Estructura de Datos

Maria C. Torres

Maria C. Torres

Ing. Electrónica (UNAL)
M.E. Ing. Eléctrica (UPRM)
Ph.D. Ciencias e Ingeniería de la Computación y la Información (UPRM)
Profesora asociada
Dpto. Ciencias de la Computación y la Decisión

mctorresm@unal.edu.co

HORARIO DE ATENCIÓN: Martes 10:00 am a 12:00 m – Oficina 313 M8A

Contenido del Curso

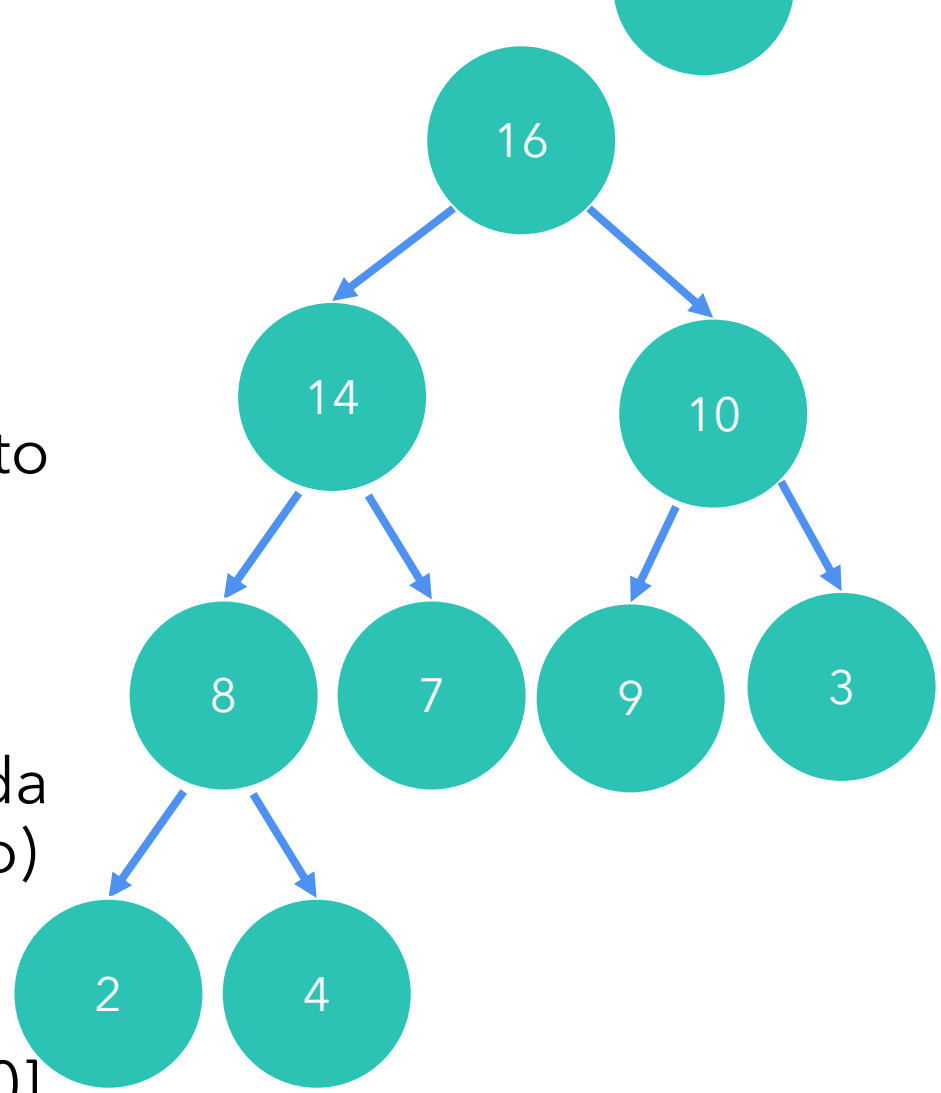
- ❑ Introducción: revisión fundamentos y POO
- ❑ Análisis de complejidad
- ❑ Arreglos
- ❑ Listas enlazadas
- ❑ Pilas y colas
- ❑ **Heap**
- ❑ Arboles binarios
- ❑ Tablas hash
- ❑ Grafos

Heap

- Heap
- Operaciones
- Heapsort
- Análisis de algoritmos
- Aplicaciones

HEAP

- Arreglo de objetos que se puede ver como un árbol binario casi completo
- Cada nodo del árbol corresponde a un elemento en un arreglo
- El árbol está completamente lleno en todos los niveles excepto el ultimo
- El arreglo que representa el HEAP tiene asociada una capacidad (dada por la longitud del arreglo) y un tamaño (determinado por el número de elementos en el HEAP)
- La raíz del HEAP se encuentra en la posición A[0]



16	14	10	8	7	9	3	2	4	

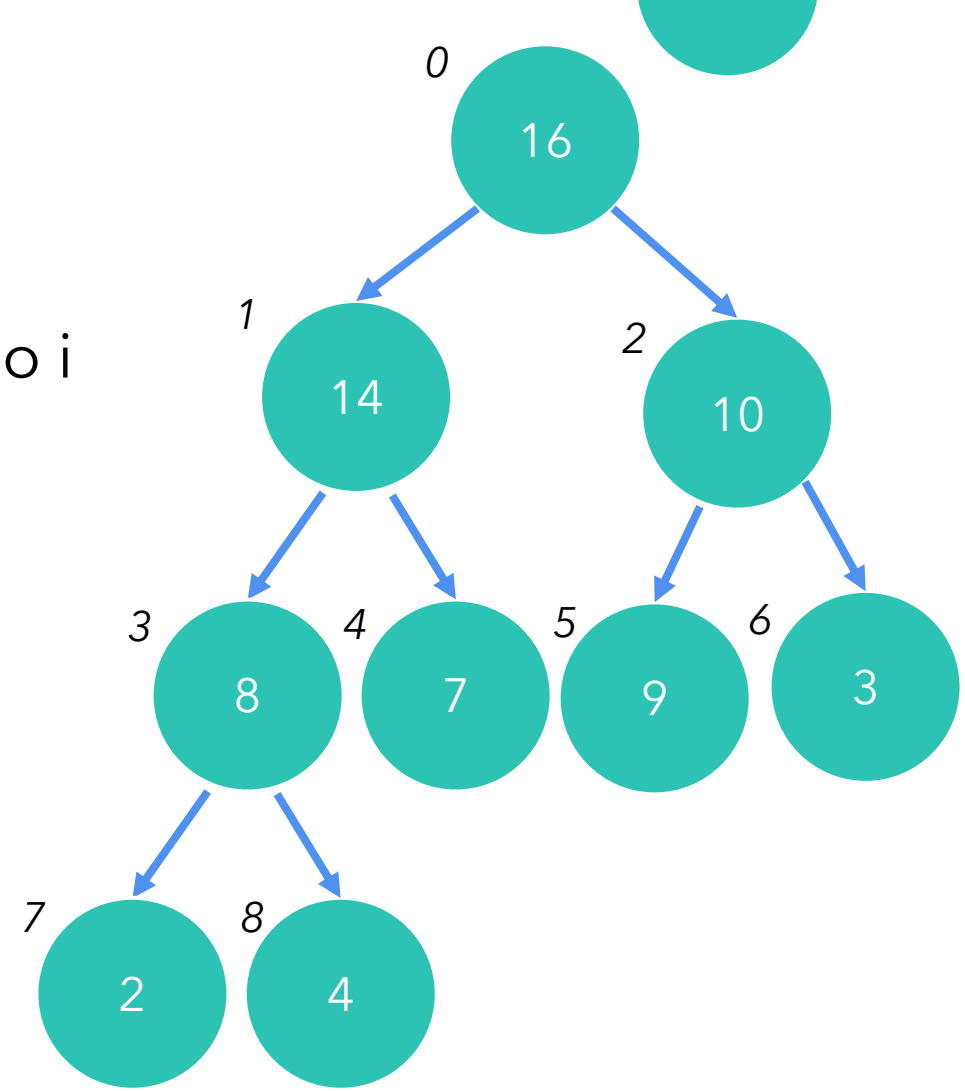
HEAP

- Dada esta estructura, es sencillo determinar el padre, hijo izquierdo e hijo derecho de un nodo i

```
Parent(i)
1. return ⌊i/2⌋−1

Left(i)
1. return 2i+1

Right(i)
1. return 2i+2
```



16	14	10	8	7	9	3	2	4	
0	1	2	3	4	5	6	7	8	9

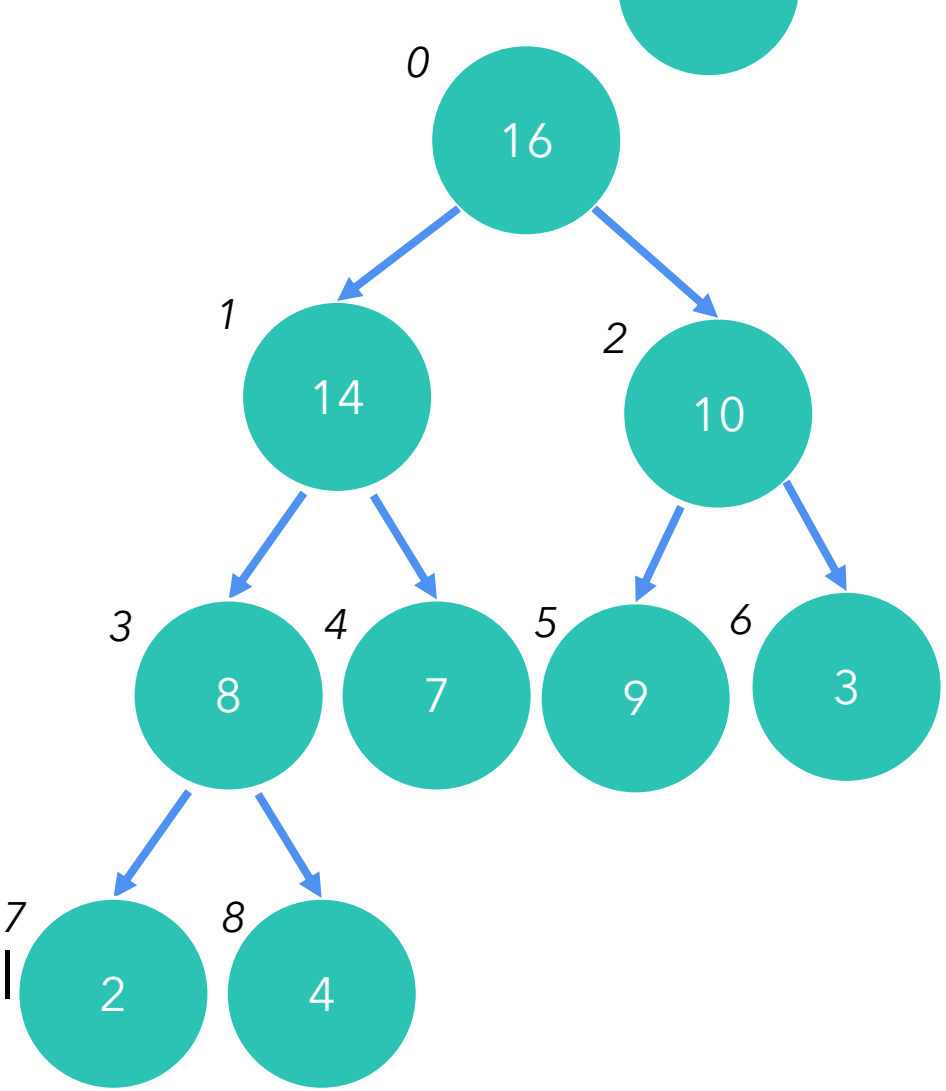
HEAP

Existen dos tipos

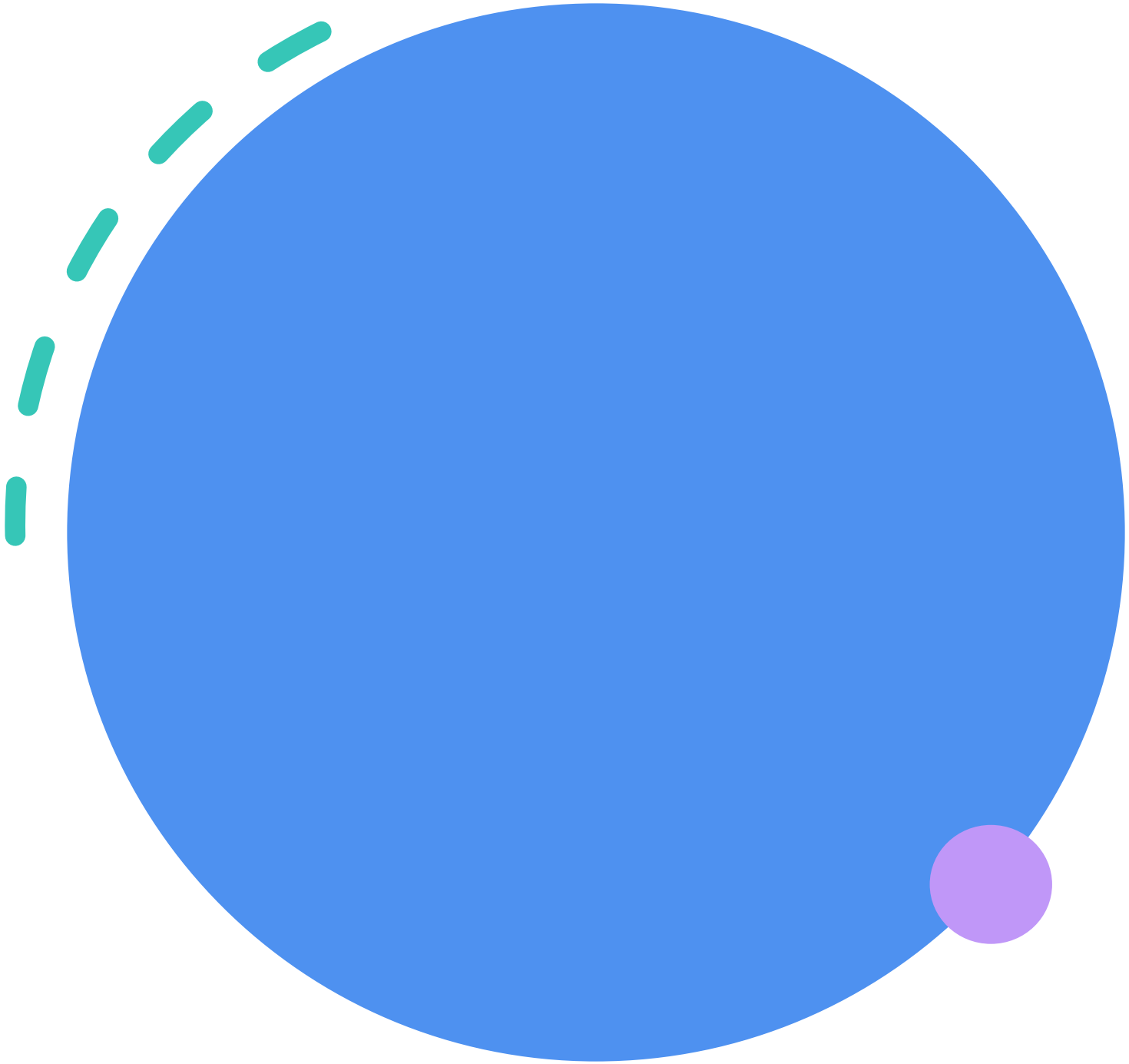
- MAX-HEAP
- MIN-HEAP

❑*MAX-HEAP Propiedad:* Cada nodo i diferente a la raíz, satisface:
  - $A[Parent(i)] \geq A[i]$
  - La raíz contiene el valor máximo

❑*MIN-HEAP Propiedad:* Cada nodo i diferente a l raíz, satisface:
  - $A[Parent(i)] \leq A[i]$
  - La raíz contiene el valor mínimo



16	14	10	8	7	9	3	2	4	
0	1	2	3	4	5	6	7	8	9



Heap

- Heap
- **Operaciones**
- Heapsort
- Análisis de algoritmos
- Aplicaciones

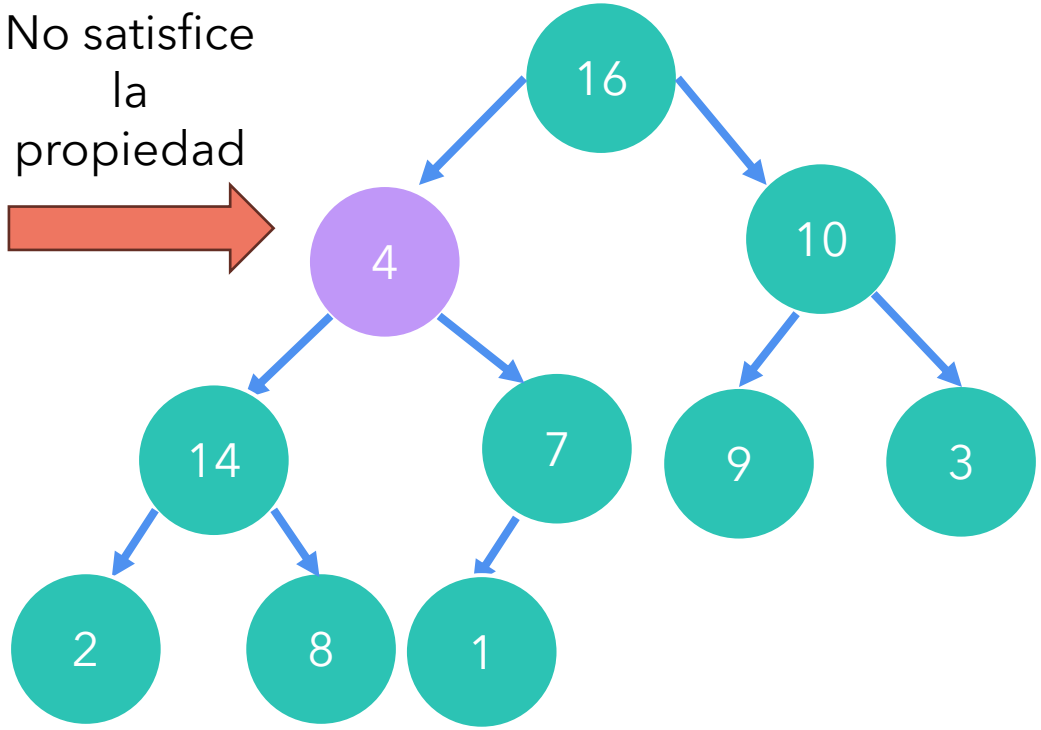
Operaciones de un HEAP

Operación	Descripción	Complejidad
MAX-HEAPIFY	Mantiene la propiedad heap	O(log n)
BUILD-MAX-HEAP	Construye un heap a partir de un arreglo no ordenado	O(n log n)
HEAPSORT	Algoritmo de ordenamiento, organiza los datos en el mismo arreglo	O(n log n)
MAX-HEAP-INSERT	Inserta un dato en el arreglo manteniendo la propiedad HEAP	O(log n)
HEAP-EXTRACT-MAX	Retorna y elimina de la colección el valor máximo	O(log n)
HEAP-MAXIMUM	Retorna el valor máximo	O(1)

