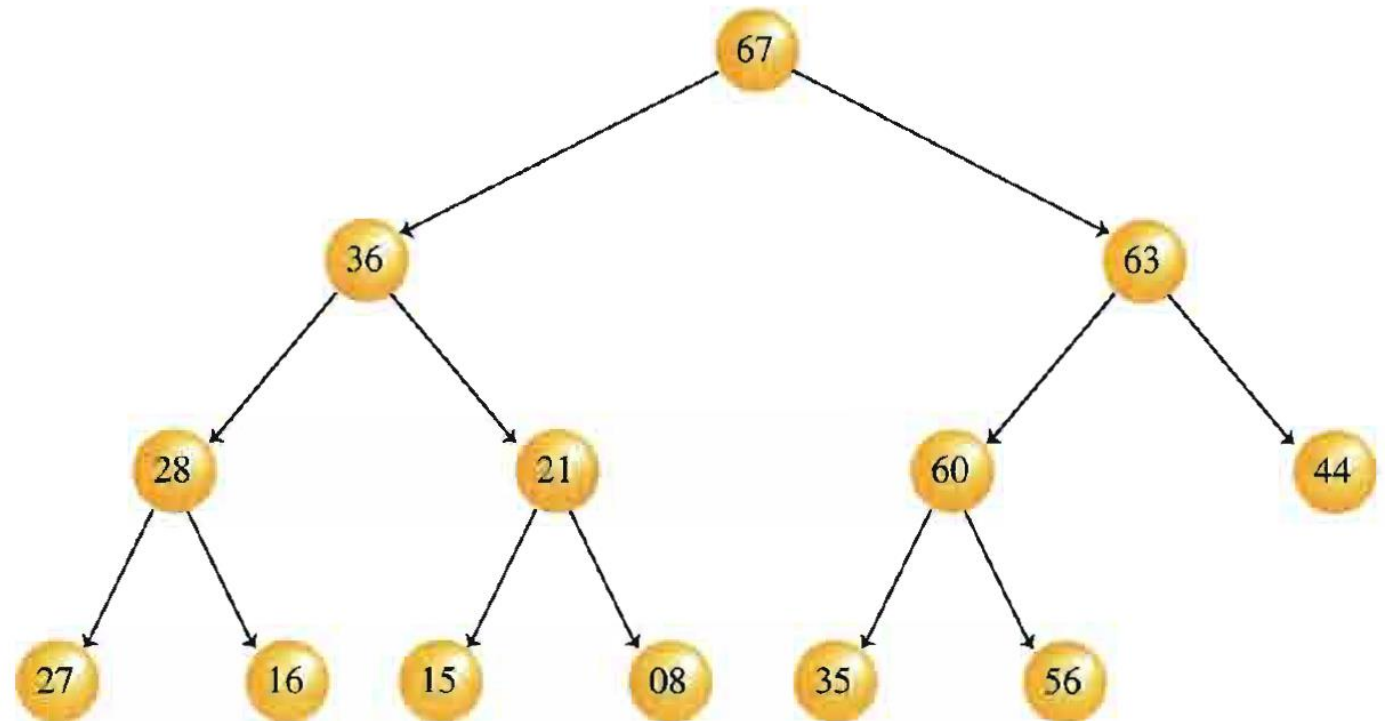


Métodos de ordenamiento

- Ordenación Heapsort.
- Se desea agregar al siguiente montículo el elemento 63:

- $63 > 56$ sí hay intercambio
- $63 > 60$ sí hay intercambio
- $63 > 67$ no hay intercambio

- Después de hacer el cambio a la posición correcta:



Métodos de ordenamiento

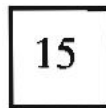
- Ordenación Heapsort.
- Para insertar elementos en un montículo vacío, se analiza el siguiente ejemplo:

Insertar las siguientes claves:

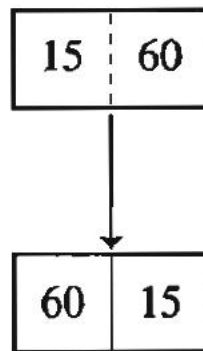
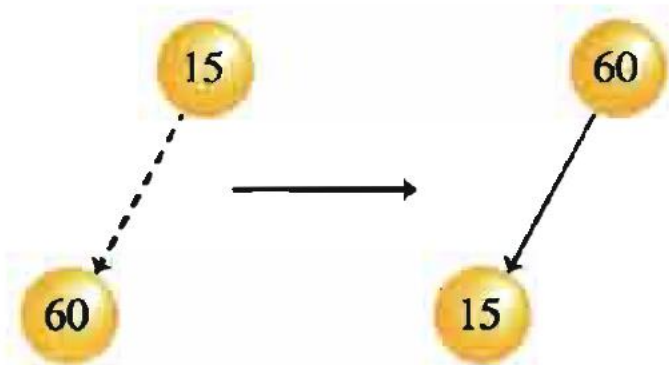
15, 60, 08, 16, 44, 27, 12, 35