Operaciones de un HEAP – MAX-HEAPIFY

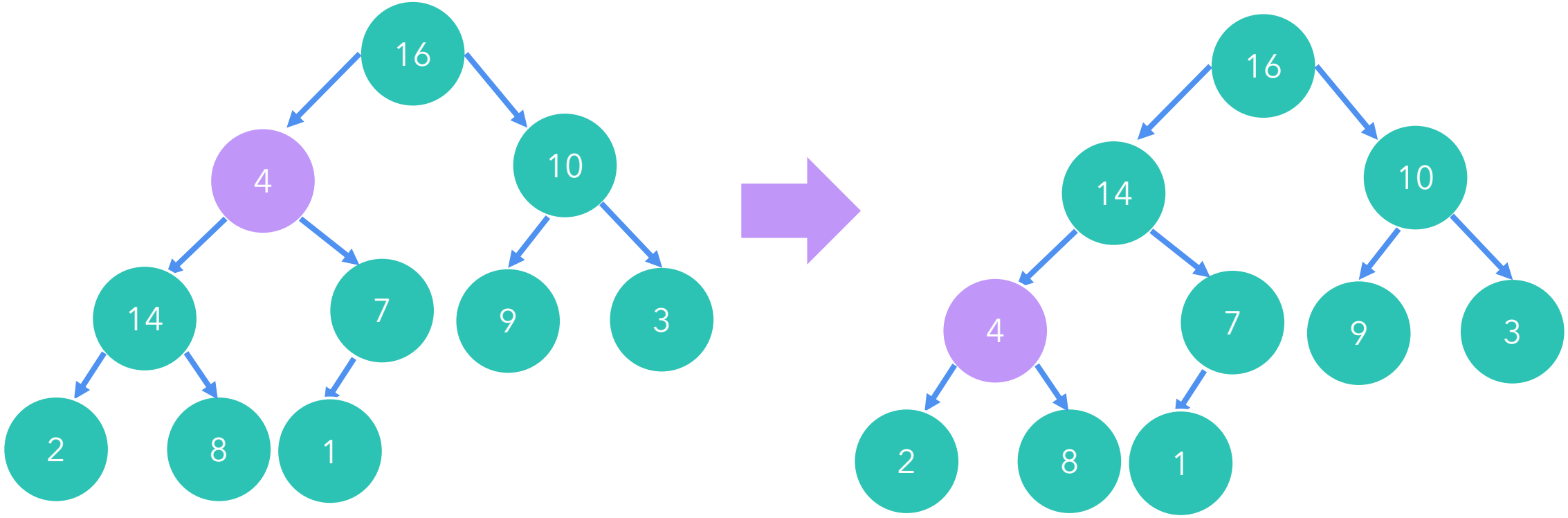
MAX-HEAP Propiedad: Cada nodo i diferente a la raíz, satisface:

$$A[Parent(i)] \geq A[i]$$



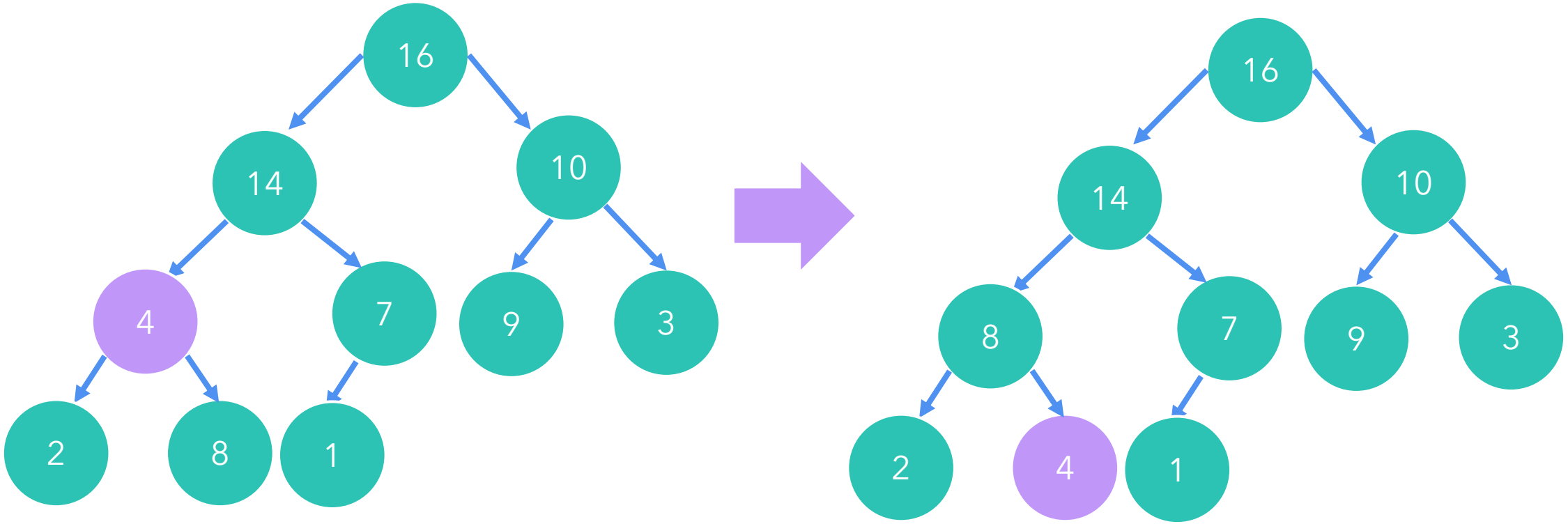
- Dado un nodo MAX-HEAPIFY verifica que se cumpla la propiedad, sino realiza los movimientos de los datos para que se cumpla

Operaciones de un HEAP – MAX-HEAPIFY



- Desplaza el nodo que cumple la propiedad hacia abajo hasta que se ubique de forma correcta

Operaciones de un HEAP – MAX-HEAPIFY



- Desplaza el nodo que cumple la propiedad hacia abajo hasta que se ubique de forma correcta

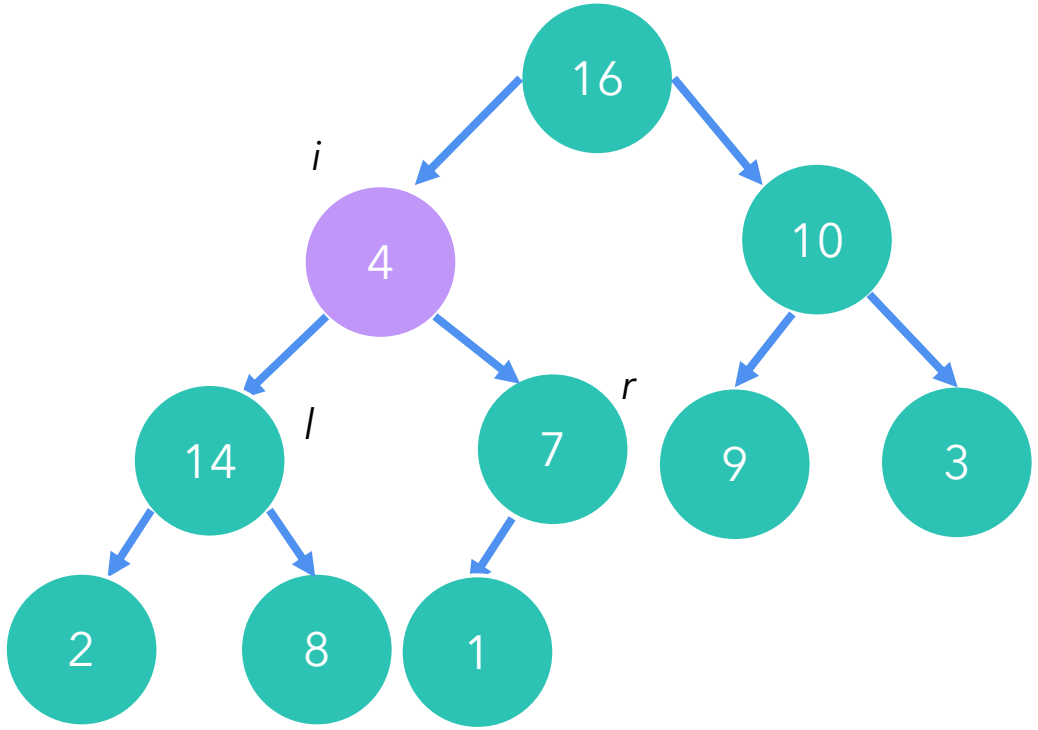
Operaciones de un HEAP – MAX-HEAPIFY

Algoritmo:

1. Obtener el hijo Izquierdo y derecho
2. Determinar el nodo con el valor máximo (raíz, hijo izquierdo, o hijo derecho)
3. Realizar el intercambio de datos si es necesario
4. Repetir hasta que los datos queden organizados

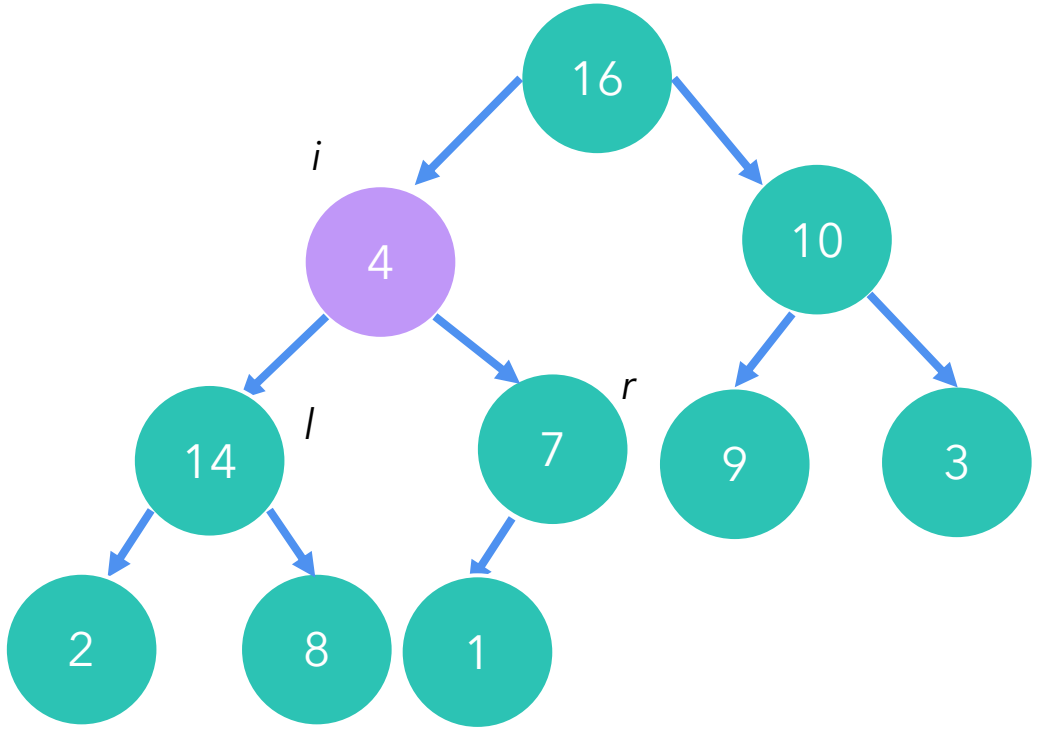
```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

Operaciones de un HEAP – MAX-HEAPIFY




```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

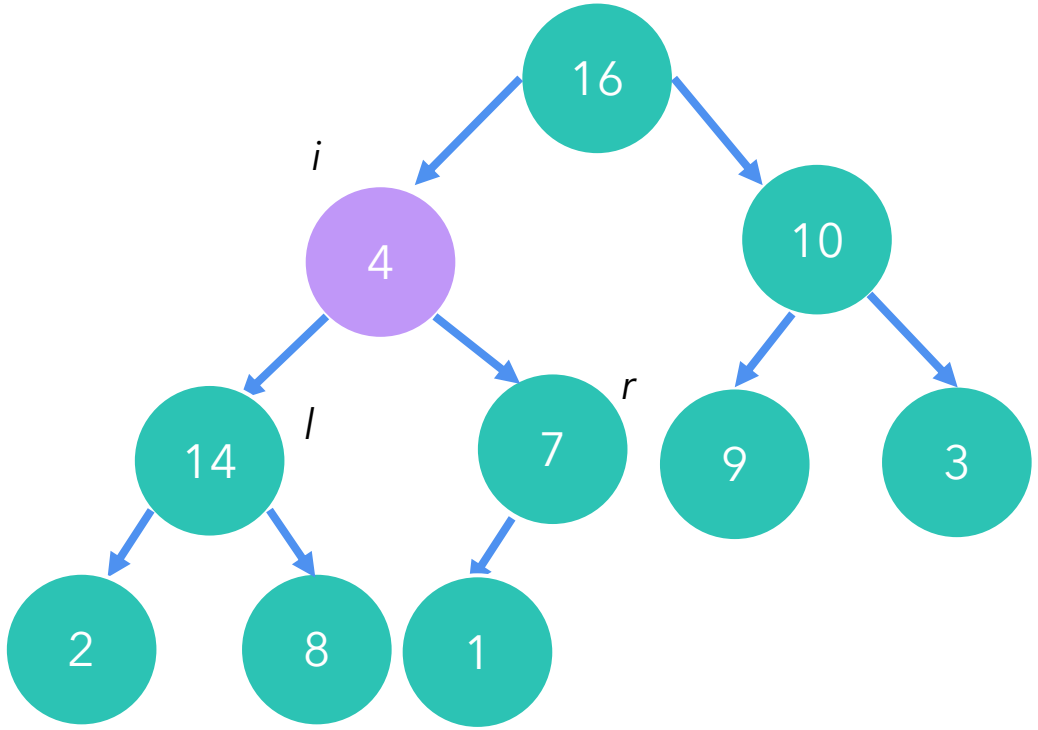
Operaciones de un HEAP – MAX-HEAPIFY



largest = l

```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

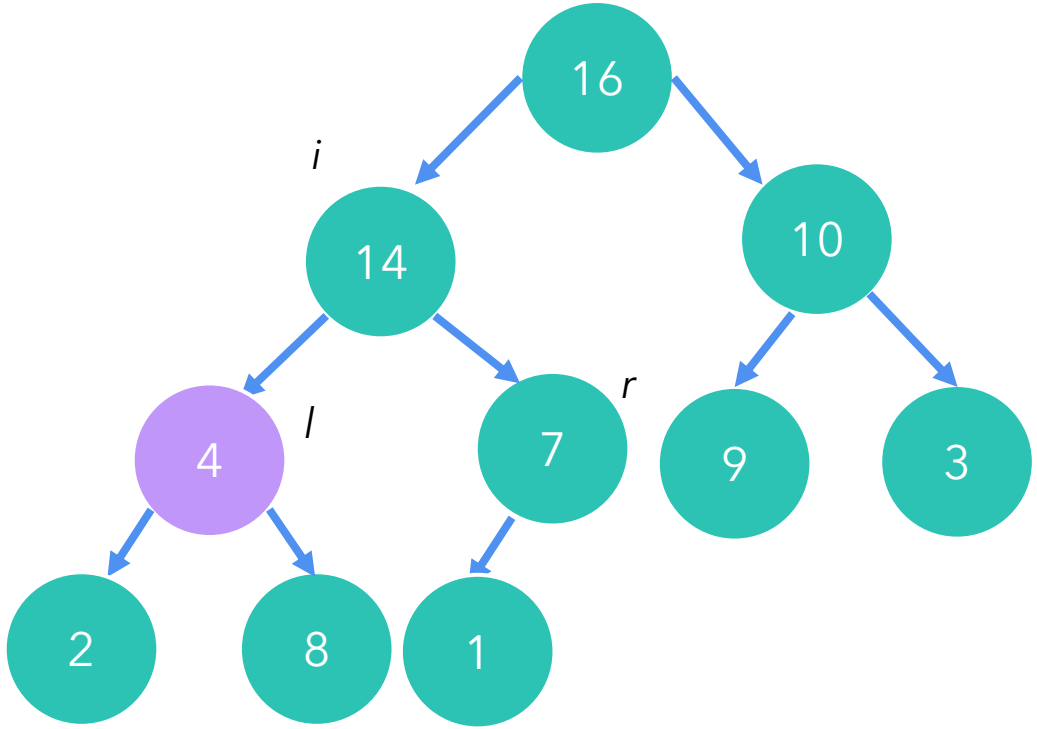
Operaciones de un HEAP – MAX-HEAPIFY



largest = l

```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

Operaciones de un HEAP – MAX-HEAPIFY



largest = l

```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

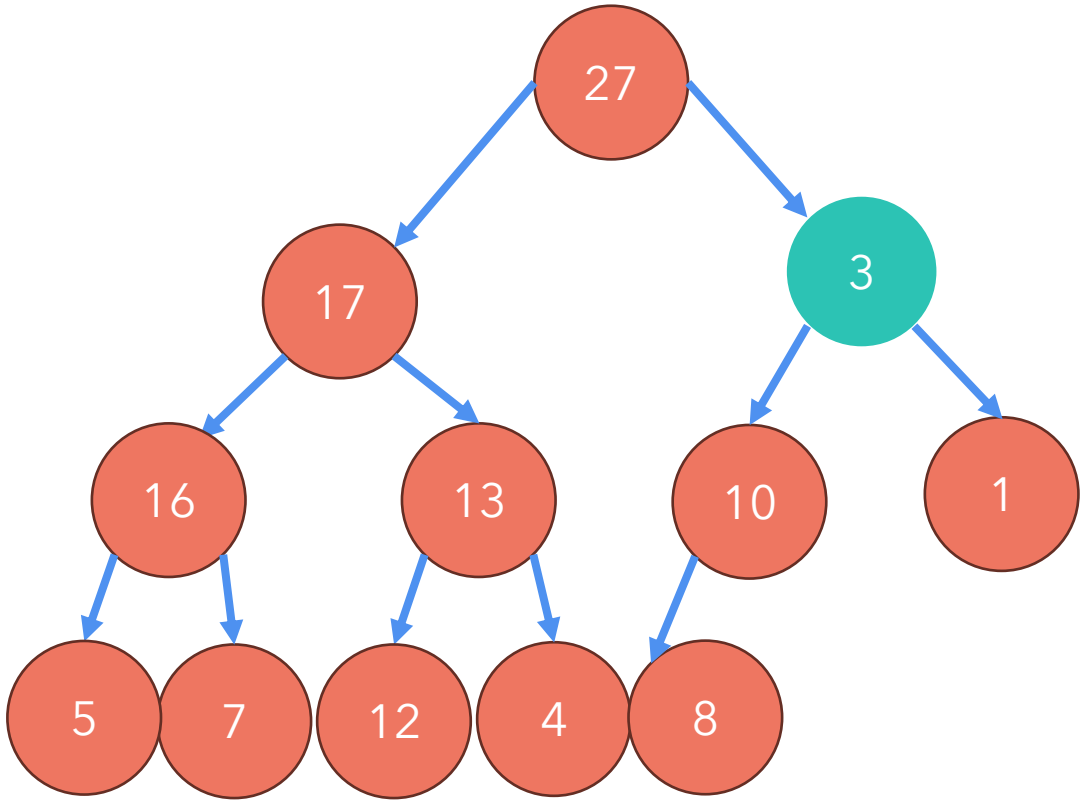
Ejemplo

Aplique el algoritmo
MAX-HEAPIFY(A,3,14) para
A = [27-17- 3- 16 -13 -10-1 -5-7 -12-4- 8- 9- 0]

```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

Ejemplo

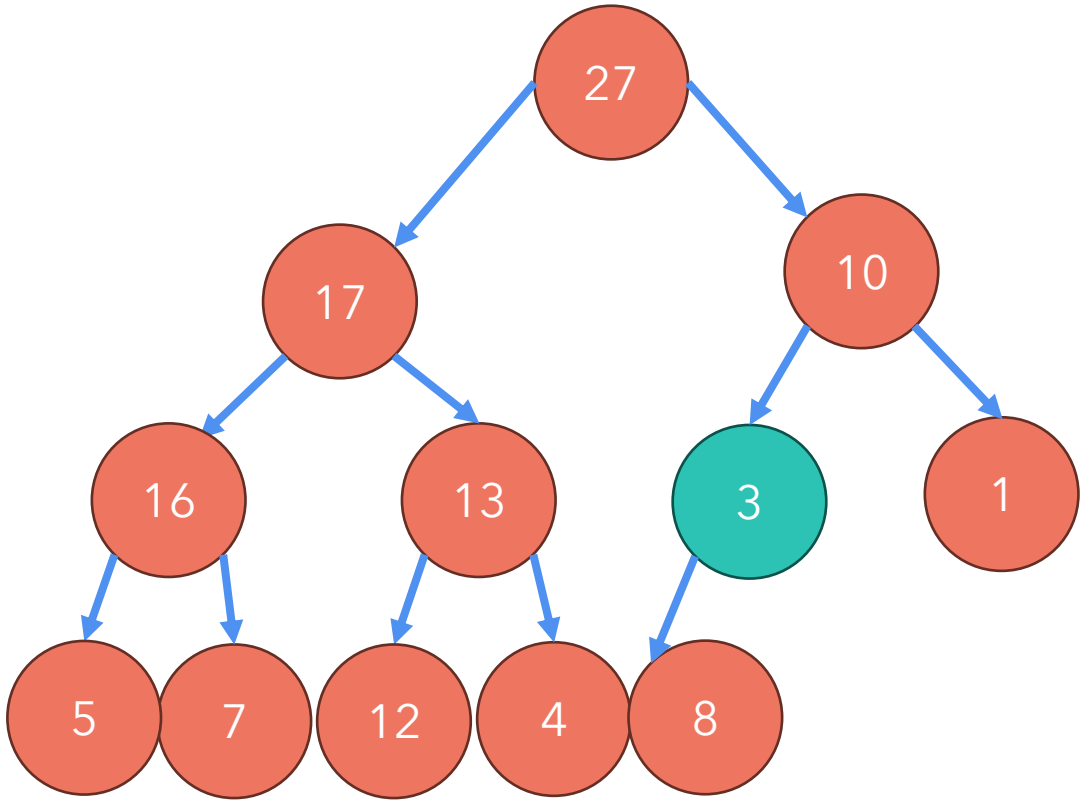
Aplique el algoritmo
MAX-HEAPIFY(A,3,14) para
A = [27-17- 3- 16 -13 -10-1 -5-7 -12-4- 8- 9- 0]




```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

Ejemplo

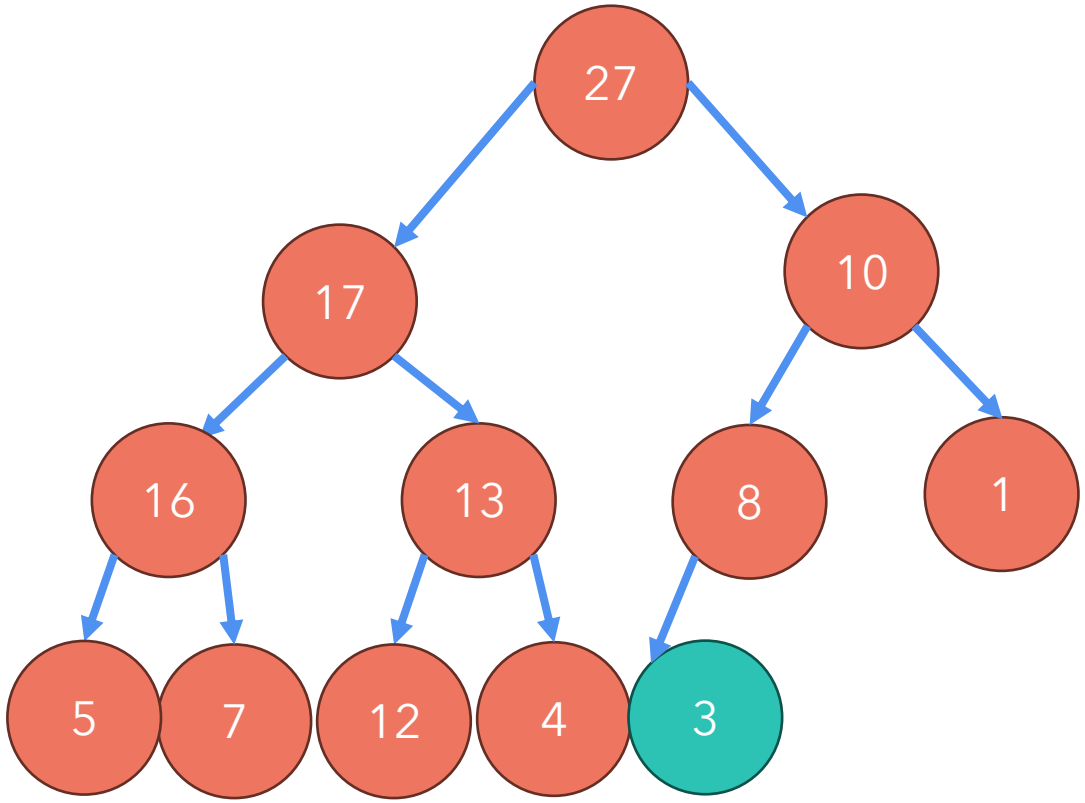
Aplique el algoritmo
MAX-HEAPIFY(A,3,14) para
A = [27-17- 3- 16 -13 -10-1 -5-7 -12-4- 8- 9- 0]




```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

Ejemplo

Aplique el algoritmo
MAX-HEAPIFY(A,3,14) para
A = [27-17- 3- 16 -13 -10-1 -5-7 -12-4- 8- 9- 0]




```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

Operaciones de un HEAP – MAX-HEAPIFY

```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

O(1)

T(n/2)

Operaciones de un HEAP – MAX-HEAPIFY

```
MAX-HEAPIFY(A,i,heap_size)
1.  l = Left(i)
2.  r = Right(i)
3.  IF l ≤ heap_size & A[l]>A[i]
4.     largest = l
5.  ELSE
6.     largest = i
7.  IF r ≤ heap_size & A[r]>A[largest]
8.     largest = r
9.  IF largest!=i
10.    temp = A[i]
11.    A[i] = A[largest]
12.    A[largest] = temp
13.    MAX-HEAPIFY(A, largest, heap_size)
```

O(1)

T(n/2)

T(n) = T(n/2)+O(1)

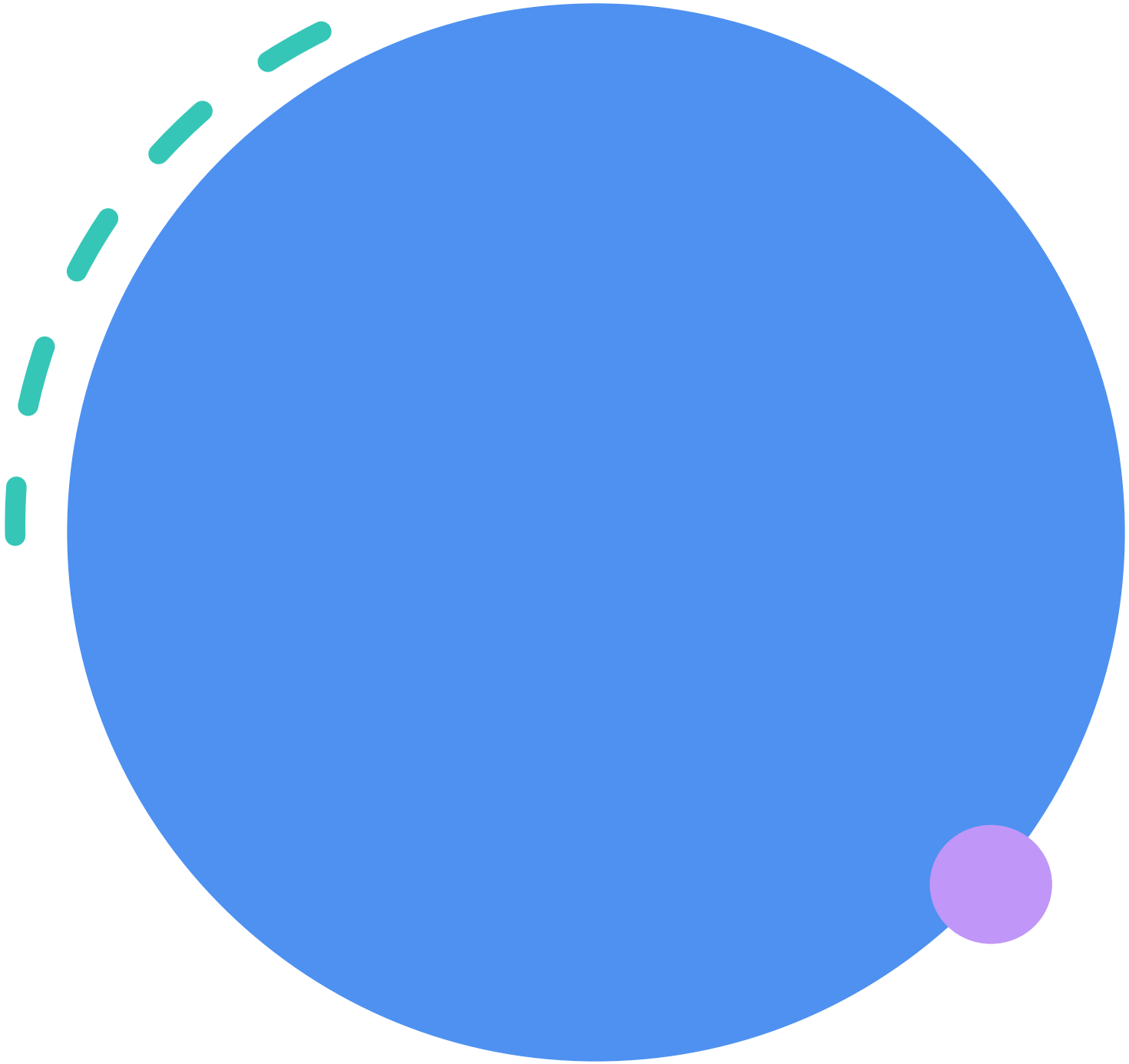
$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a \geq 1$, b> 1, y $f(n)$ una función asintóticamente positiva

f(n) = O(1)
a = 1
b = 2

- Si $f(n) = \Theta\left(n^{\log_b a}\right)$ entonces $T(n) = \Theta(n^{\log_b a} \log n)$

T(n) = O(log n)



Operaciones de un HEAP

Operación	Descripción	Complejidad
MAX-HEAPIFY	Mantiene la propiedad heap	O(log n)
BUILD-MAX-HEAP	Construye un heap a partir de un arreglo no ordenado	O(n log n)
HEAPSORT	Algoritmo de ordenamiento, organiza los datos en el mismo arreglo	O(n log n)
MAX-HEAP-INSERT	Inserta un dato en el arreglo manteniendo la propiedad HEAP	O(log n)
HEAP-EXTRACT-MAX	Retorna y elimina de la colección el valor máximo	O(log n)
HEAP-MAXIMUM	Retorna el valor máximo	O(1)

Operaciones de un HEAP – BUILD-MAX-HEAP

- Construye un heap a partir de un arreglo no ordenado
- Podemos usar el algoritmo MAX-HEAPIFY para convertir un arreglo en un MAX-HEAP
- Para ello solo es necesario recorrer el arreglo en forma descendente desde A.length/2 hasta 0 aplicando la operación MAX-HEAPIFY estudiada anteriormente

BUILD-MAX-HEAP(A)

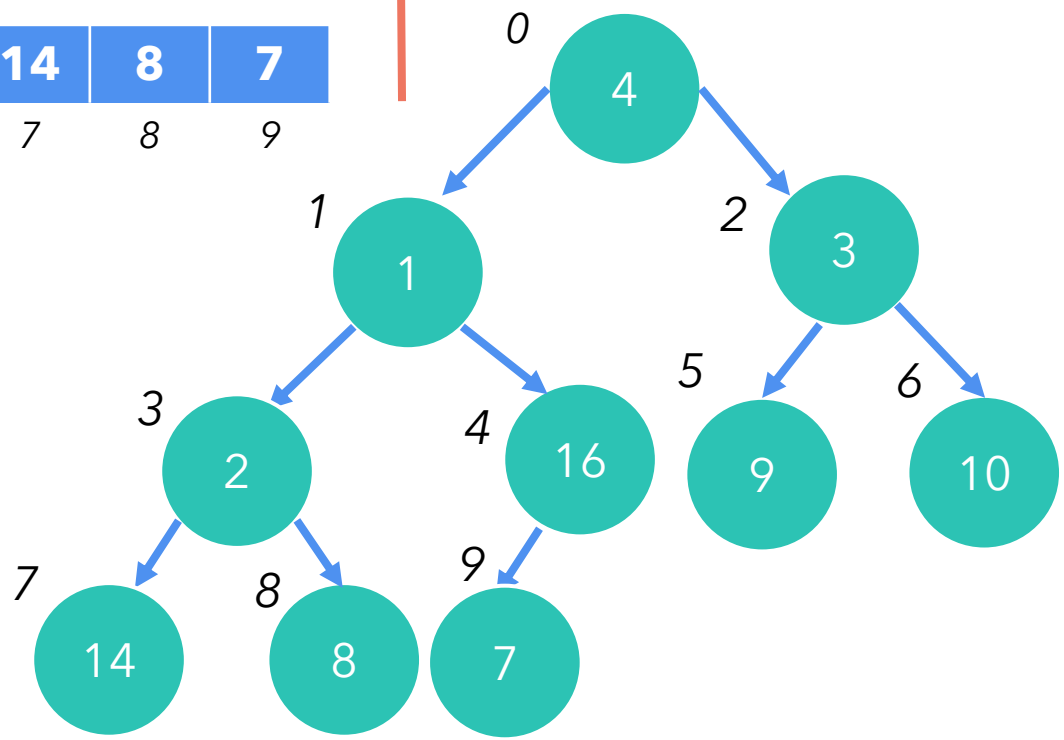
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	1	3	2	16	9	10	14	8	7
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



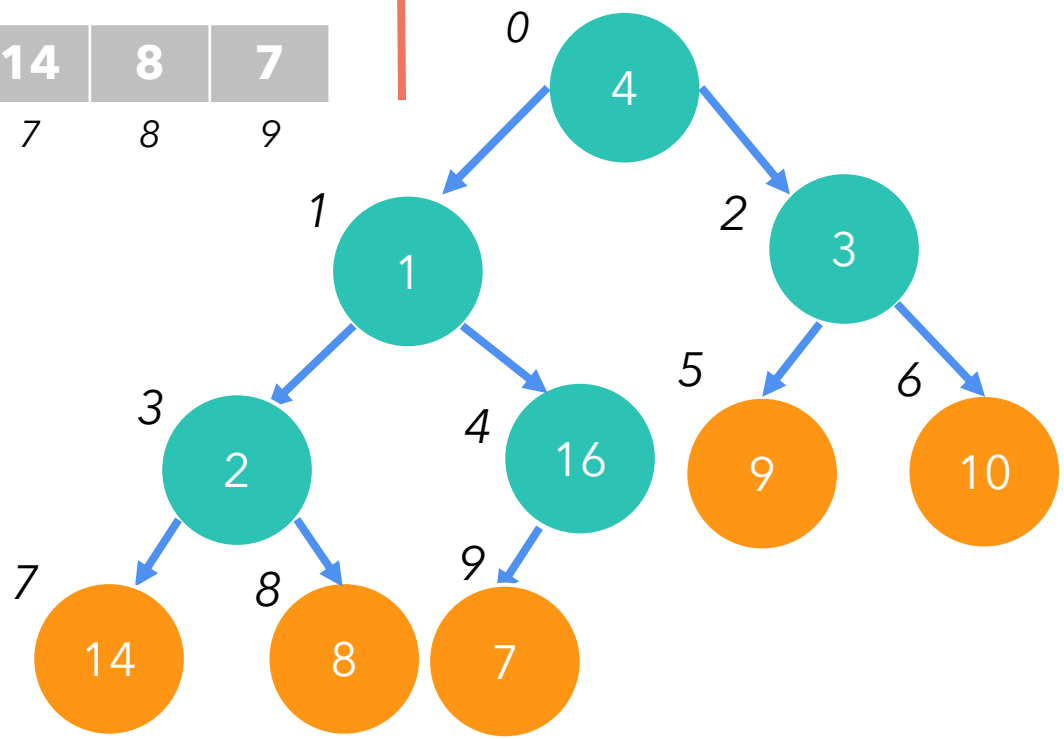
Noté que no satisfice la propiedad

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	1	3	2	16	9	10	14	8	7
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



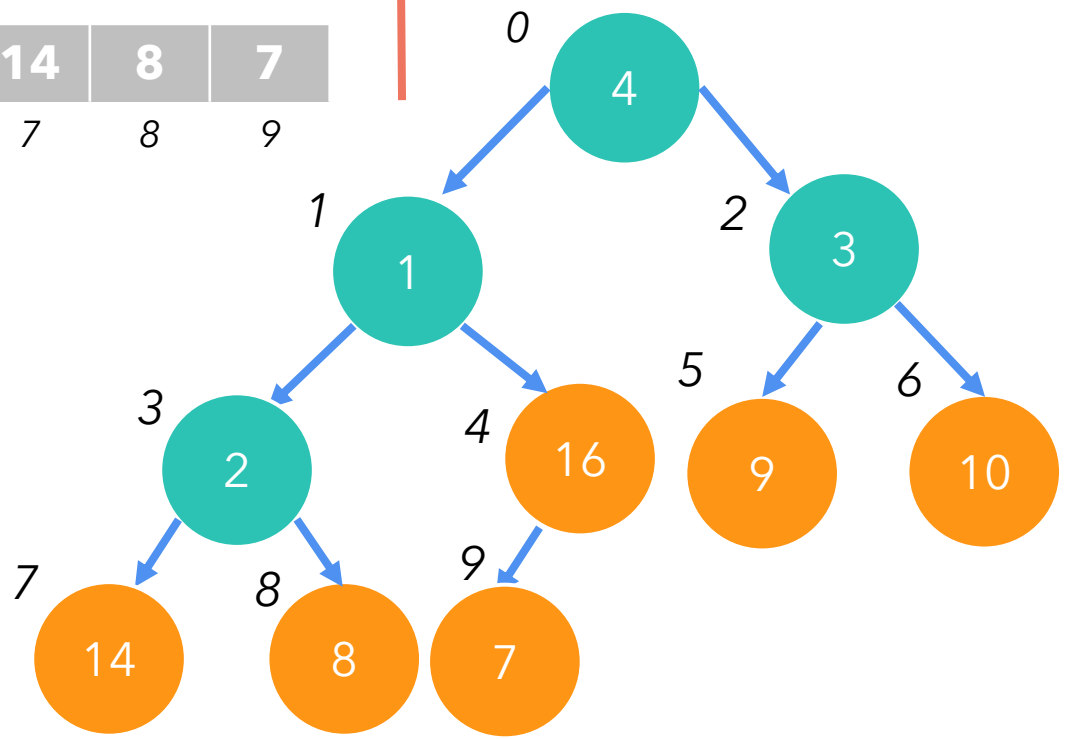
Se aplica el algoritmo MAX-HEAPIFY a las posiciones 4,3,2,1, y 0

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	1	3	2	16	9	10	14	8	7
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



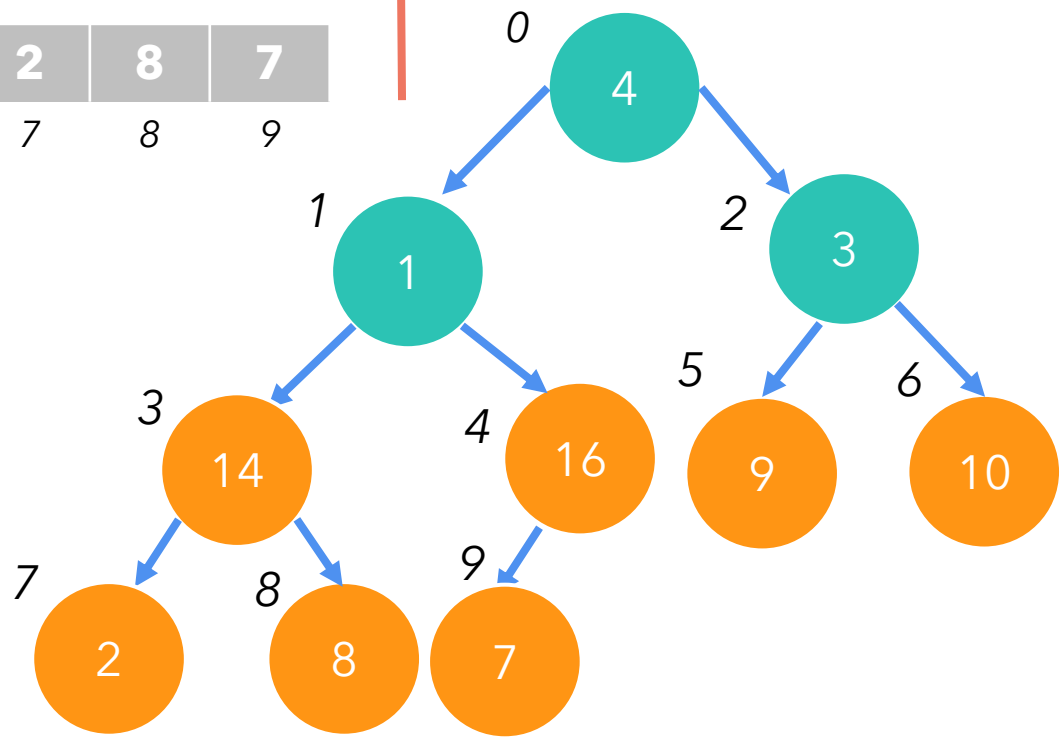
Para i =4

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	1	3	14	16	9	10	2	8	7
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



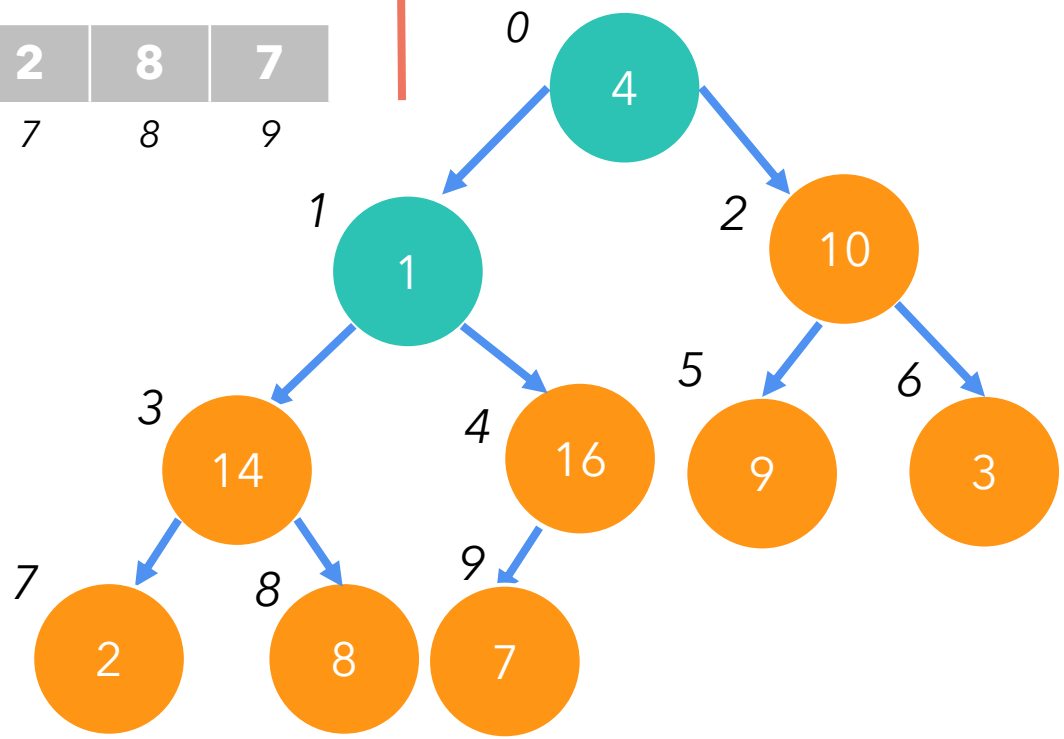
Para i =3

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	1	10	14	16	9	3	2	8	7
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