15, 60, 08, 16, 44, 27, 12, 35

a) INSERCIÓN: CLAVE 15

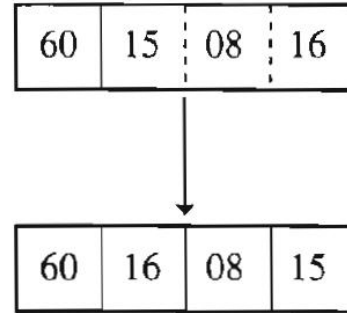
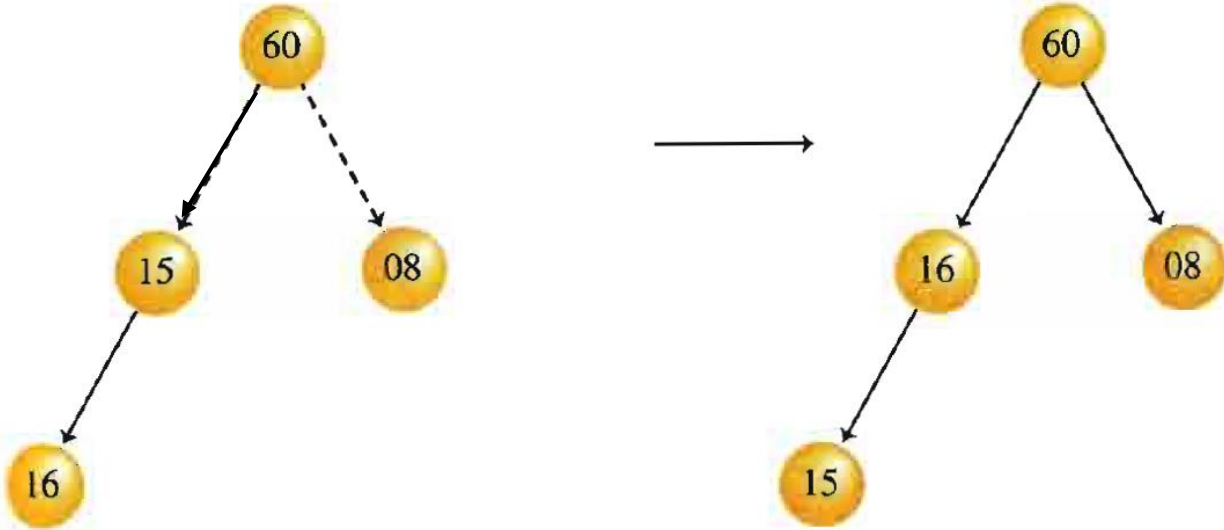


b) INSERCIÓN: CLAVE 60

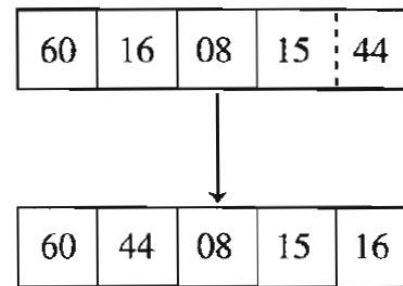
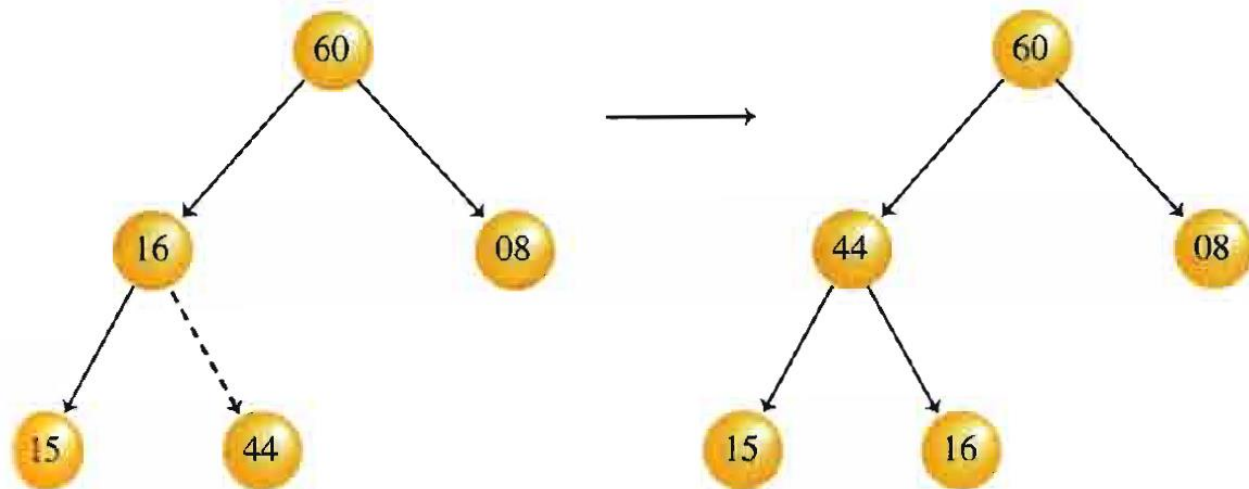


15, 60, 08, 16, 44, 27, 12, 35

c) INSERCIÓN: CLAVES 08 y 16

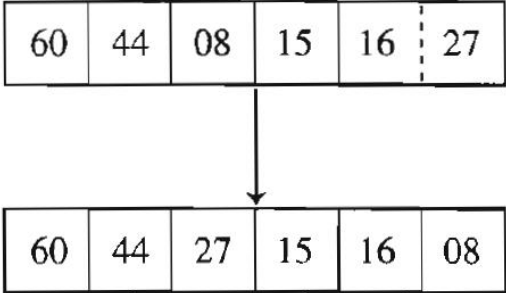
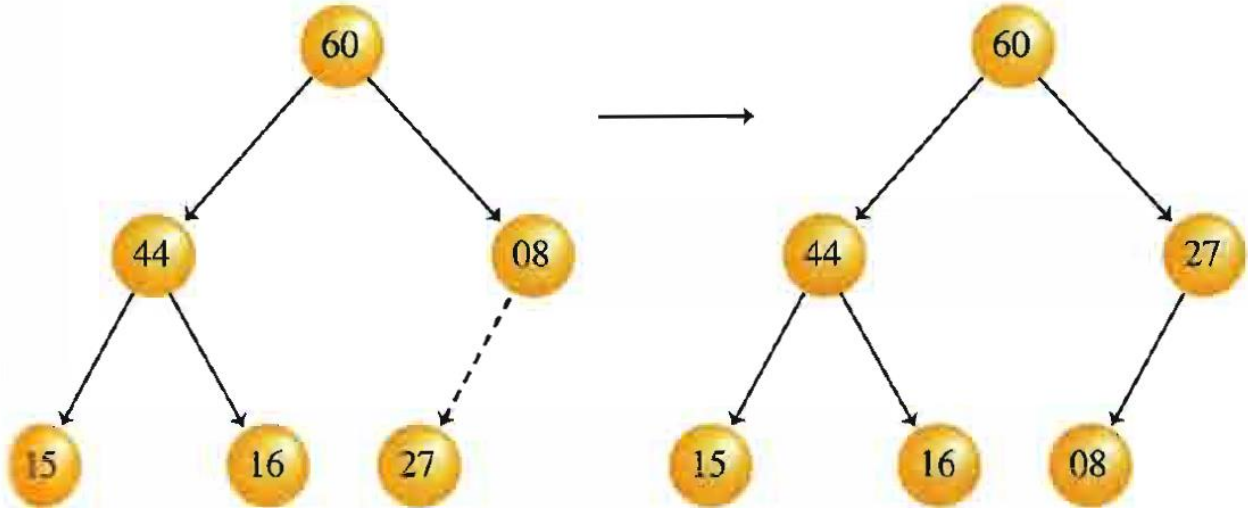