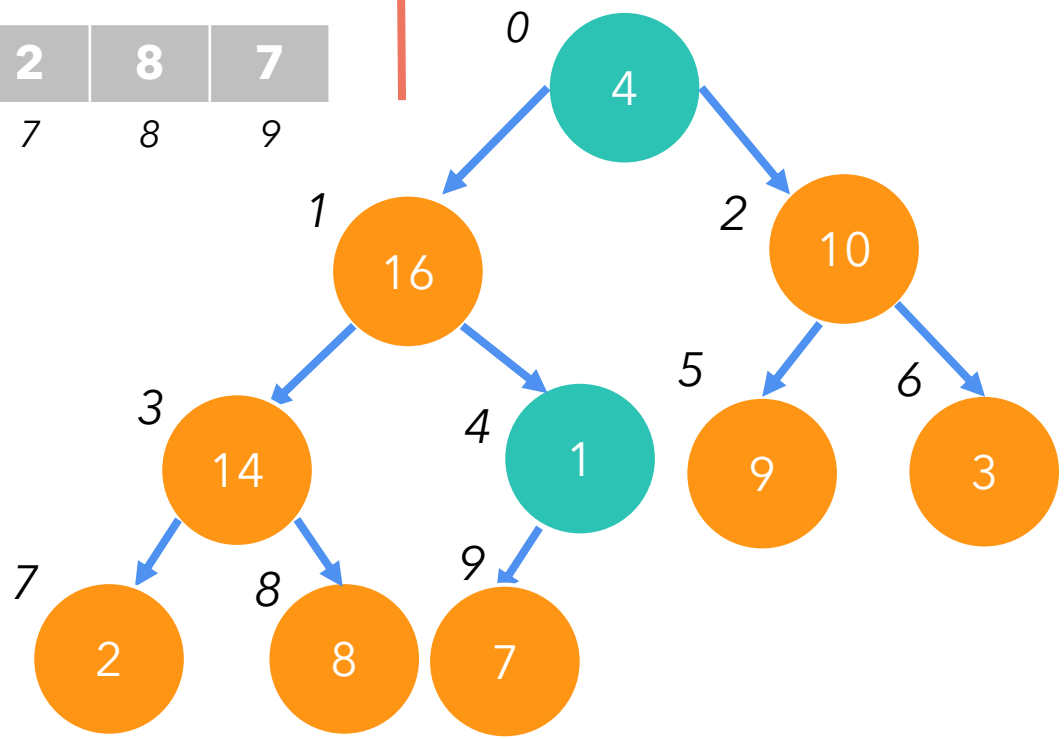
Para i =2

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	16	10	14	1	9	3	2	8	7
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



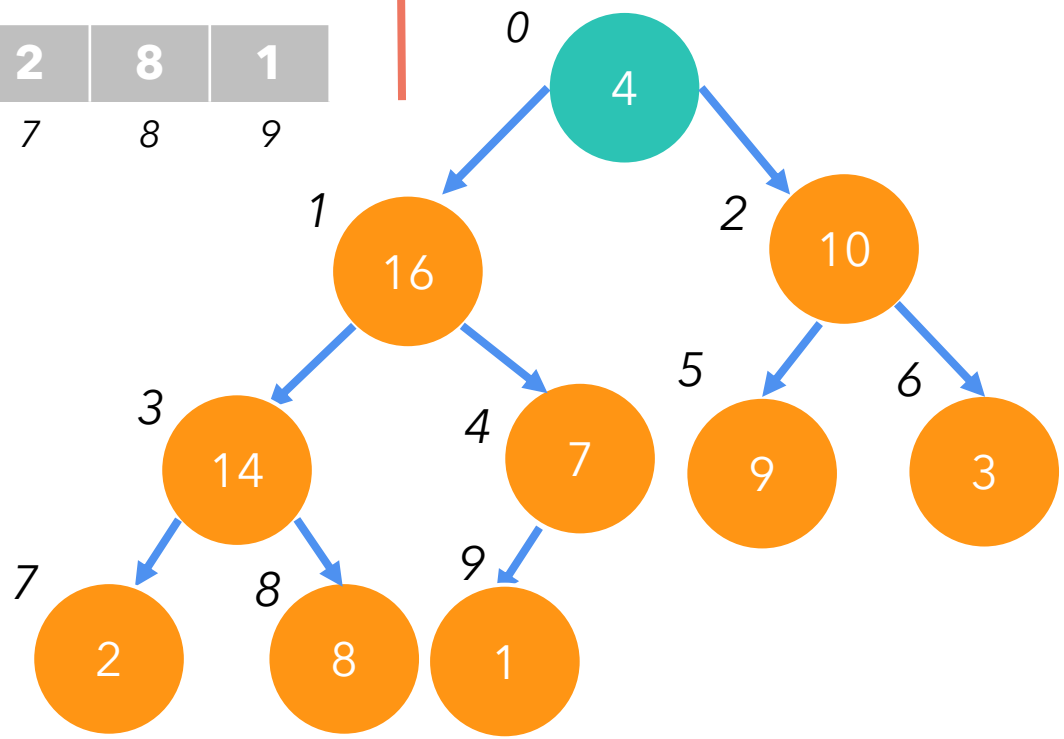
Para i =1

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

4	16	10	14	7	9	3	2	8	1
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



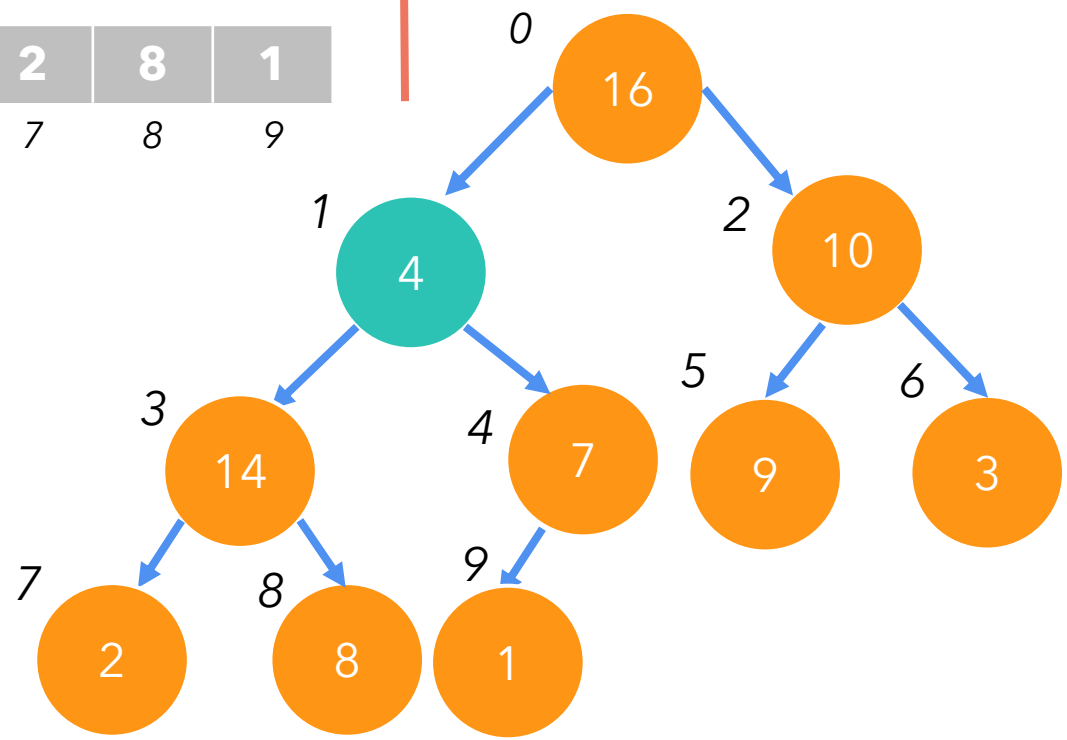
Para i =1

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

16	4	10	14	7	9	3	2	8	1
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



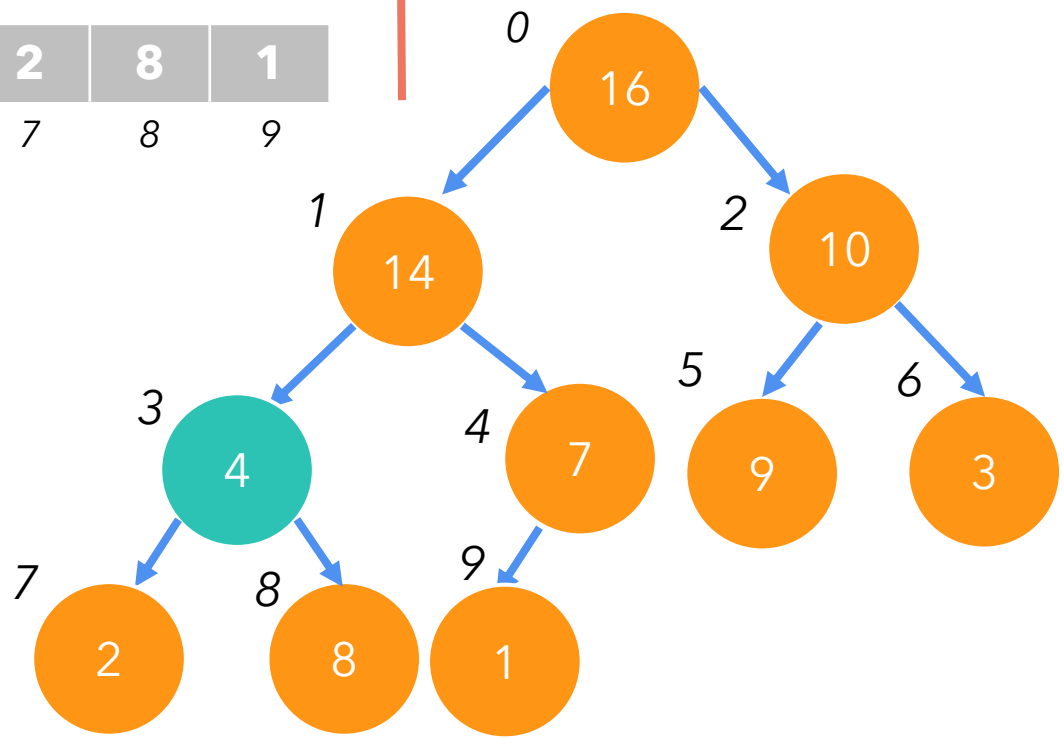
Para i =0

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

16	14	10	4	7	9	3	2	8	1
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



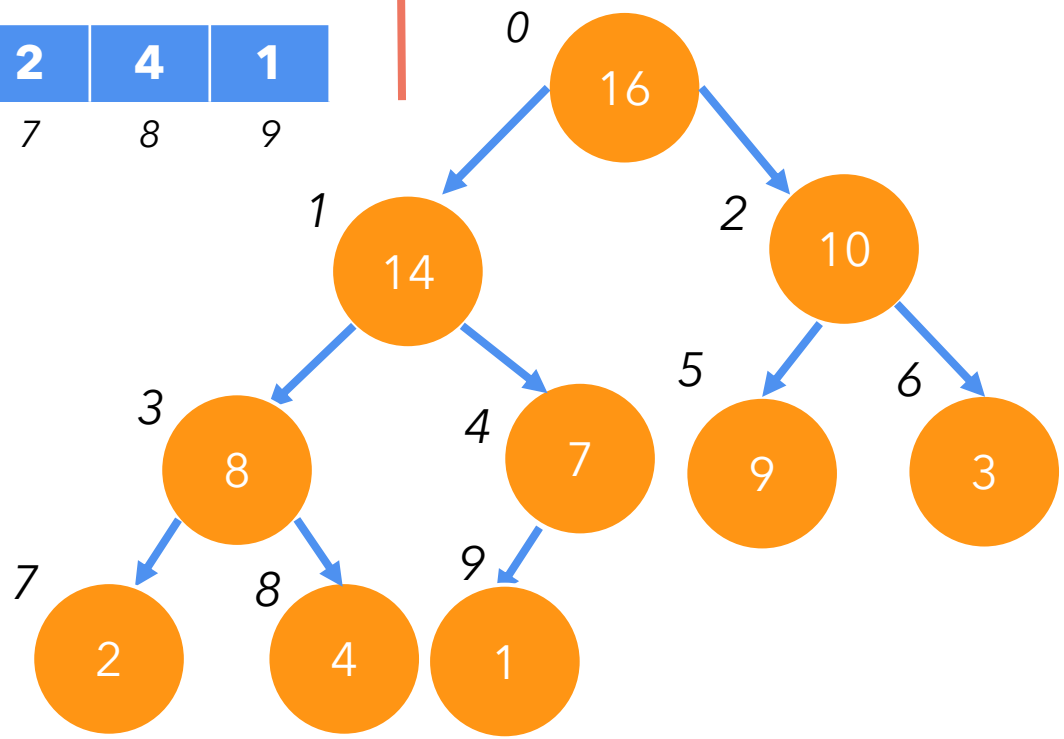
Para i =0

Operaciones de un HEAP – BUILD-MAX-HEAP

Dado el arreglo:
A = [4 -1- 3- 2 -16- 9 -10 -14 – 8 - 7]
vamos a construir un MAX-HEAP

16	14	10	8	7	9	3	2	4	1
0	1	2	3	4	5	6	7	8	9

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)



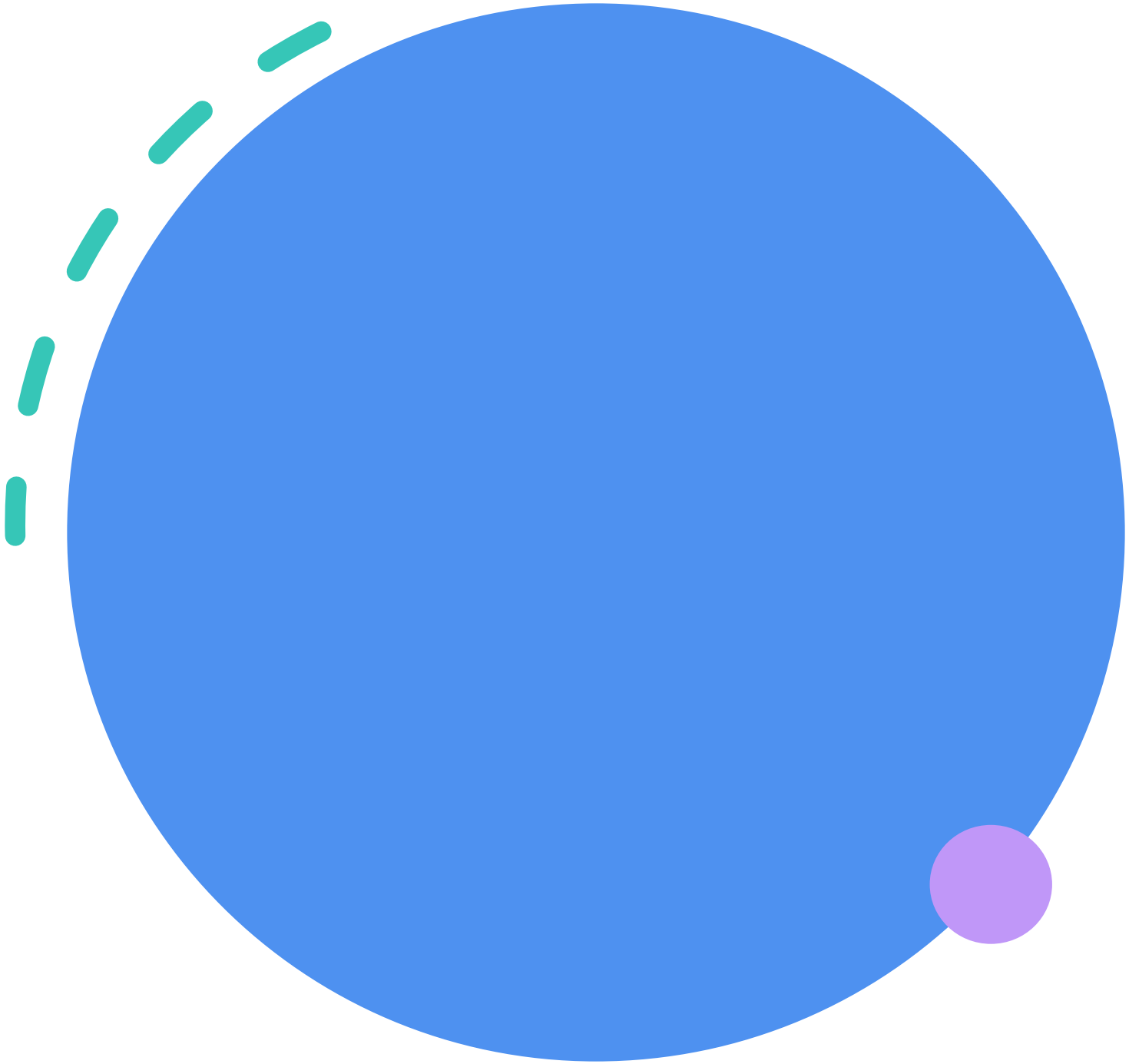
Para i =0

Operaciones de un HEAP – BUILD-MAX-HEAP

BUILD-MAX-HEAP(A)
1. FOR i = A.length/2-1 TO 0
2. MAX-HEAPIFY(A,i,heap_size)

O(log n)

O(n log n)



Operaciones de un HEAP

Operación	Descripción	Complejidad
MAX-HEAPIFY	Mantiene la propiedad heap	O(log n)
BUILD-MAX-HEAP	Construye un heap a partir de un arreglo no ordenado	O(n log n)
HEAPSORT	Algoritmo de ordenamiento, organiza los datos en el mismo arreglo	O(n log n)
MAX-HEAP-INSERT	Inserta un dato en el arreglo manteniendo la propiedad HEAP	O(log n)
HEAP-EXTRACT-MAX	Retorna y elimina de la colección el valor máximo	O(log n)
HEAP-MAXIMUM	Retorna el valor máximo	O(1)

Operaciones de un HEAP – HEAPSORT

- Algoritmo de ordenamiento
- Organiza los datos en el mismo arreglo
- Toma ventaja de la propiedad MAX_HEAP

Algoritmo

1. Construir un MAP-HEAP desde un arreglo
2. Iterativamente:
   - Intercambiar el ultimo nodo del HEAP con el nodo en 0 (valor máximo) y disminuir el tamaño del HEAP
   - Aplicar MAX-HEAPIFY al nodo en 0

Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

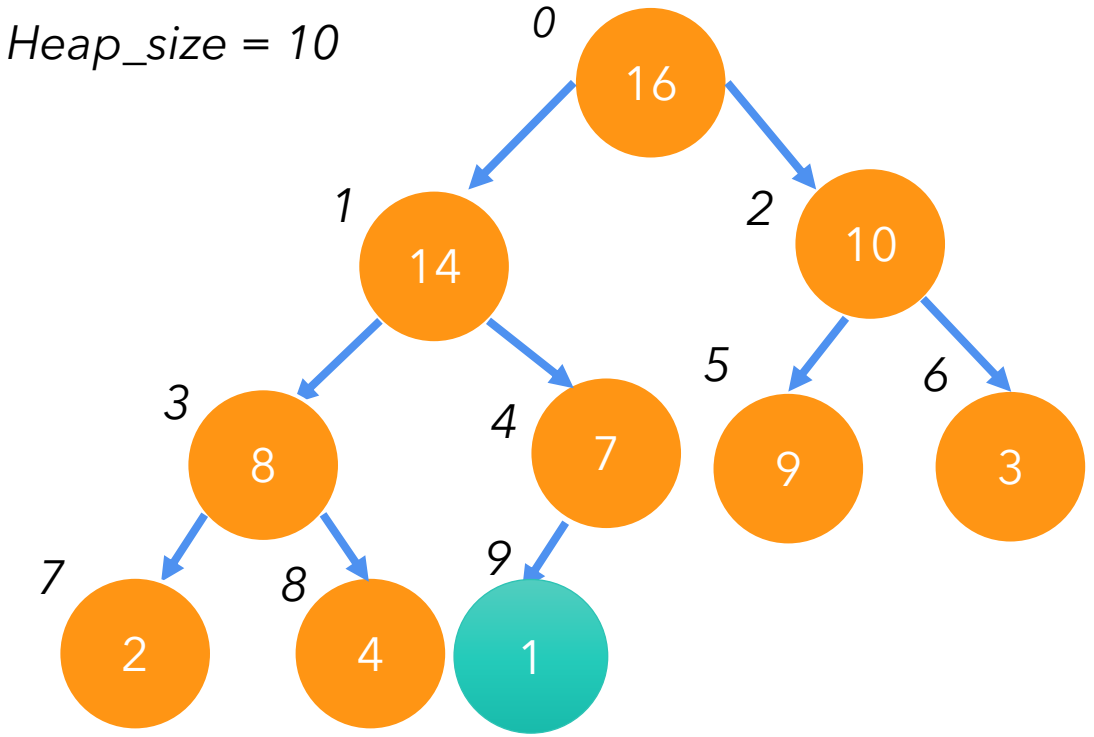
Algoritmo

1. Construir un MAP-HEAP desde un arreglo
2. Iterativamente:
   - Intercambiar el ultimo nodo del HEAP con el nodo en 0 (valor máximo) y disminuir el tamaño del HEAP
   - Aplicar MAX-HEAPIFY al nodo en 0

Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

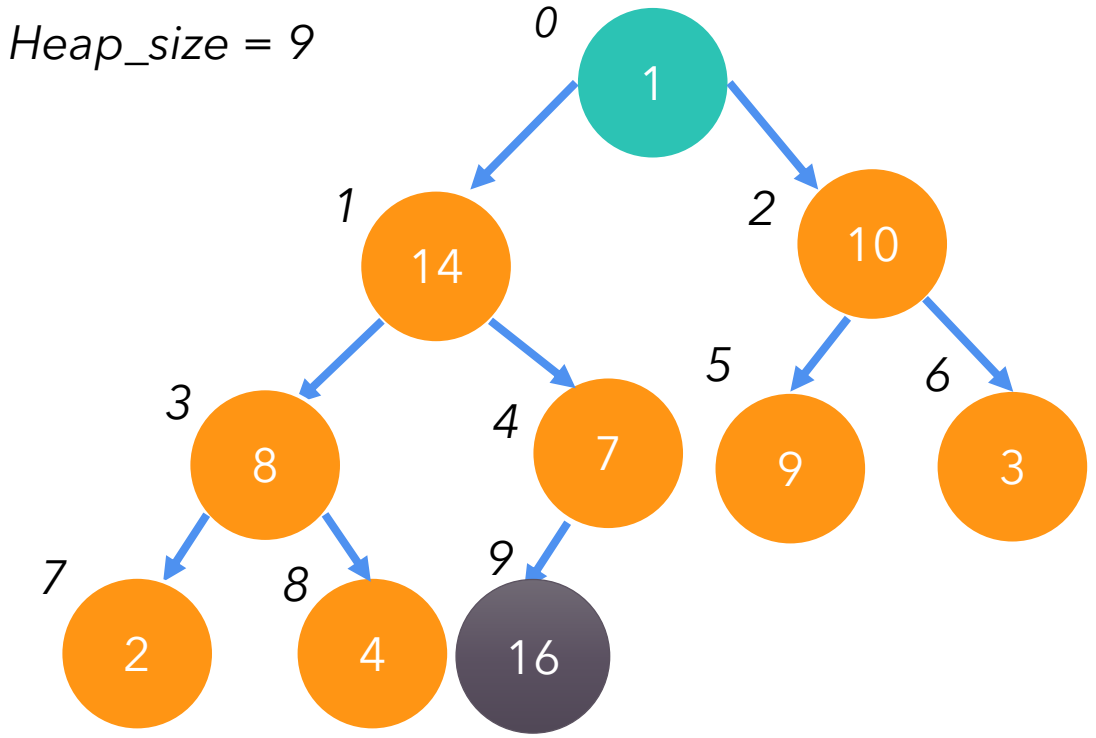
16	14	10	8	7	9	3	2	4	1
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

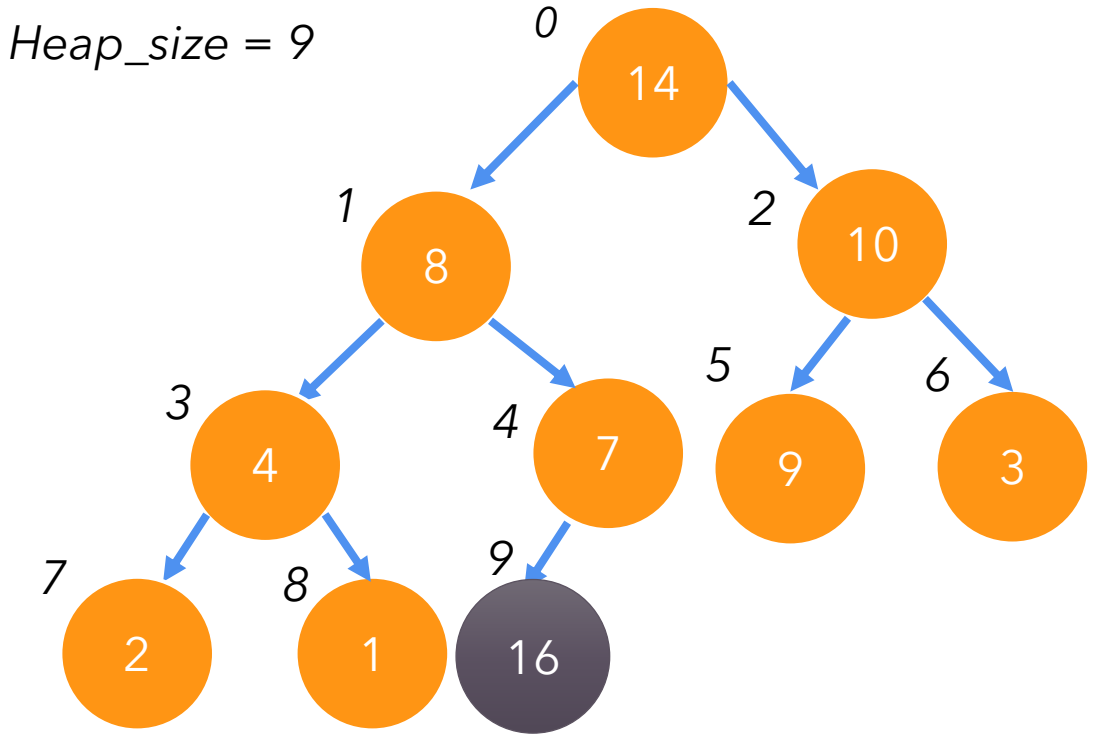
1	14	10	8	7	9	3	2	4	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

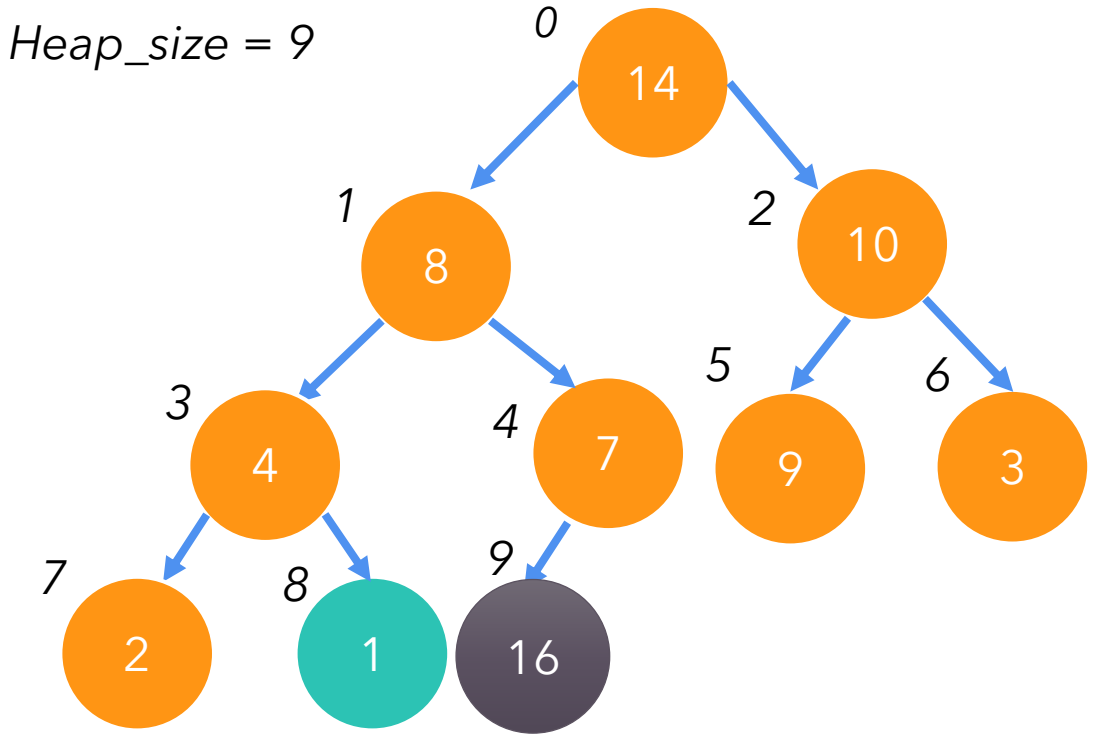
14	8	10	4	7	9	3	2	1	16
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

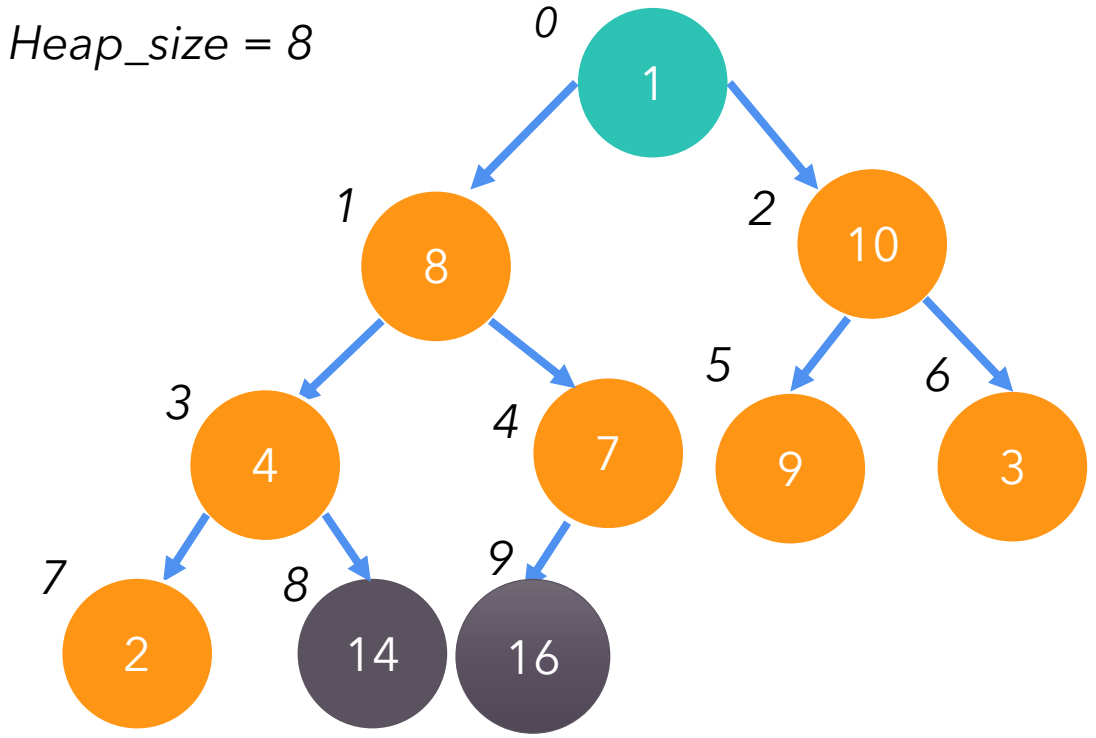
14	8	10	4	7	9	3	2	1	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

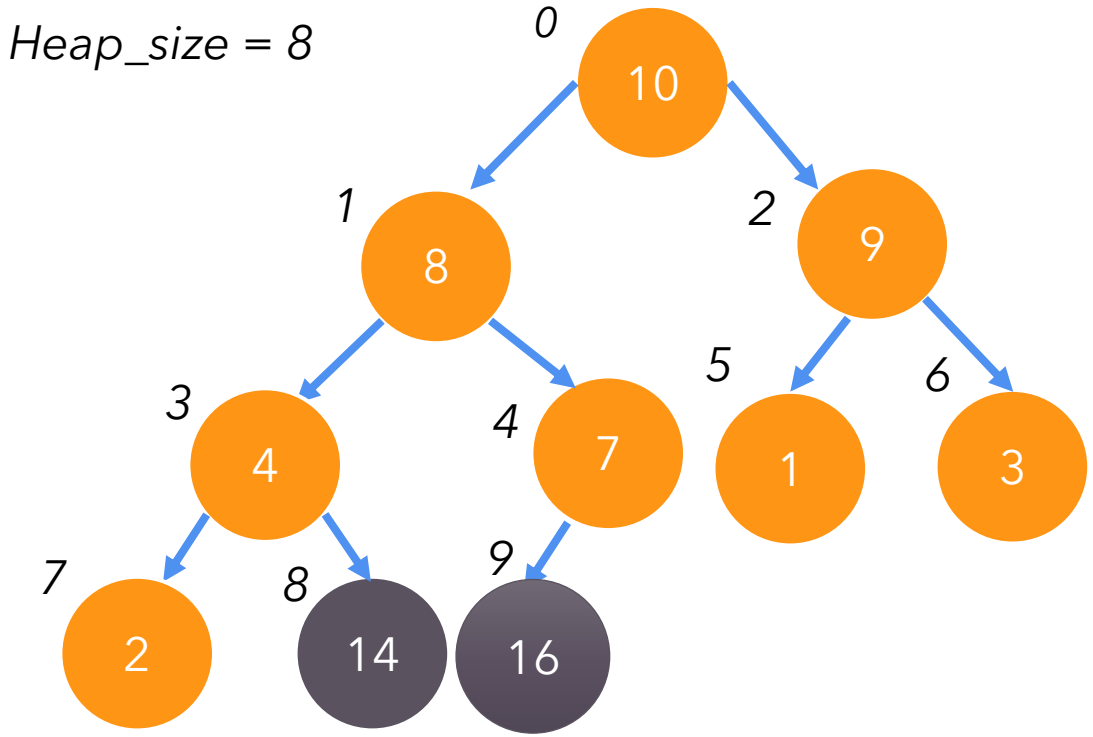
1	8	10	4	7	9	3	2	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

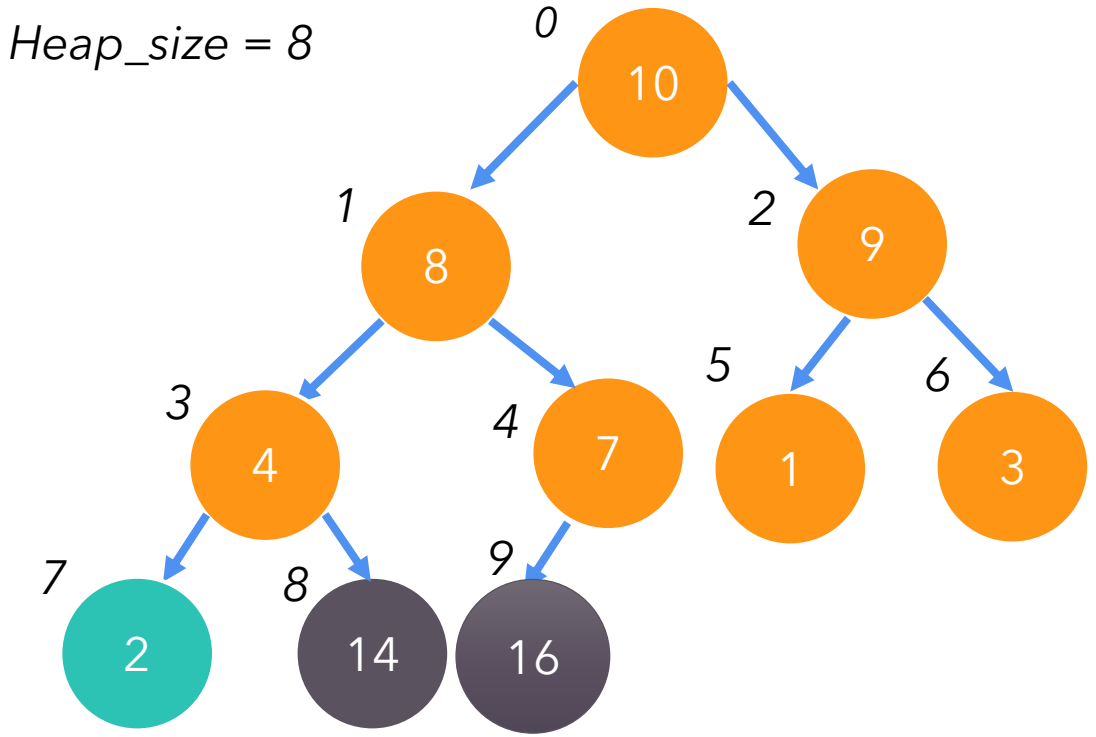
10	8	9	4	7	1	3	2	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

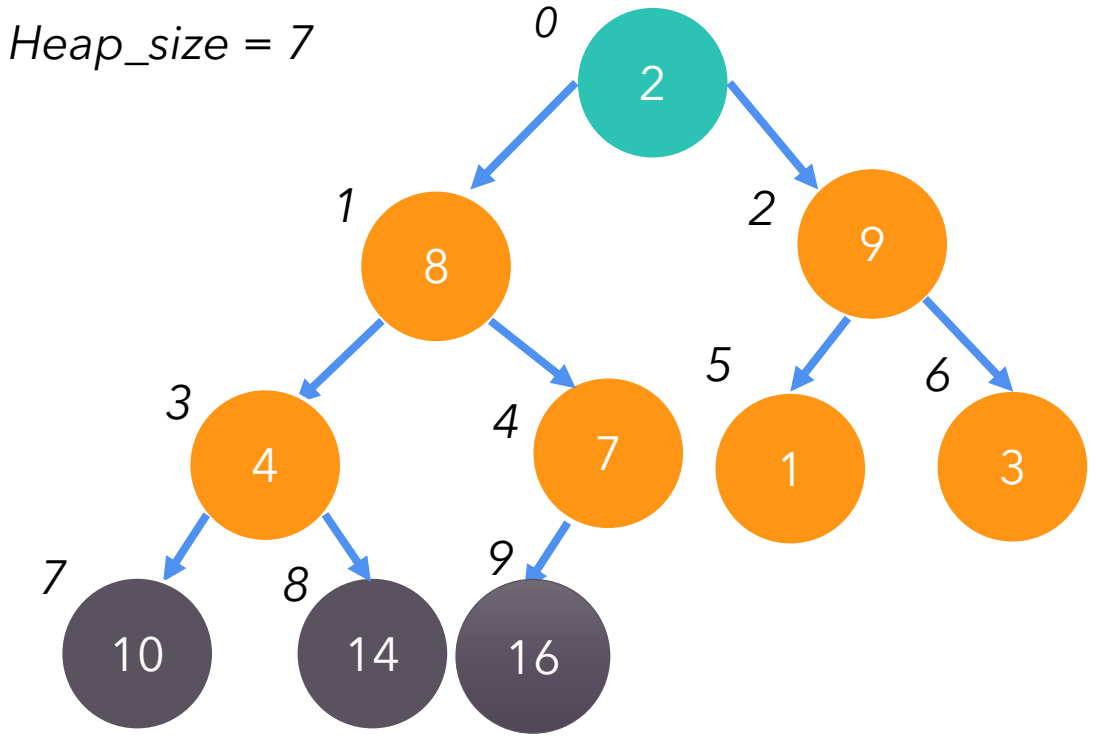
10	8	9	4	7	1	3	2	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

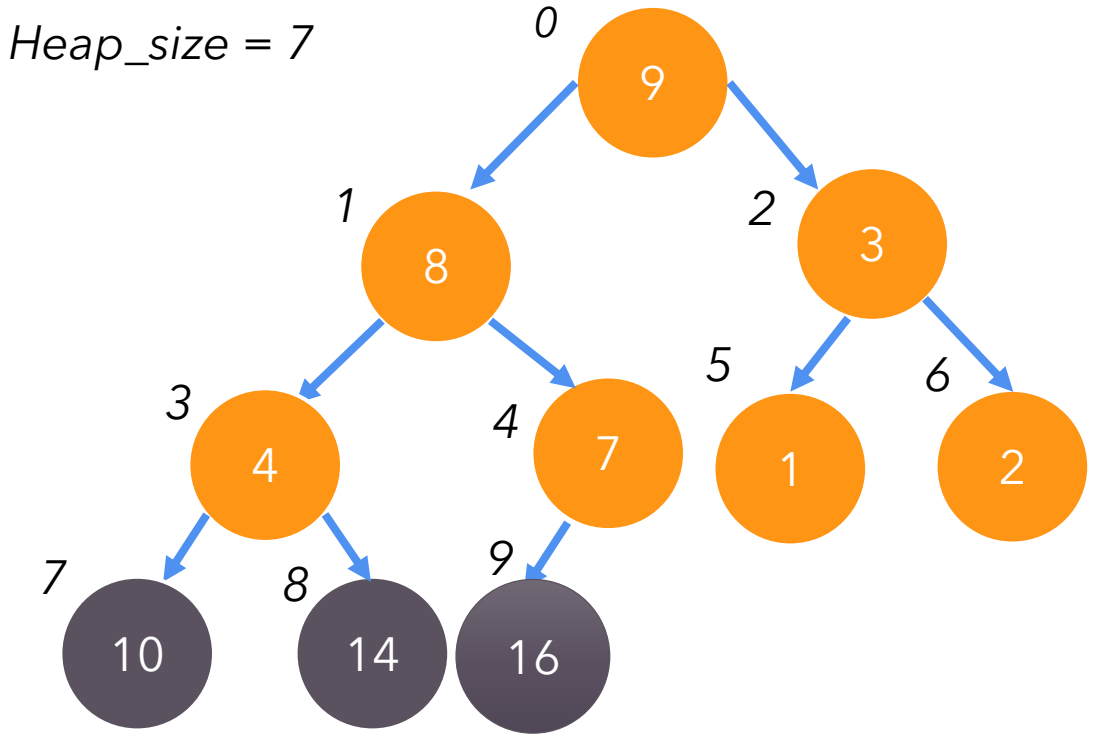
2	8	9	4	7	1	3	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