d) INSERCIÓN: CLAVE 44



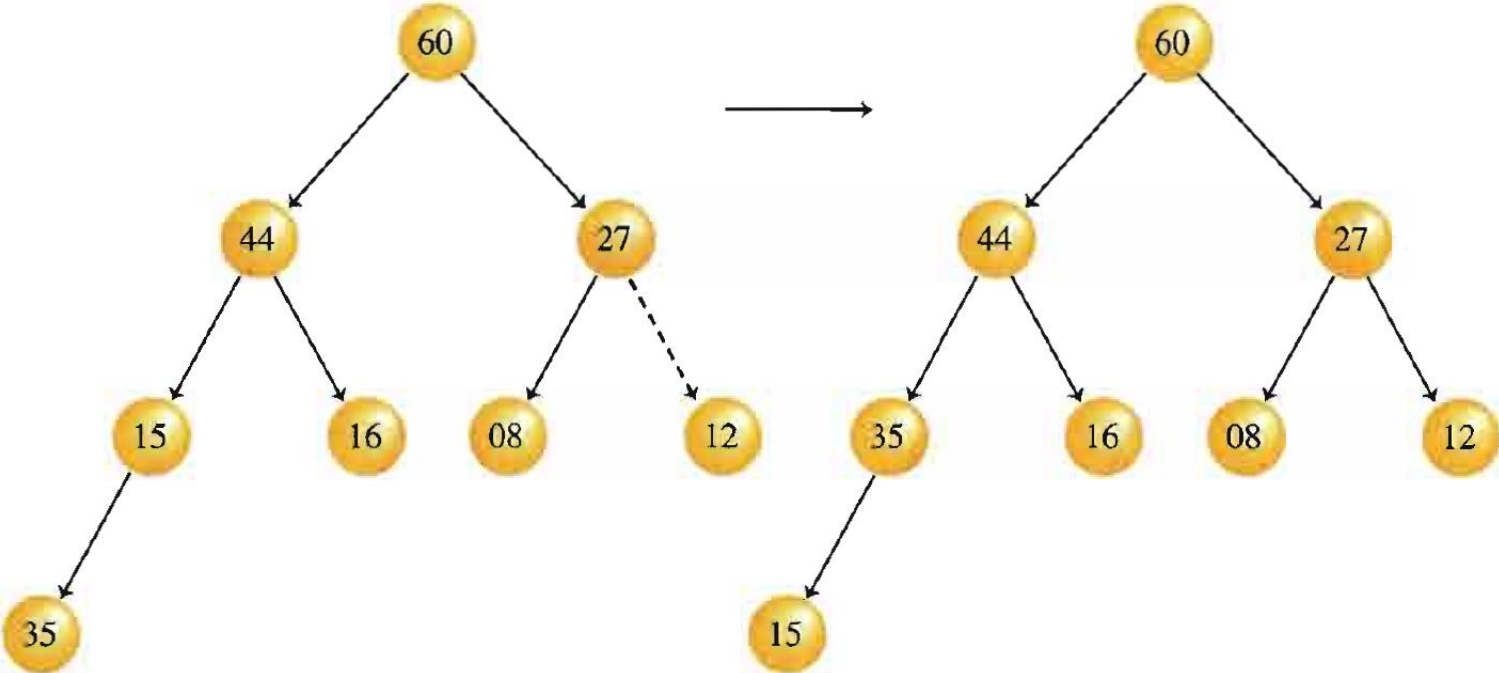
15, 60, 08, 16, 44, 27, 12, 35

INSERCIÓN: CLAVE 27



15, 60, 08, 16, 44, 27, 12, 35

f) INSERCIÓN: CLAVES 12 y 35



60	44	27	15	16	08	12	35
----	----	----	----	----	----	----	----

60	44	27	35	16	08	12	15
----	----	----	----	----	----	----	----

Métodos de ordenamiento

- Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

- El proceso para obtener los elementos ordenados del árbol se efectúa eliminando la raíz del montículo de forma repetida.
- Y los pasos para realizar la eliminación, son los siguientes:
 - Se reemplaza la raíz con el elemento que ocupa la última posición del montículo.
 - Se verifica si el valor de la raíz es menor que el valor más grande de sus hijos, si se cumple la condición, se realiza el intercambio, si no se cumple la condición, el algoritmo se detiene y el elemento queda ubicado en su posición correcta dentro del montículo.

El paso 2 se realiza de manera recursiva.