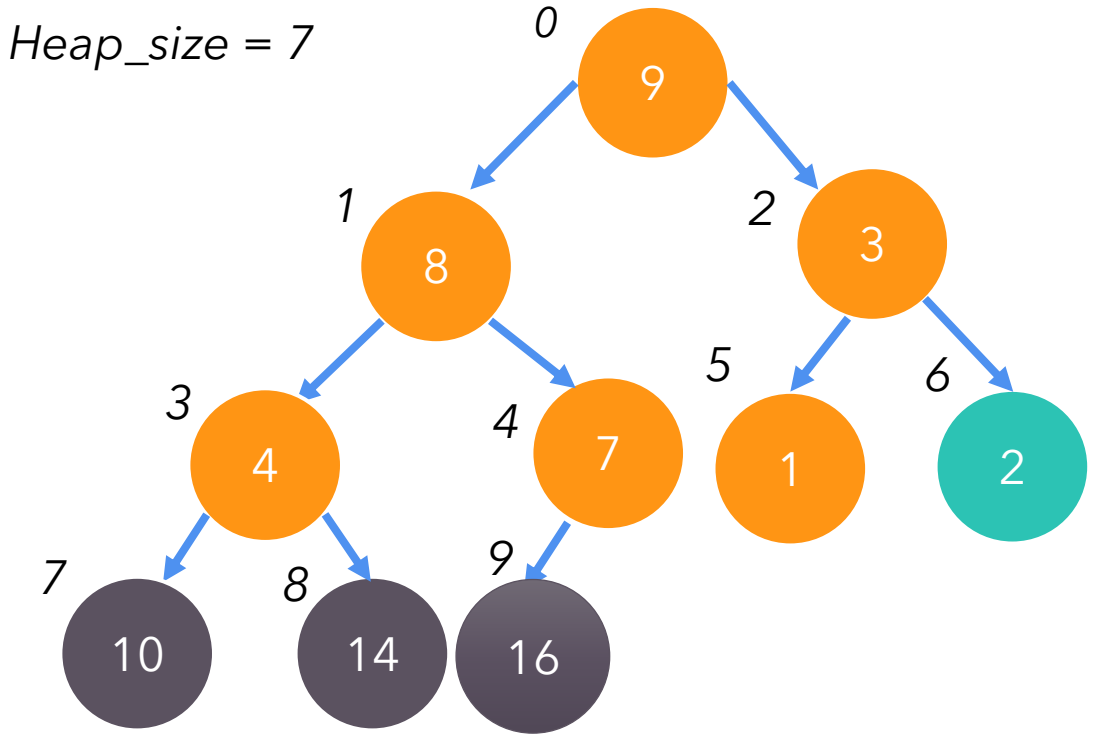
9	8	3	4	7	1	2	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

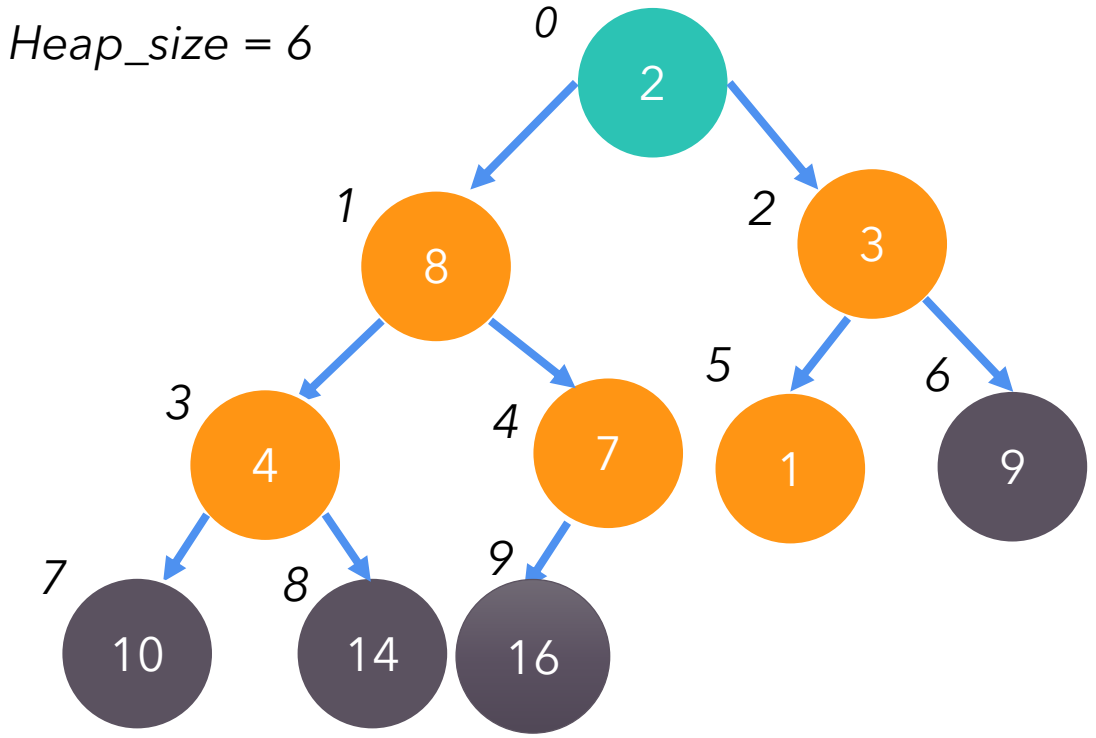
9	8	3	4	7	1	2	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

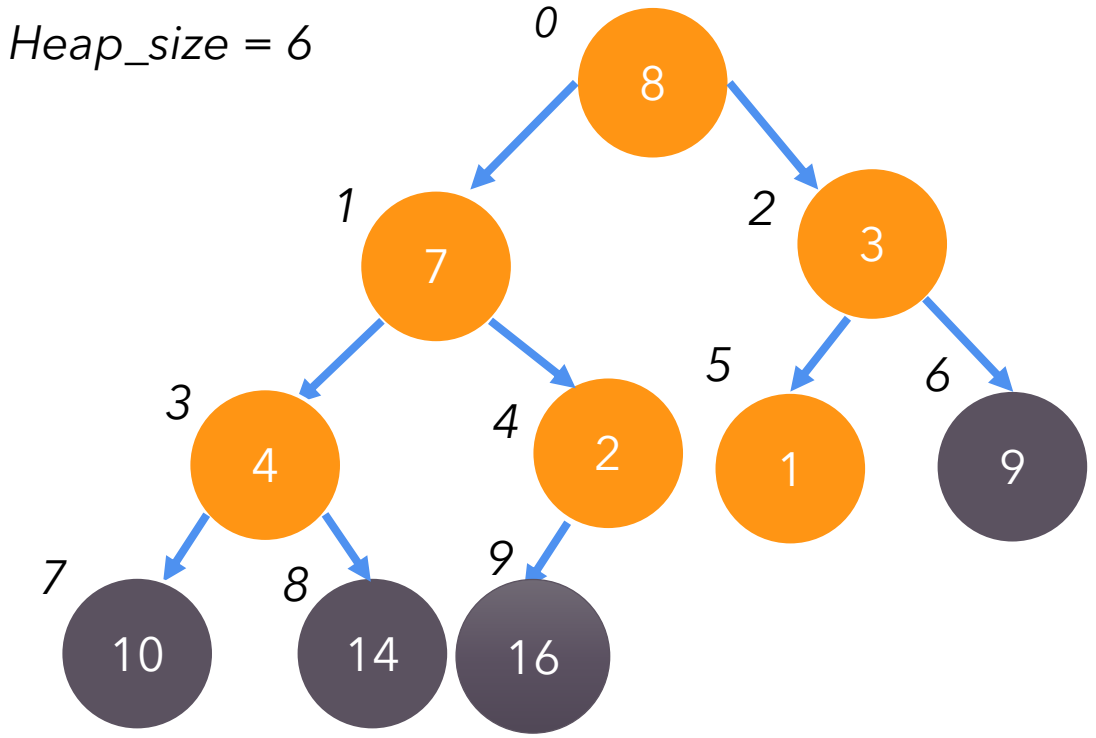
2	8	3	4	7	1	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

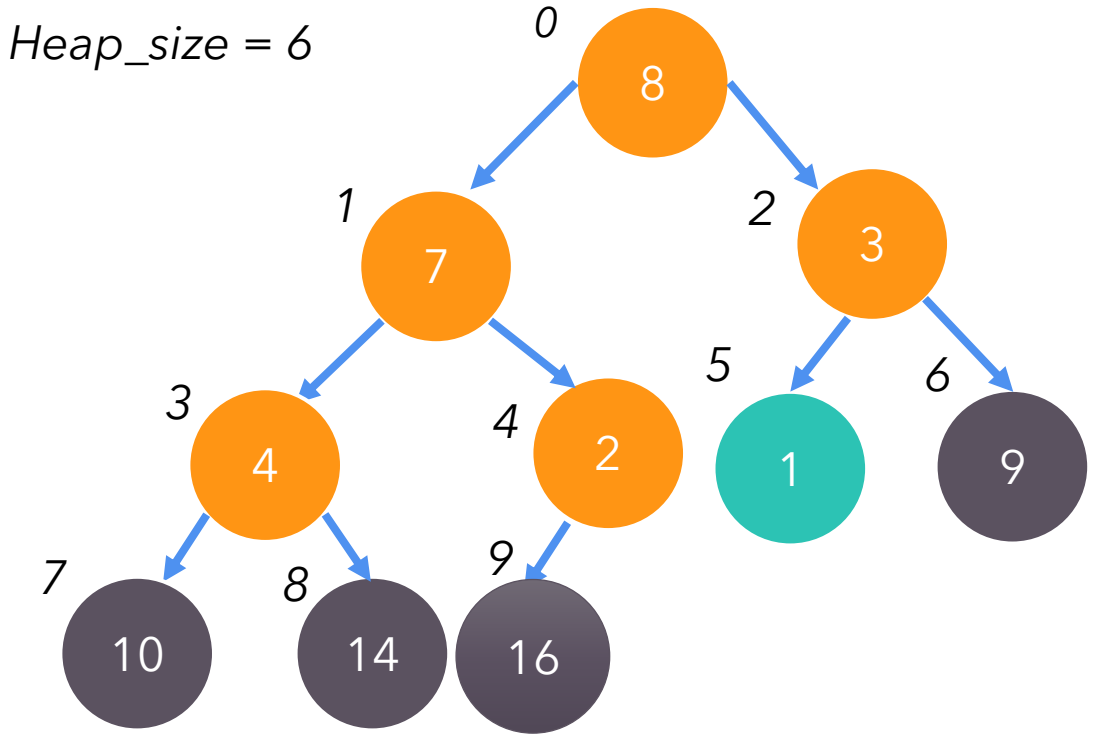
8	7	3	4	2	1	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

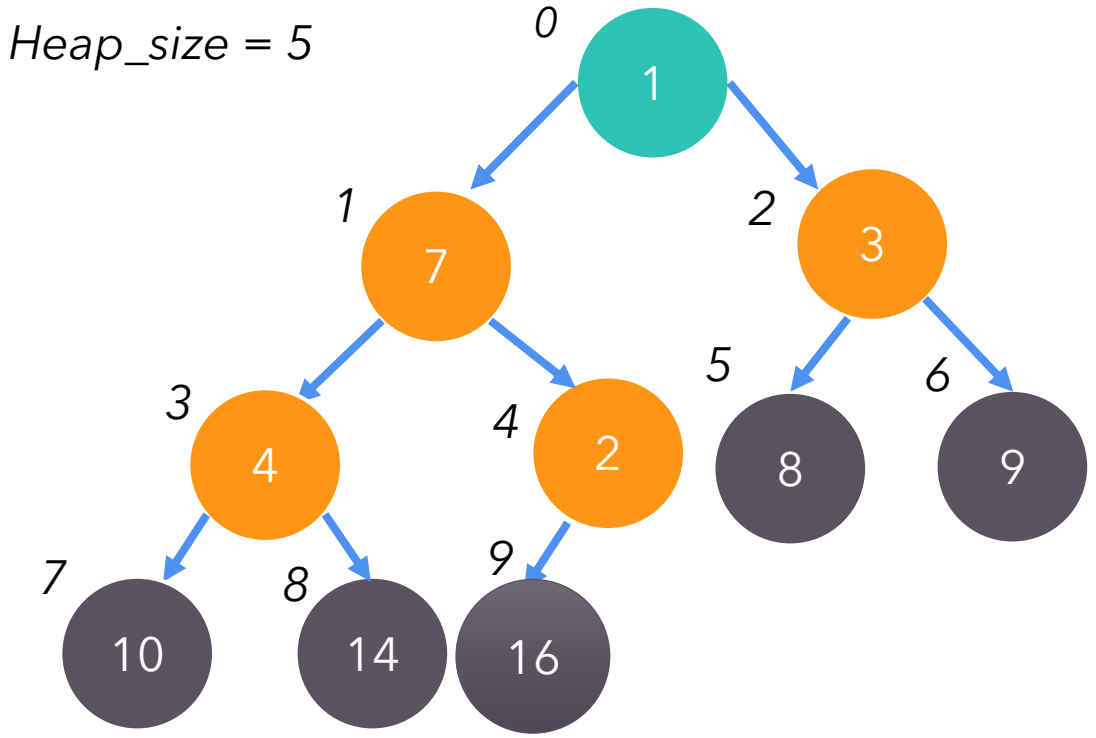
8	7	3	4	2	1	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

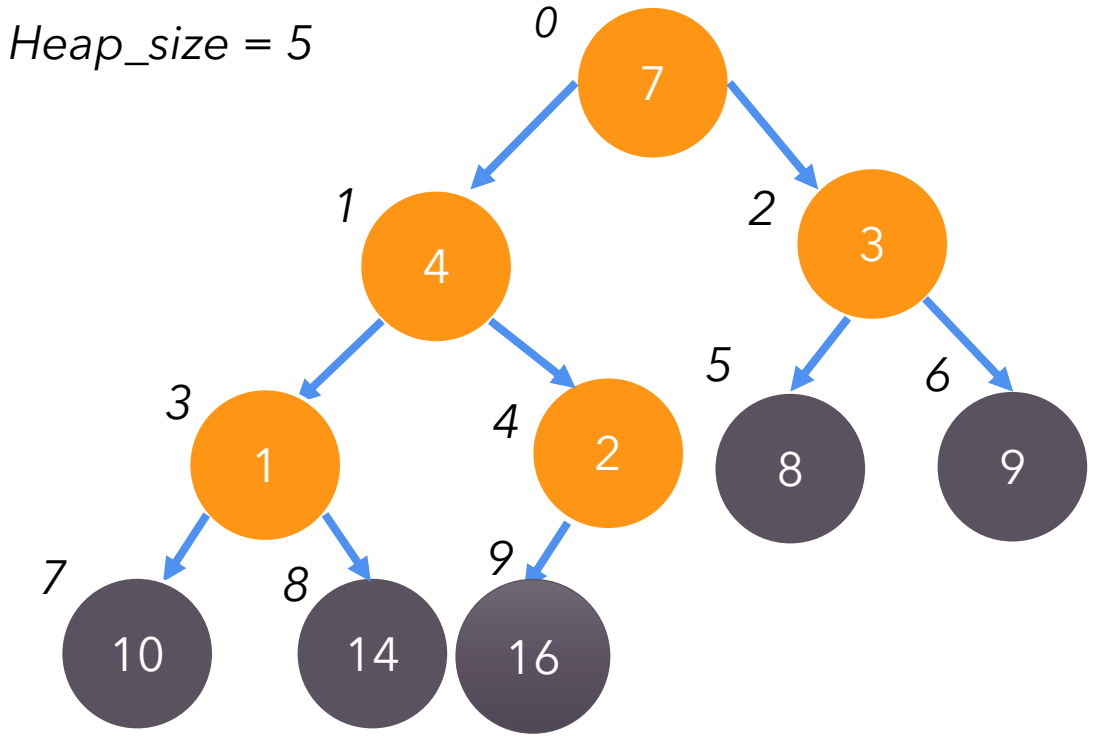
1	7	3	4	2	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

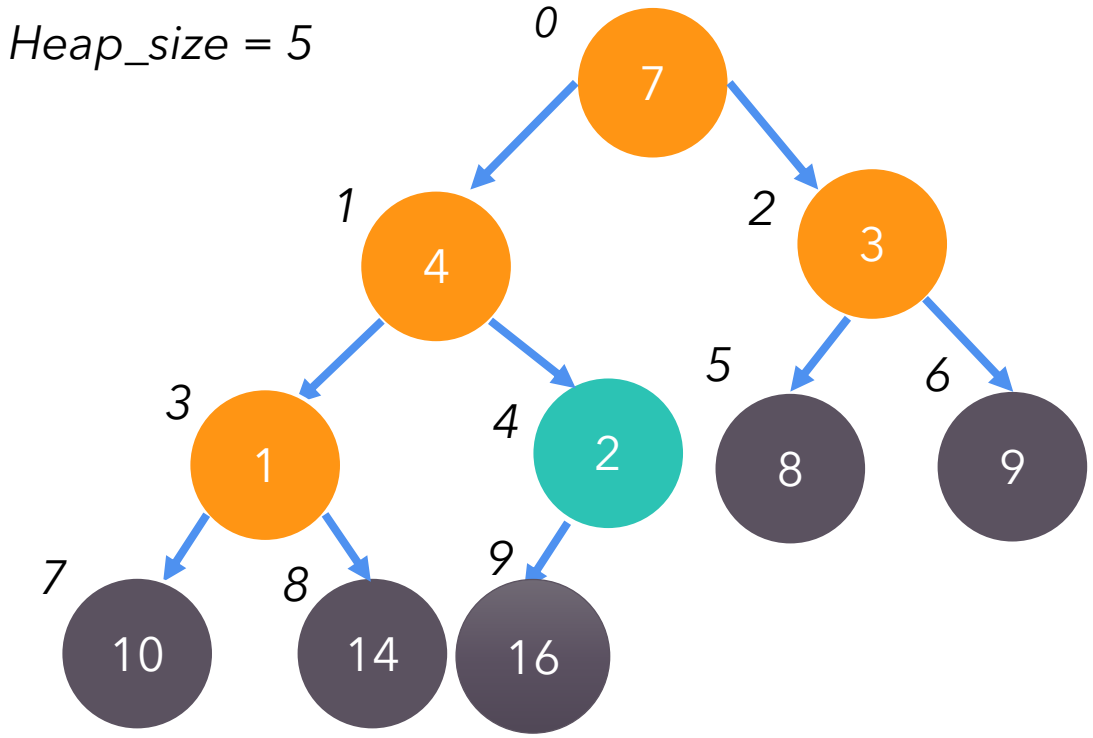
7	4	3	1	2	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

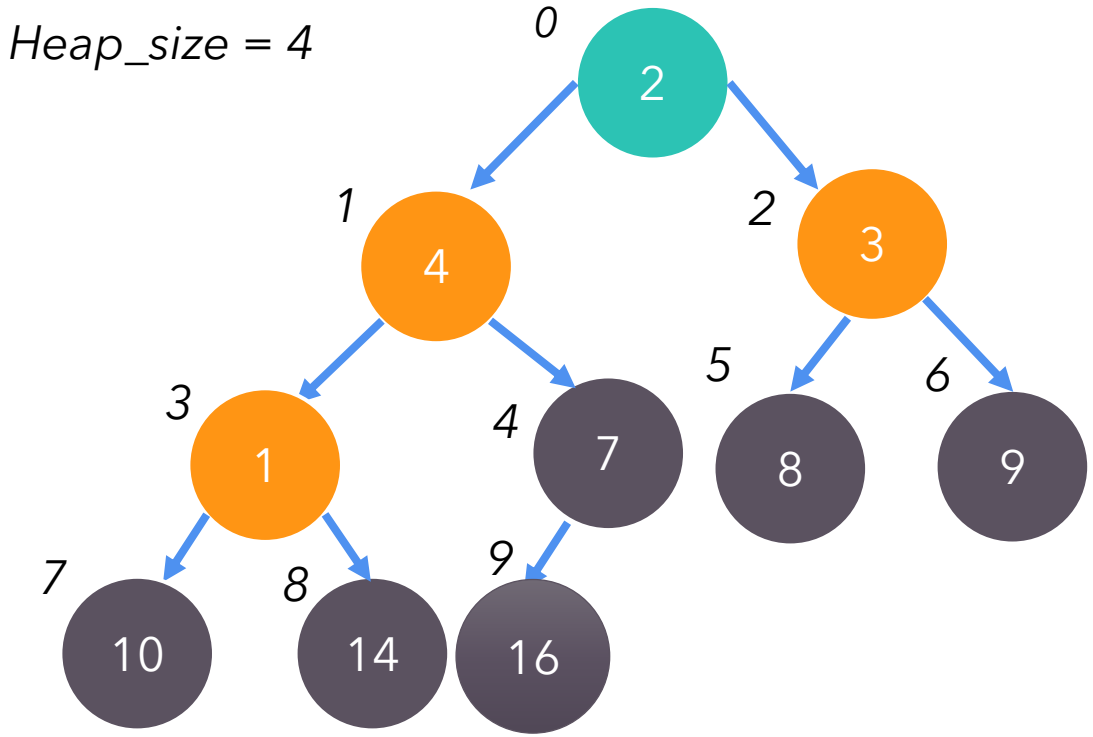
7	4	3	1	2	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