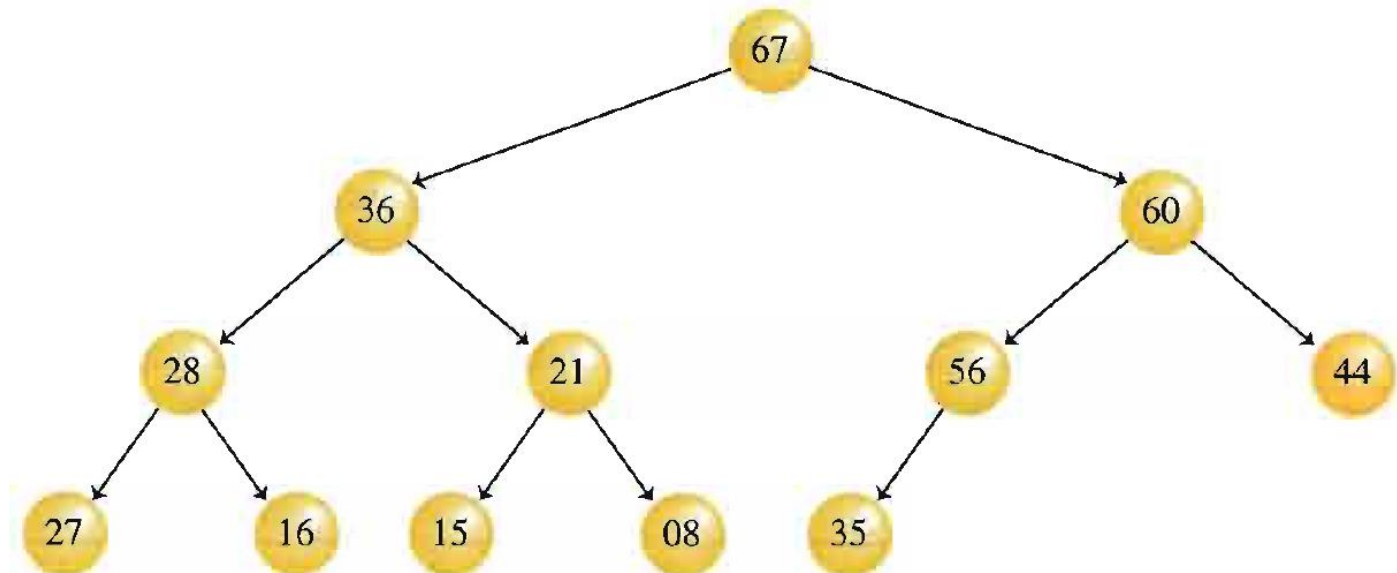
Métodos de ordenamiento

- Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

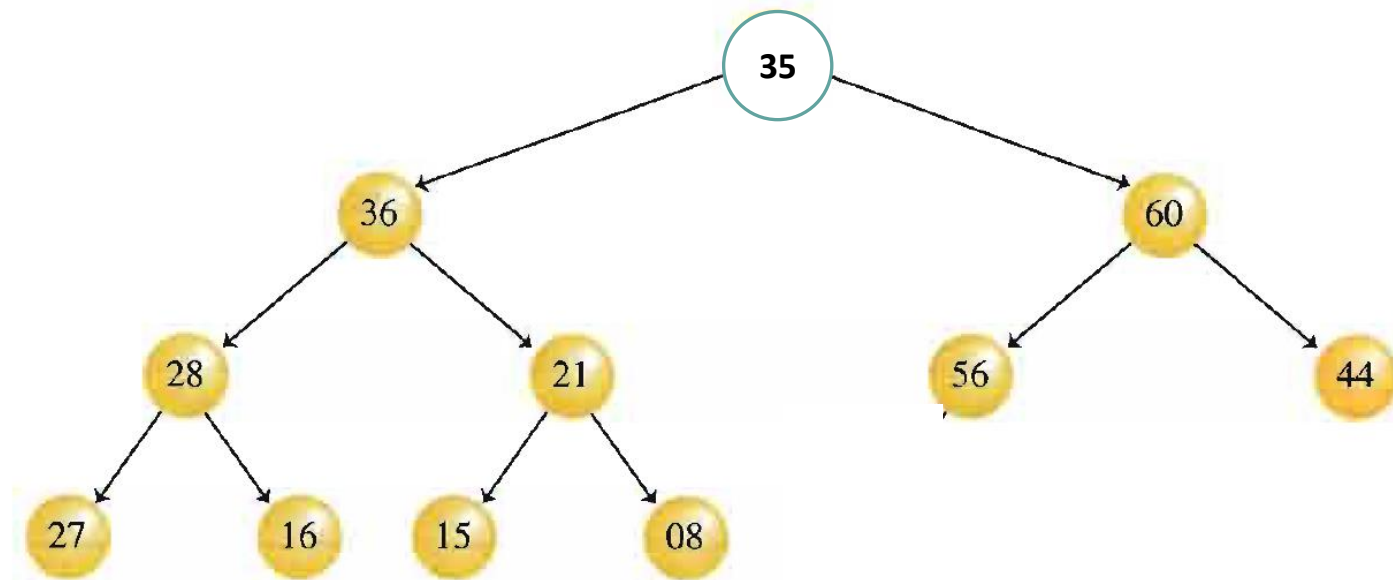
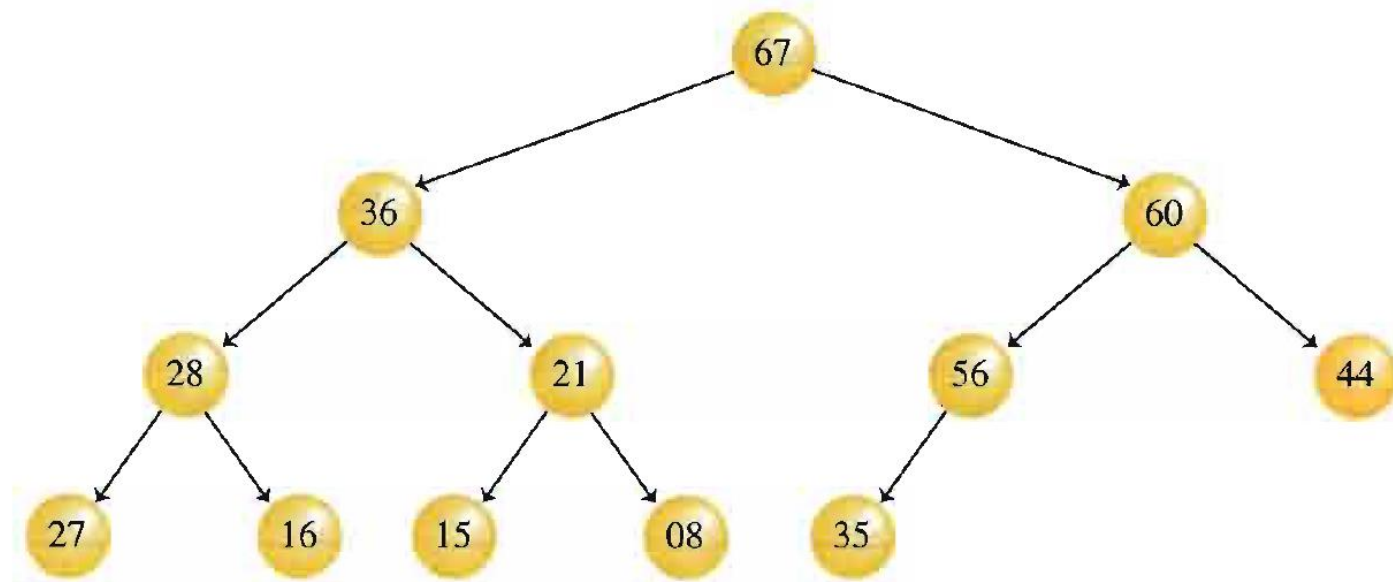
- Ejemplo

- Se desea eliminar la raíz del siguiente montículo (el número 67)



Métodos de ordenamiento

- Ordenación Heapsort.
ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL
- Se reemplaza la raíz por el último elemento del montículo y se realizan las comparaciones:
 - $35 < 60$ Sí, hay intercambio.
 - $35 < 56$ Sí, hay intercambio.



Métodos de ordenamiento

- Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

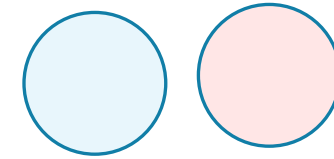
- Ejemplo 2

- Se desea eliminar la raíz, de manera repetida, del siguiente montículo presentado como arreglo:

67	56	60	44	21	28	36	15	35	16	13	08	27	12	07	10
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

- Importante: Al reemplazar la raíz, esta se colocará en la última posición del arreglo; es decir, la primera vez se colocará en la posición n , la segunda en $n-1$ y así sucesivamente.

67	56	60	44	21	28	36	15	35	16	13	08	27	12	07	10
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

PRIMERA ELIMINACIÓN DE LA RAÍZ

- Se intercambia la raíz con el elemento que ocupa la última posición del montículo: el 10 se intercambia con el 67 y las comparaciones son las siguientes:

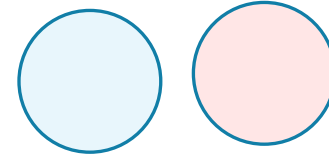
$A[1] < A[3]$ $(10 < 60)$ sí hay intercambio
 $A[3]$ es el mayor de los hijos de $A[1]$

$A[3] < A[7]$ $(10 < 36)$ sí hay intercambio
 $A[7]$ es el mayor de los hijos de $A[3]$

$A[7] < A[14]$ $(10 < 12)$ sí hay intercambio
 $A[14]$ es el mayor de los hijos de $A[7]$

60 56 36 44 21 28 12 15 35 16 13 08 27 10 07 **67**

60	56	36	44	21	28	12	15	35	16	13	08	27	10	60	67
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



- Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

- SEGUNDA ELIMINACIÓN DE LA RAÍZ

- Se intercambia la raíz con el elemento que ocupa la última posición del montículo: el 7 se intercambia con el 60 y las comparaciones son las siguientes:

$A[1] < A[2]$ $(07 < 56)$ sí hay intercambio

$A[2]$ es el mayor de los hijos de $A[1]$

$A[2] < A[4]$ $(07 < 44)$ sí hay intercambio

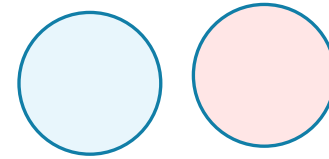
$A[4]$ es el mayor de los hijos de $A[2]$

$A[4] < A[9]$ $(07 < 35)$ sí hay intercambio

$A[9]$ es el mayor de los hijos de $A[4]$

56	44	36	35	21	28	12	15	07	16	13	08	27	10	60	67
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

56	44	36	35	21	28	12	15	07	16	13	08	27	10	60	67
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



- Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

- TERCERA ELIMINACIÓN DE LA RAÍZ

- Se intercambia la raíz con el elemento que ocupa la última posición del montículo: el 10 se intercambia con el 56 y las comparaciones son las siguientes:

$A[1] < A[2]$ $(10 < 44)$ sí hay intercambio

$A[2]$ es el mayor de los hijos de $A[1]$

$A[2] < A[4]$ $(10 < 35)$ sí hay intercambio

$A[4]$ es el mayor de los hijos de $A[2]$

$A[4] < A[8]$ $(10 < 15)$ sí hay intercambio

$A[8]$ es el mayor de los hijos de $A[4]$

44	35	36	15	21	28	12	10	07	16	13	08	27	56	60	67
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

56 44 36 35 21 28 12 15 07 16 13 08 27 10 60 67

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

○ Ordenación Heapsort.

ELIMINACIÓN DE MONTÍCULOS DEL ÁRBOL

○ Al hacer la eliminación de manera repetida, el arreglo queda ordenado:

	Eliminación													Montículo			
4	36	35	28	15	21	27	12	10	07	16	13	08	44	56	60	67	
5	35	21	28	15	16	27	12	10	07	08	13	36	44	56	60	67	
6	28	21	27	15	16	13	12	10	07	08	35	36	44	56	60	67	
7	27	21	13	15	16	08	12	10	07	28	35	36	44	56	60	67	
8	21	16	13	15	07	08	12	10	27	28	35	36	44	56	60	67	
9	16	15	13	10	07	08	12	21	27	28	35	36	44	56	60	67	
10	15	12	13	10	07	08	16	21	27	28	35	36	44	56	60	67	
11	13	12	08	10	07	15	16	21	27	28	35	36	44	56	60	67	
12	12	10	08	07	13	15	16	21	27	28	35	36	44	56	60	67	
13	10	07	08	12	13	15	16	21	27	28	35	36	44	56	60	67	
14	08	07	10	12	13	15	16	21	27	28	35	36	44	56	60	67	
15	07	08	10	12	13	15	16	21	27	28	35	36	44	56	60	67	

Métodos de ordenamiento

- Ordenación Heapsort.