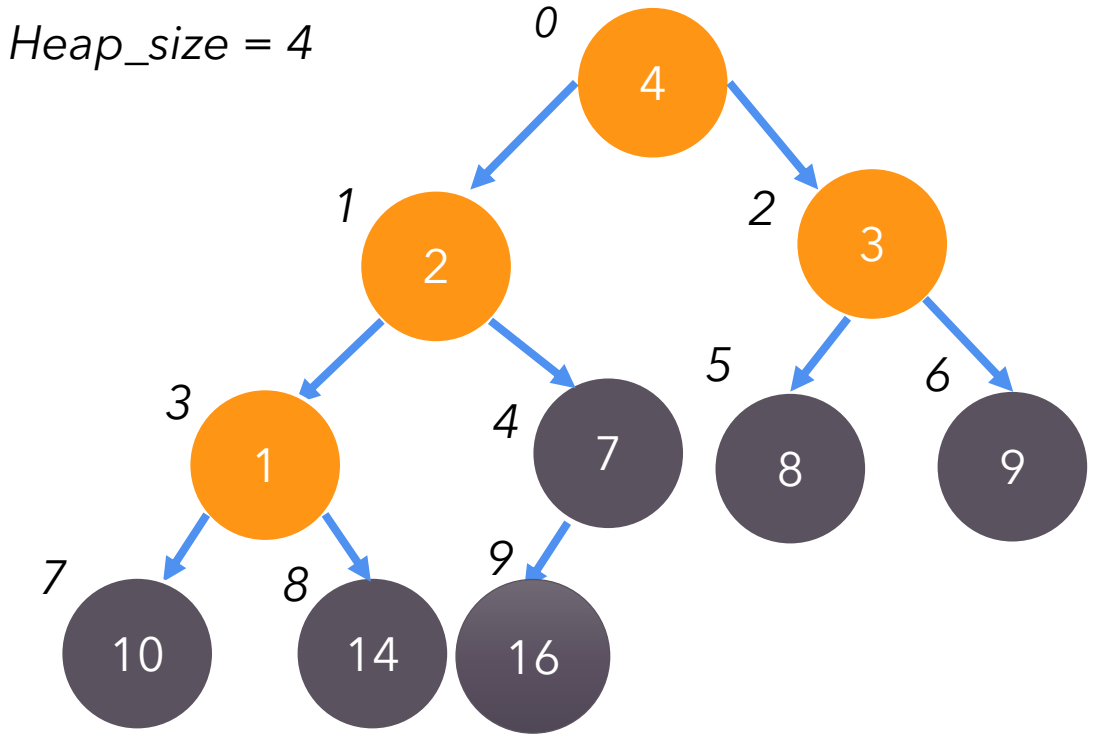
2	4	3	1	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

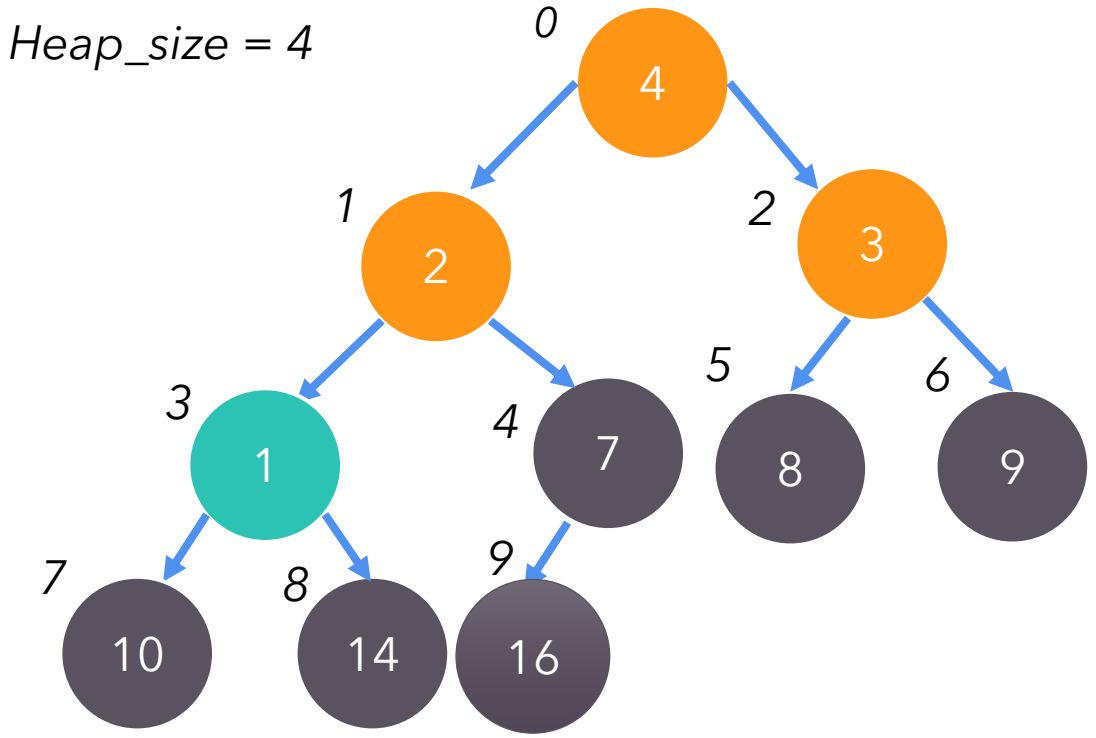
4	2	3	1	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

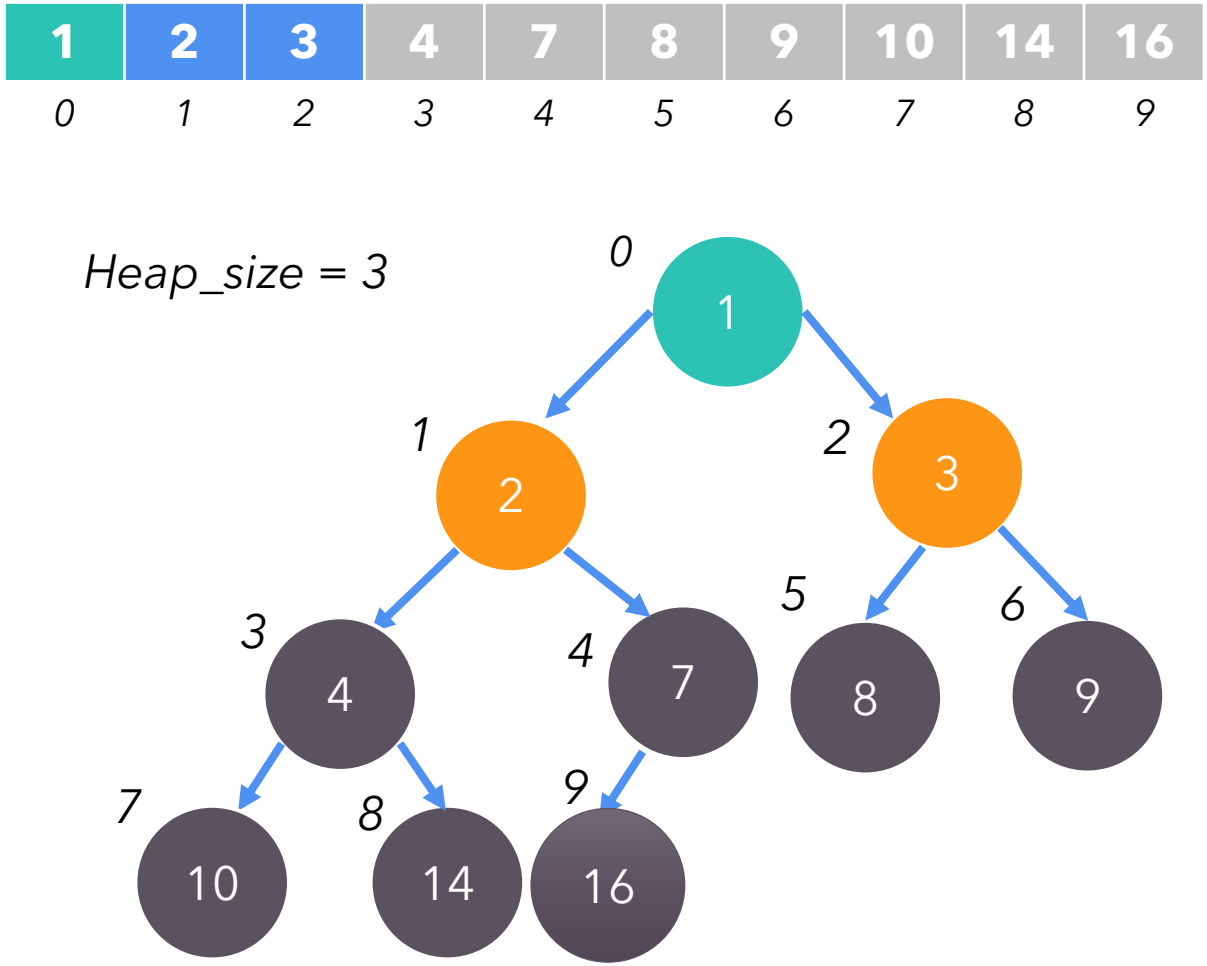
```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

4	2	3	1	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

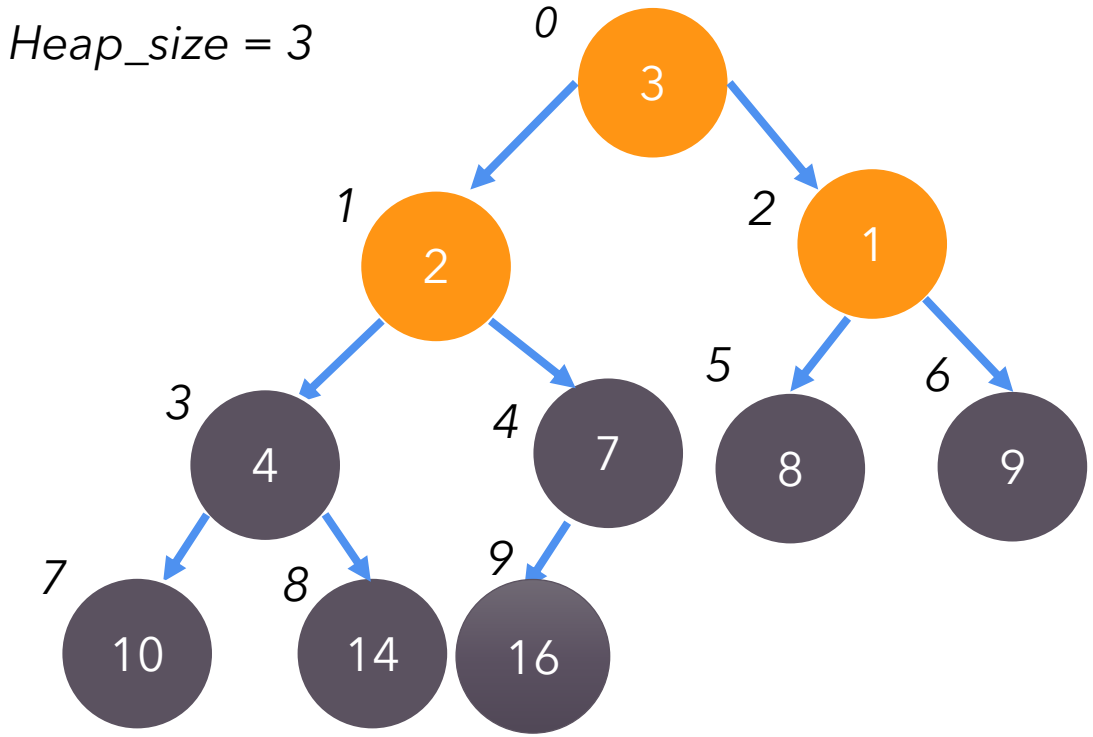
```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

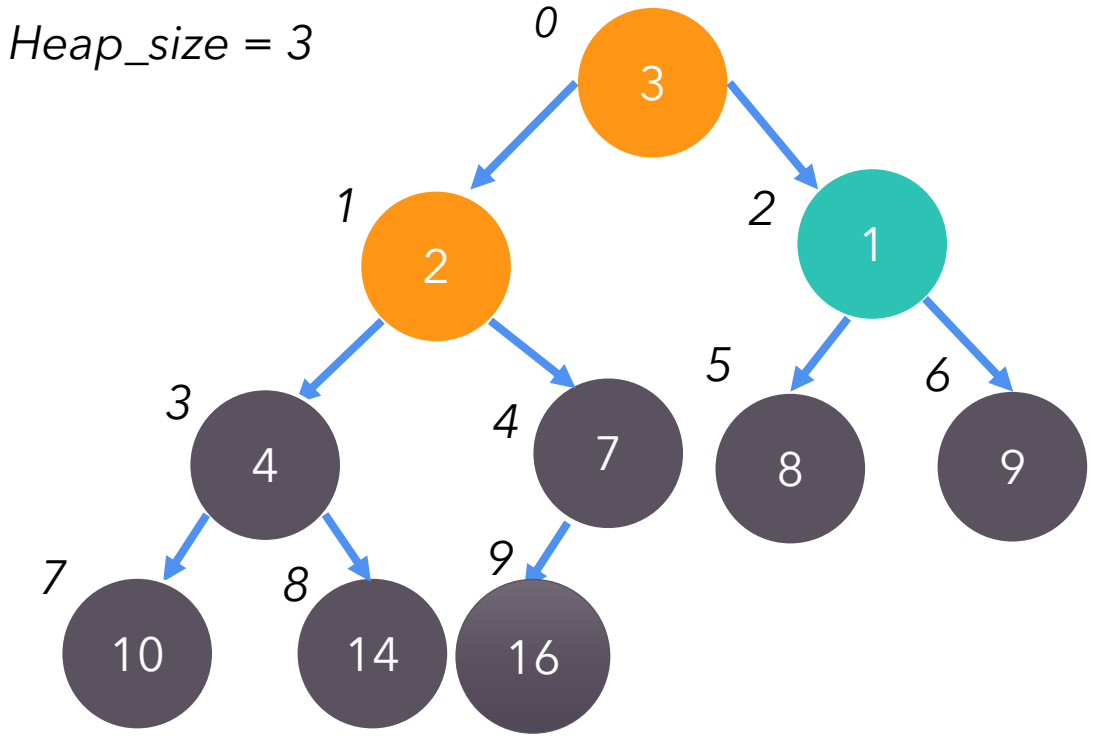
3	2	1	4	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

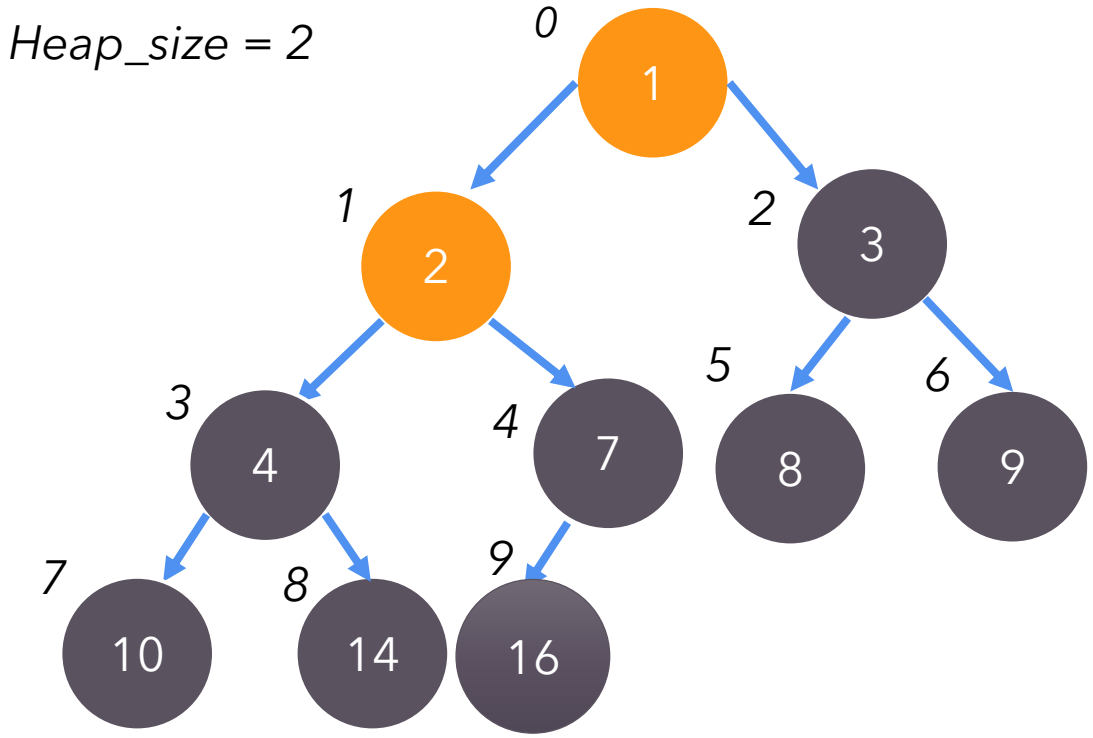
3	2	1	4	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

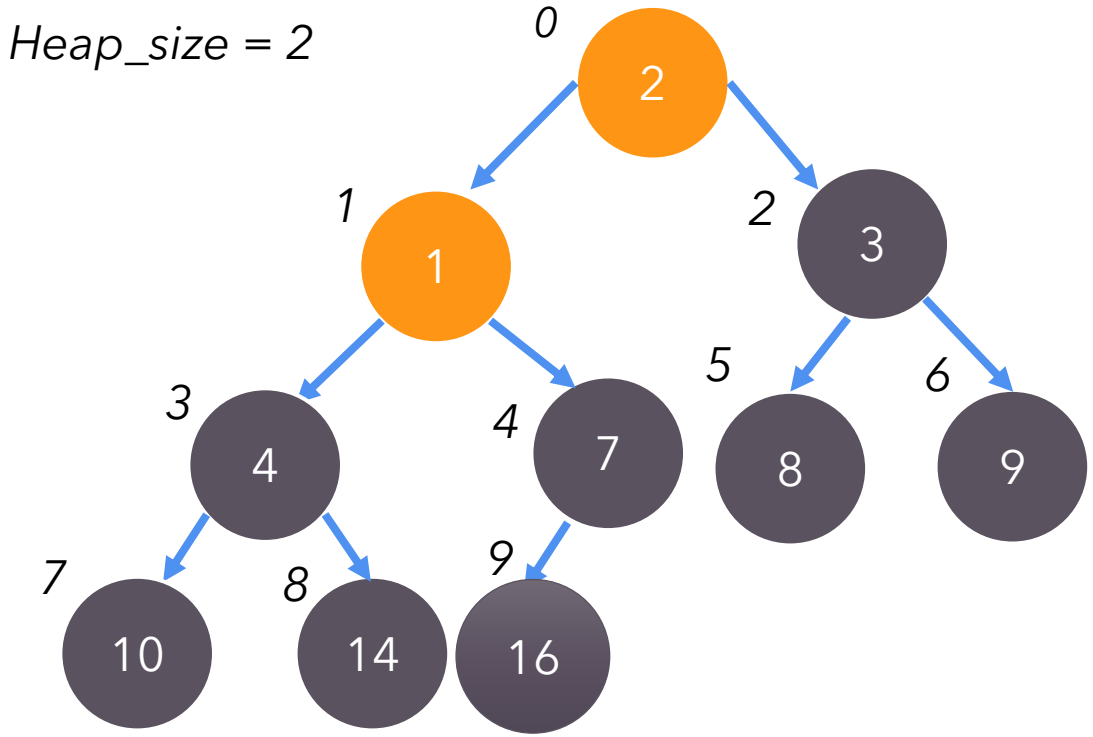
1	2	3	4	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

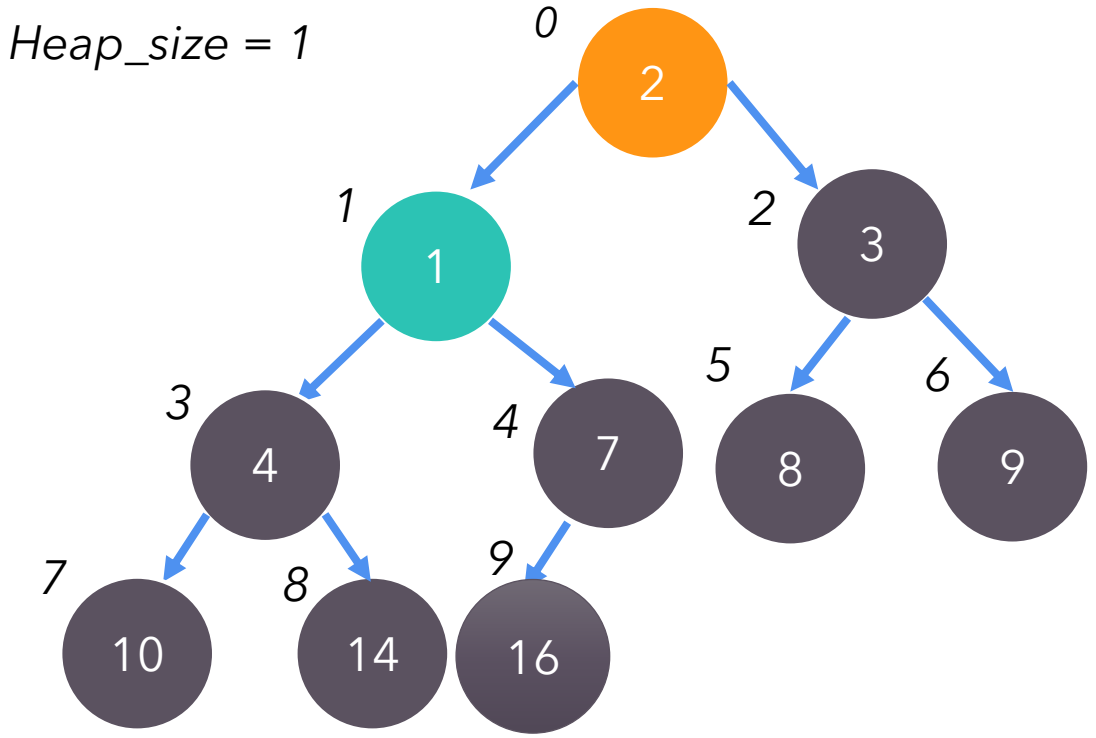
2	1	3	4	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