El proceso de ordenación por el método del montículo, consta de dos partes:

Construir el montículo.

Eliminar repetidamente la raíz del montículo.

Métodos de ordenamiento

- Ordenación Mergesort.
- Ordenamiento por combinación o mezcla, es un algoritmo de ordenamiento eficiente. Desarrollado en 1945 por John Von Neumann.

Métodos de ordenamiento

- Ordenación Mergesort.
- **¿Cómo funciona?**
- Para ordenar un vector, el algoritmo de ordenamiento por combinación lo divide en dos sub vectores de igual tamaño, ordena cada sub vector y después los combina en un vector más grande.
- Con un número impar de elementos, el algoritmo crea los dos sub vectores de tal forma que uno tenga más elementos que el otro.
- La implementación es recursiva.

Métodos de ordenamiento

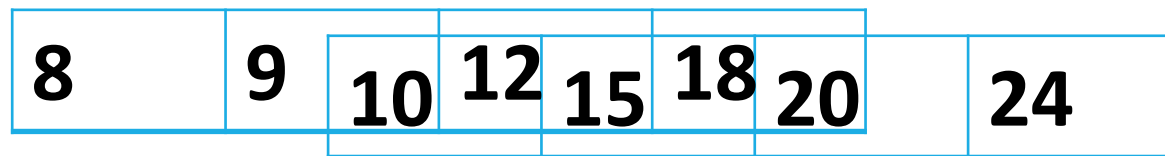
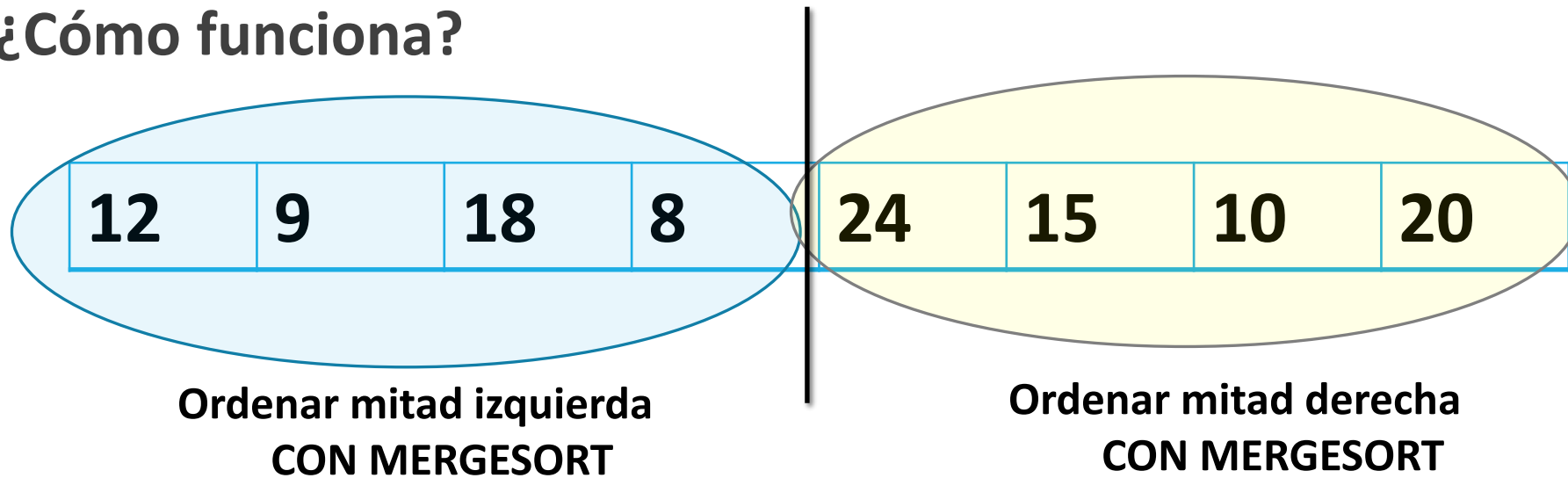
- Ordenación Mergesort.
- **¿Cómo funciona?**
- Para ordenar un vector, el algoritmo de ordenamiento por combinación lo divide en dos sub vectores de igual tamaño, ordena cada sub vector y después los combina en un vector más grande.
- Con un número impar de elementos, el algoritmo crea los dos sub vectores de tal forma que uno tenga más elementos que el otro.
- La implementación es recursiva.

Métodos de ordenamiento

- Ordenación Mergesort.
- **¿Cómo funciona?**
- El paso básico es un vector con un elemento que, desde luego, está ordenado, por lo que el ordenamiento por combinación regresa de inmediato cuando se le llama con un vector de un elemento.
- El paso recursivo divide a un vector de dos o más elementos en dos sub vectores de igual tamaño, ordena en forma recursiva cada sub vector y después los combina en un vector ordenado de mayor tamaño.

Métodos de ordenamiento

- Ordenación Mergesort.
- ¿Cómo funciona?



MEZCLAR

Métodos de ordenamiento

- Ordenación Mergesort.
- **¿Cómo funciona?**

Mergesort(listaElementos)

Mergesort(listalq)

Mergesort(listader)

Mezclar(listalq, listader)

Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?

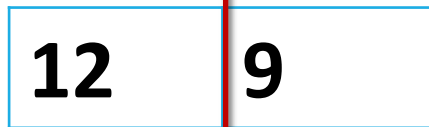


listalzq

Merge(listalzq)



Merge(listalzq)



Merge(listalzq)



Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?



listalq

Merge(listalq)



Merge(listalq)



listaDer

- Mergesort(listaElementos)
- Mergesort(listalq) ✓
- Mergesort(listaDer)
- Mezclar(listalq, listaDer)

Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?



listalq

Merge(listalq)



Merge(listalq) Merge(listaDer)



Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?

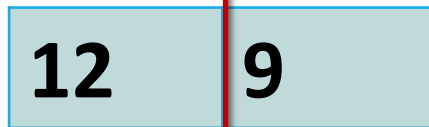


listalq

Merge(listalq)



Merge(listalq) Merge(listaDer)



Mergesort(listaElementos)

Mergesort(listalq)

Mergesort(listaDer) ✓

Mezclar(listalq, listaDer)

Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?

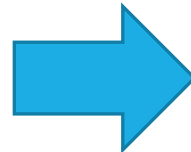
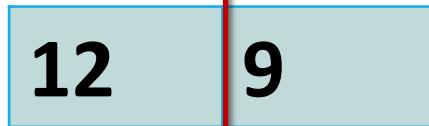


listalq

Merge(listalq)



Merge(listalq) Merge(listaDer)



Mezclar(listalq, listaDer)

Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?

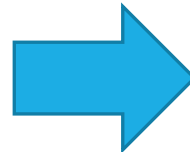
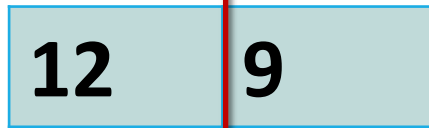


listalq

Merge(listalq)



Merge(listalq) Merge(listaDer)



Mezclar(listalq, listaDer)

Mergesort(listaElementos)

Mergesort(listalq)

Mergesort(listaDer)

Mezclar(listalq, listaDer) ✓

Métodos de ordenamiento

- Ordenación Mergesort. ¿Cómo funciona?



listalq

Merge(listalq)

Merge(listaDer)

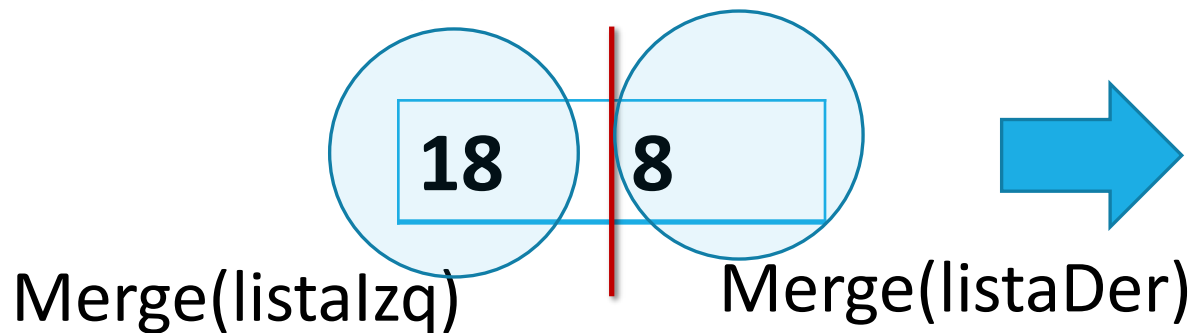
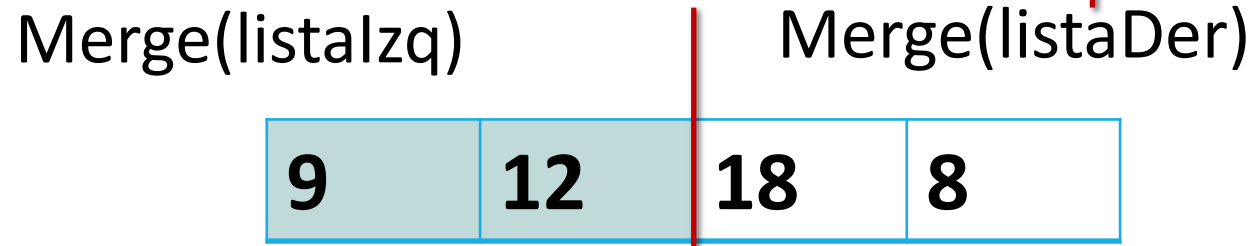
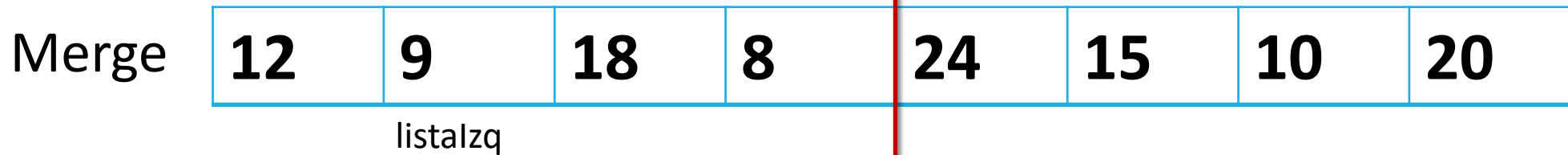


- Mergesort(listaElementos)
- Mergesort(listalq) ✓
- Mergesort(listaDer)
- Mezclar(listalq, listaDer)

Métodos de ordenamiento

Mergesort(listaElementos)
Mergesort(listalq)
Mergesort(listader)
Mezclar(listalq, listader)

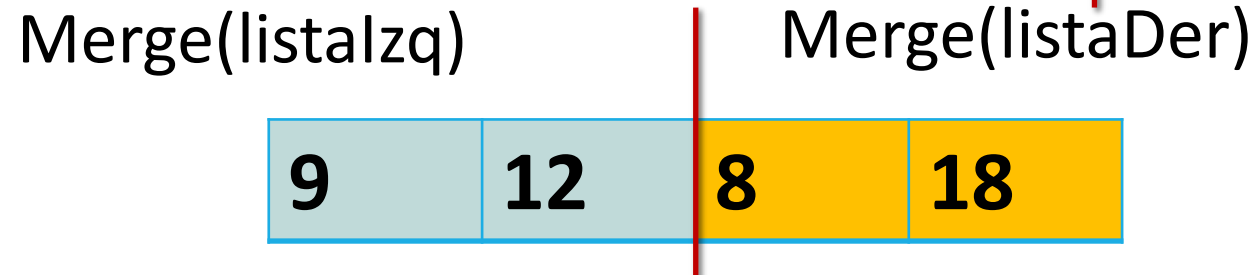
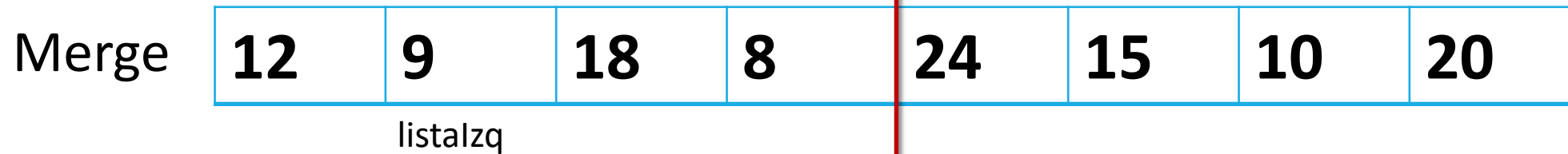
Ordenación Mergesort. ¿Cómo funciona?



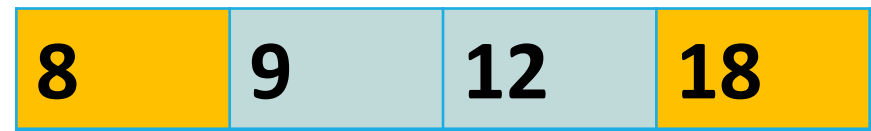
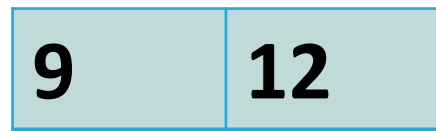
Métodos de ordenamiento

Mergesort(listaElementos)
Mergesort(listalq)
Mergesort(listader)
Mezclar(listalq, listader)

Ordenación Mergesort. ¿Cómo funciona?



Mezclar(listalq, listader)

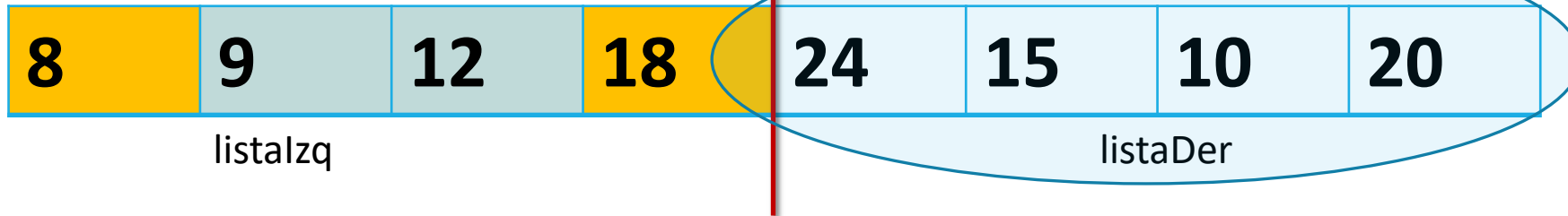


Métodos de ordenamiento

Mergesort(listaElementos)
Mergesort(listalq)
Mergesort(listader)
Mezclar(listalq, listader)

Ordenación Mergesort. ¿Cómo funciona?

Merge



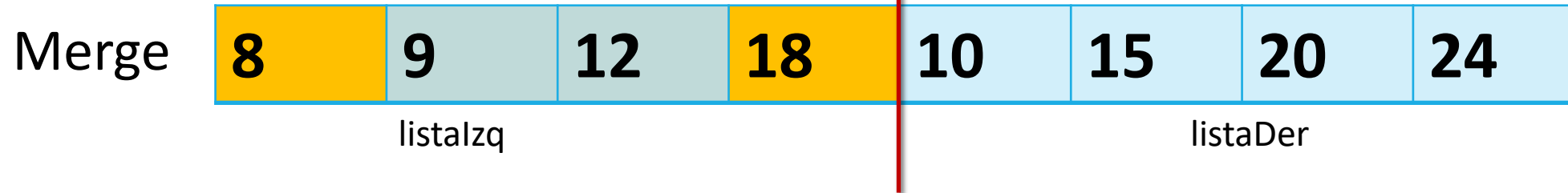
Ordena la lista derecha exactamente igual que la lista izquierda.



Métodos de ordenamiento

Mergesort(listaElementos)
Mergesort(listaIzq)
Mergesort(listaDer)
Mezclar(listaIzq, listaDer)

- Ordenación Mergesort. ¿Cómo funciona?



Una vez ordenada lista izquierda y lista derecha.
Continúa con:
Mezclar(listaIzq, listaDer)

Métodos de ordenamiento

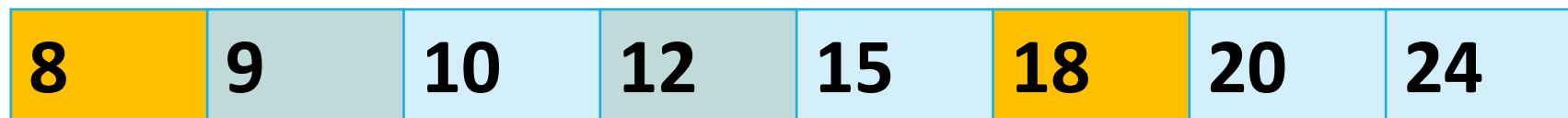
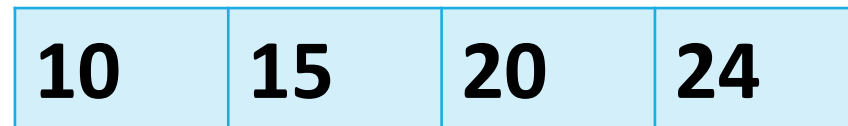
Mergesort(listaElementos)
Mergesort(listalq)
Mergesort(listader)
Mezclar(listalq, listader)

Ordenación Mergesort. ¿Cómo funciona?

Merge



Mezclar(listalq, listaDer)



Bibliografía

Cairó, O. y Guardati, S. (2002). Estructuras de Datos, 2da. Edición. McGraw-Hill.

Deitel P.J. y Deitel H.M. (2008) Cómo programar en C++. 6ª edición. Prentice Hall.

Joyanes, L. (2006). Programación en C++: Algoritmos, Estructuras de datos y objetos. McGraw-Hill.