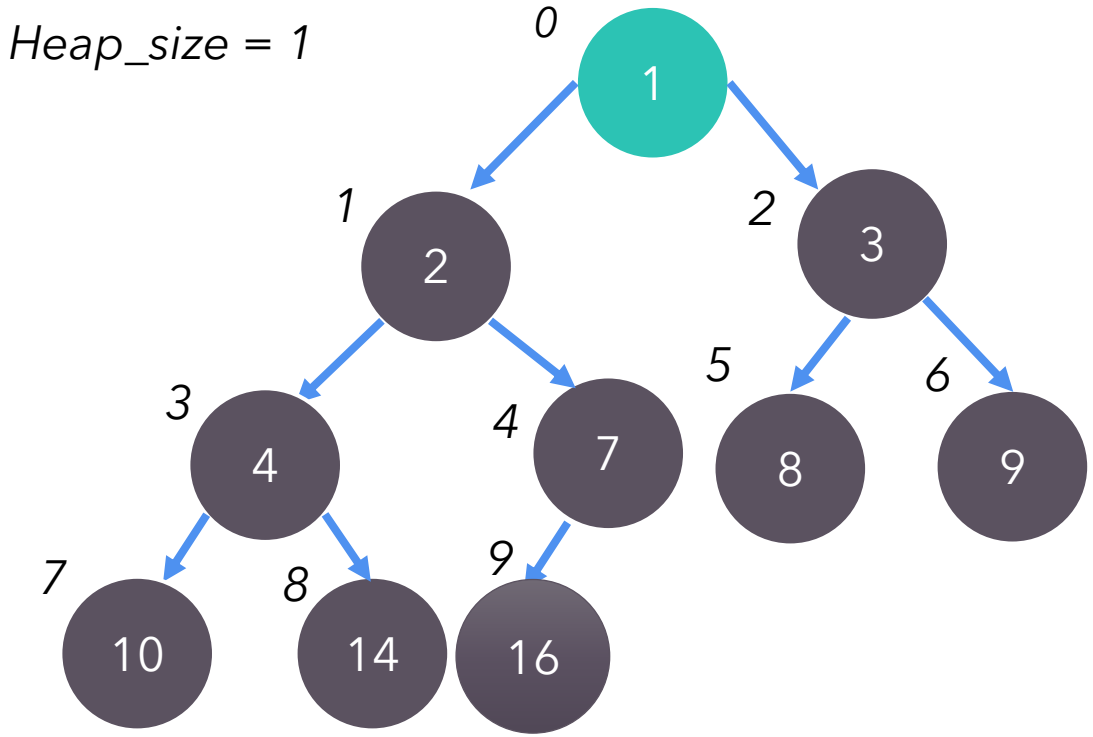
2	1	3	4	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

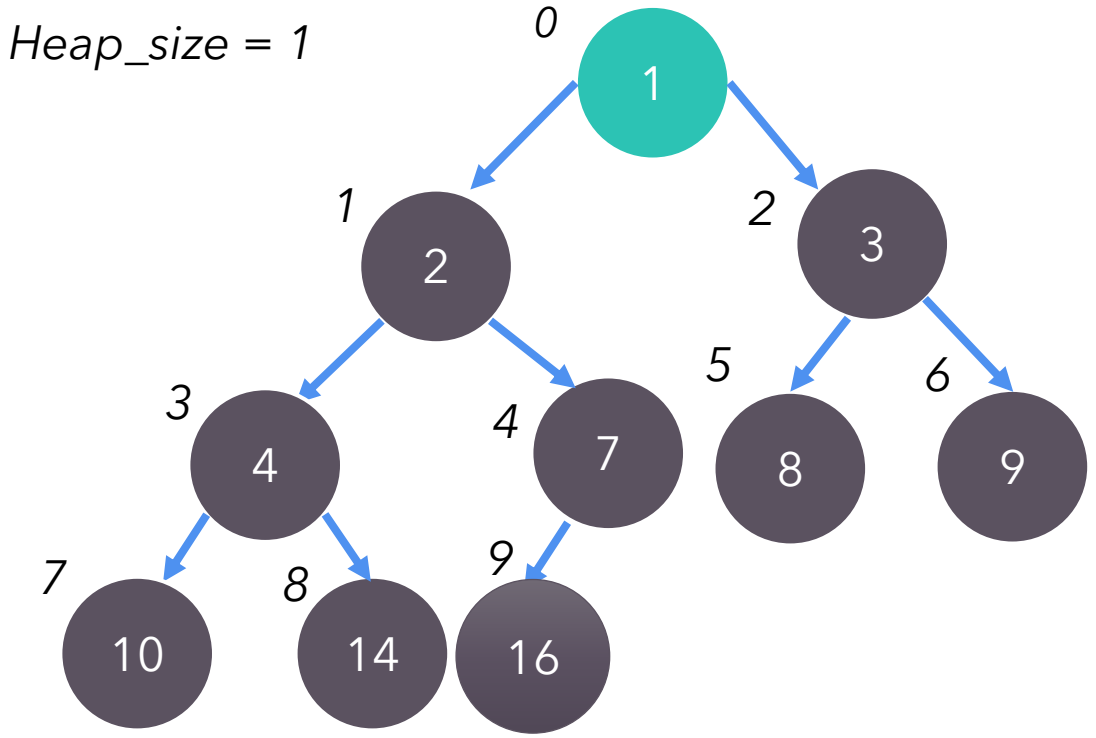
1	2	3	4	7	8	9	10	14	16
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

1	2	3	4	7	8	9	10	14	16
0	1	2	3	4	5	6	7	8	9



Operaciones de un HEAP – HEAPSORT

```
HEAPSORT(A)
1.  BUILD-MAX-HEAP(A)
2.  heap_size = A.length
3.  FOR i=A.length-1 TO 1
4.      temp = A[i]
5.      A[i] = A[0]
6.      A[0] = temp
7.      heap_size--
8.      MAX-HEAPIFY(A,0,heap_size)
```

O(n log n)

O(n log n)

Operaciones de un HEAP

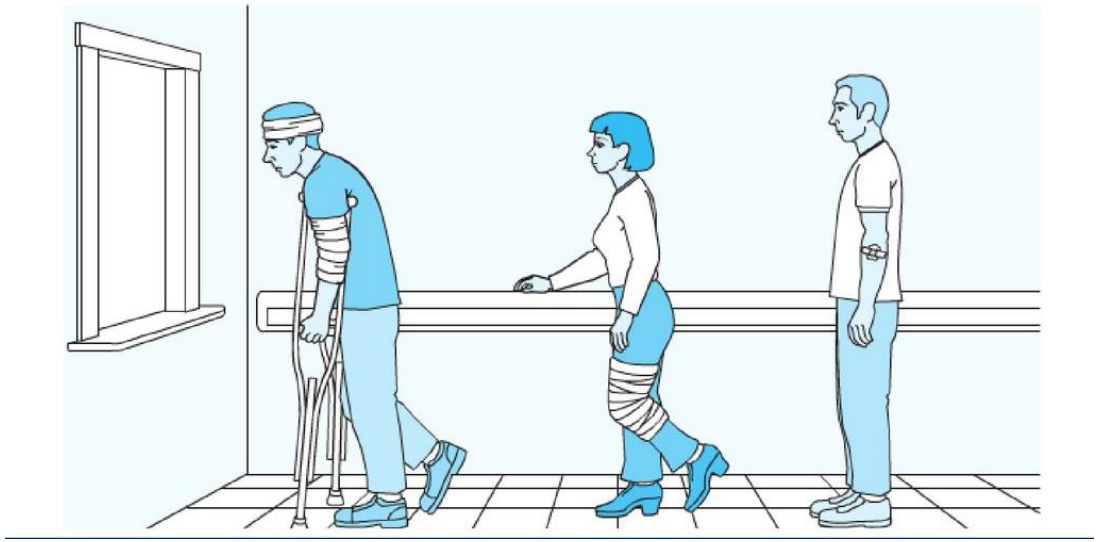
Operación	Descripción	Complejidad
MAX-HEAPIFY	Mantiene la propiedad heap	O(log n)
BUILD-MAX-HEAP	Construye un heap a partir de un arreglo no ordenado	O(n log n)
HEAPSORT	Algoritmo de ordenamiento, organiza los datos en el mismo arreglo	O(n log n)
MAX-HEAP-INSERT	Inserta un dato en el arreglo manteniendo la propiedad HEAP	O(log n)
HEAP-EXTRACT-MAX	Retorna y elimina de la colección el valor máximo	O(log n)
HEAP-MAXIMUM	Retorna el valor máximo	O(1)

Operaciones de un HEAP

Una de las principales aplicaciones de HEAP son las colas de prioridad, la cual soporta las siguientes operaciones:

- MAX-HEAP-INSERT: inserta un dato en el arreglo manteniendo la propiedad HEAP- equivalente a la operación ENQUEUE
- HEAP-EXTRACT-MAX: retorna y elimina de la colección el valor máximo – equivalente a la operación DEQUEUE
- HEAP-MAXIMUM: retorna el valor máximo – equivalente a la operación FIRST

A diferencia de la QUEUE, una cola de prioridad se organiza de acuerdo con la clave



Operaciones de un HEAP – MAX-HEAP-INSERT

MAX-HEAP-INSERT: inserta un dato en el arreglo manteniendo la propiedad HEAP- equivalente a la operación ENQUEUE

Algoritmo:

1. Insertar el dato en la última posición
2. Recorrer la rama hacia arriba hasta encontrar la posición correcta del dato

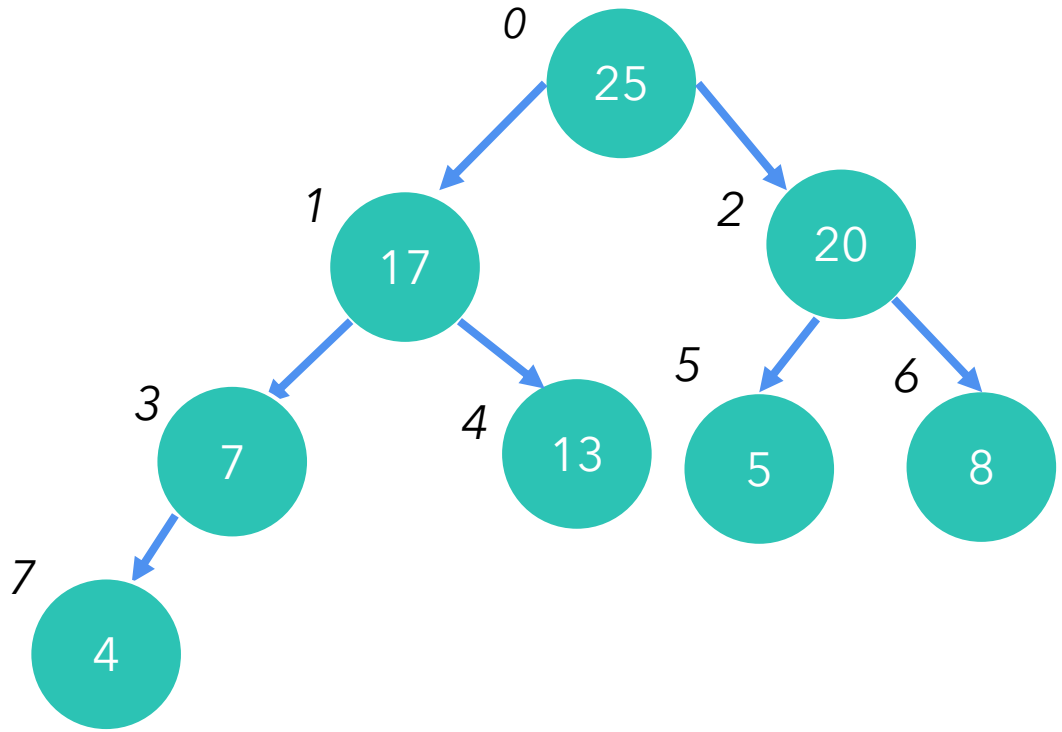
```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

Operaciones de un HEAP – MAX-HEAP-INSERT

```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

MAX-HEAP-INSERT(A,19)
heap_size = 8

25	17	20	7	13	5	8	4		
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*



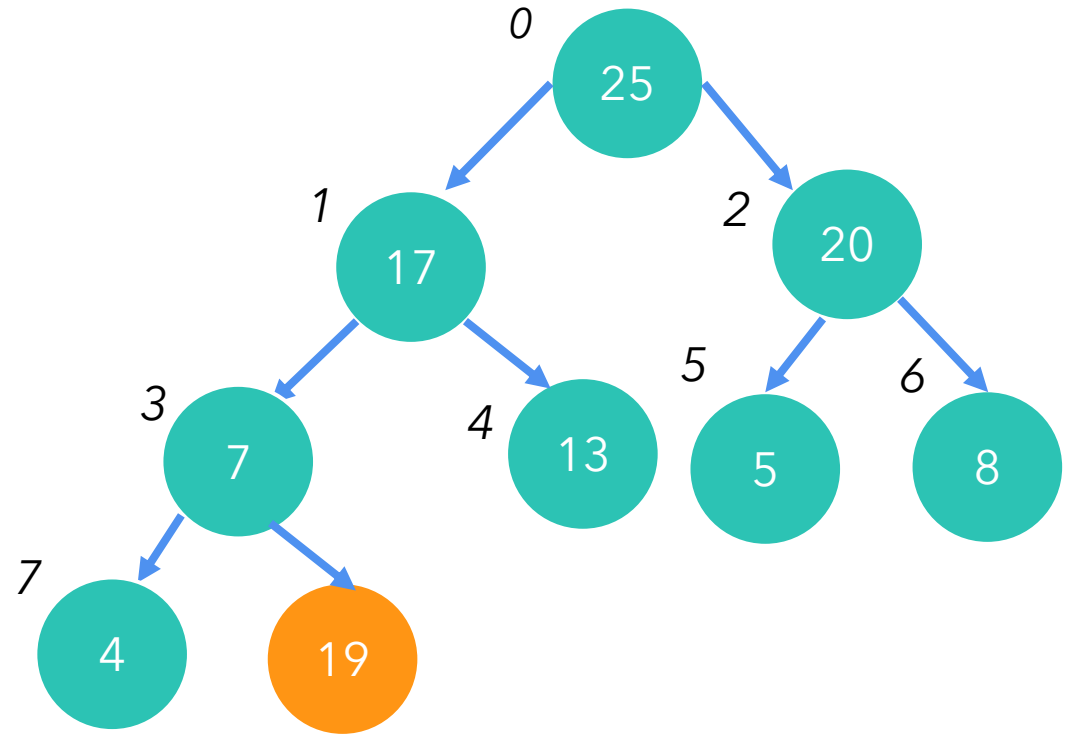
Operaciones de un HEAP – MAX-HEAP-INSERT

```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

MAX-HEAP-INSERT(A,19)

heap_size = 8

25	17	20	7	13	5	8	4	19	
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

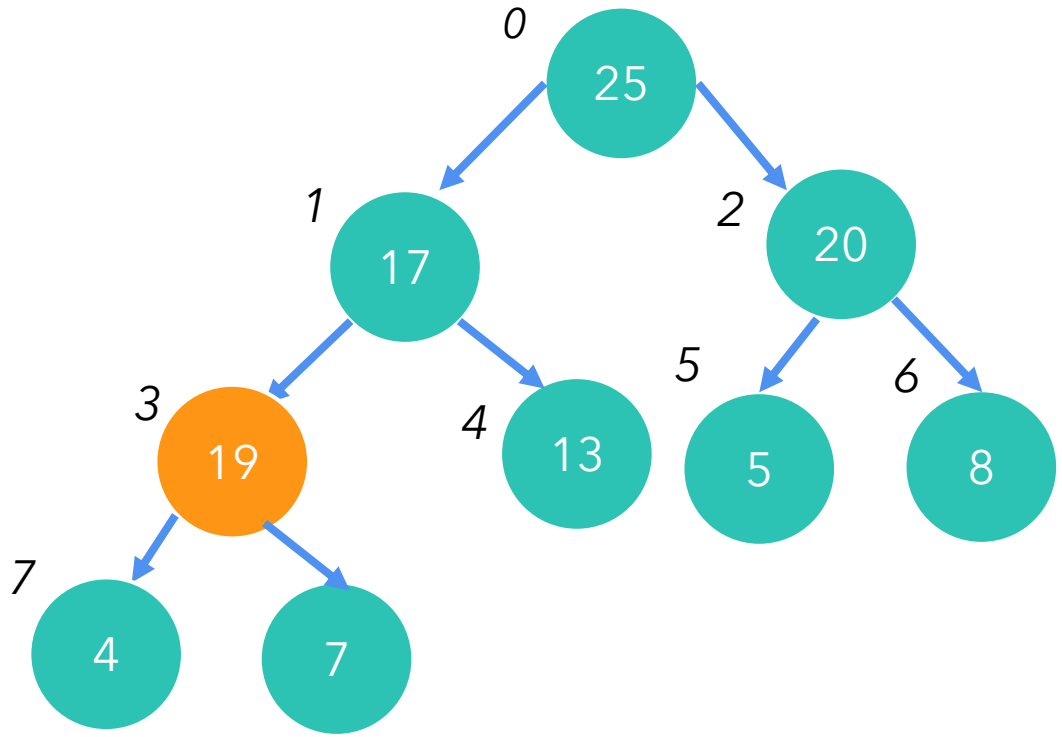


Operaciones de un HEAP – MAX-HEAP-INSERT

```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

MAX-HEAP-INSERT(A,19)
heap_size = 8

25	17	20	19	13	5	8	4	7	
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

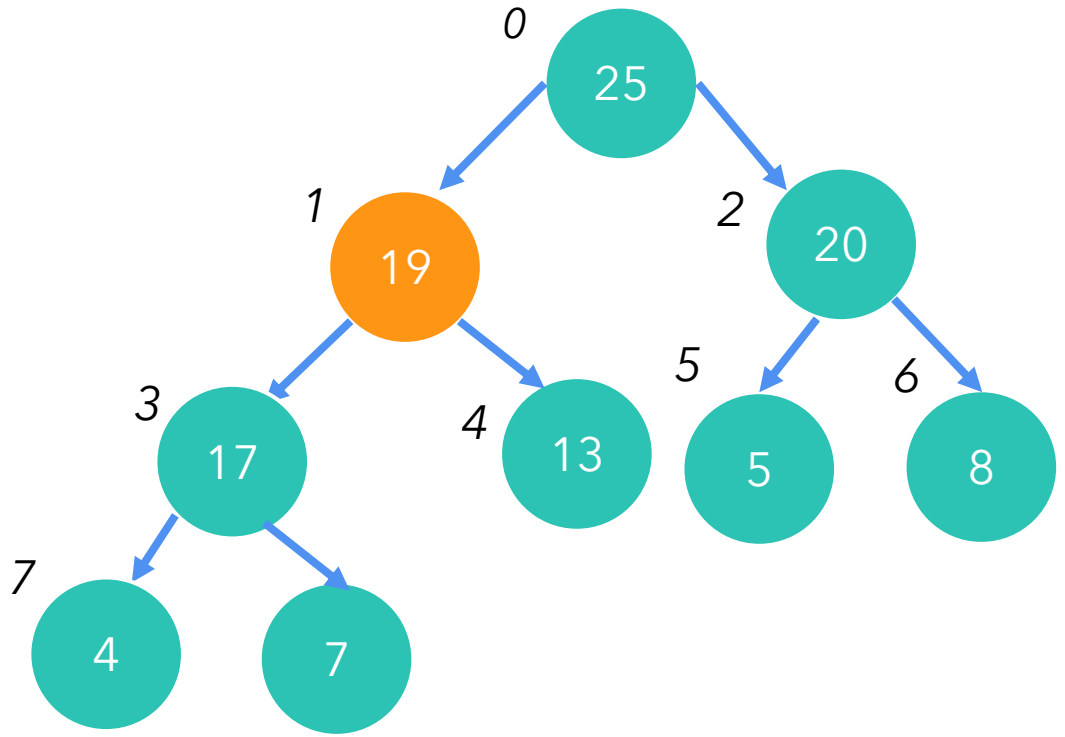


Operaciones de un HEAP – MAX-HEAP-INSERT

```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

MAX-HEAP-INSERT(A,19)
heap_size = 8

25	19	20	17	13	5	8	4	7	
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*



Operaciones de un HEAP – MAX-HEAP-INSERT

```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

MAX-HEAP-INSERT(A,19)
heap_size = 8

Operaciones de un HEAP – MAX-HEAP-INSERT

```
MAX-HEAP-INSERT(A, k)
1. heap_size = heap_size+1
2. i = heap_size
3. A[i] = k
4. WHILE i>0 & A[Parent(i)] < A[i]
5.     temp = A[Parent(i)]
6.     Parent[i] = A[i]
7.     A[i] = temp
8.     i = Parent(i)
```

O(1)

O(log n)

Operaciones de un HEAP

Operación	Descripción	Complejidad
MAX-HEAPIFY	Mantiene la propiedad heap	O(log n)
BUILD-MAX-HEAP	Construye un heap a partir de un arreglo no ordenado	O(n log n)
HEAPSORT	Algoritmo de ordenamiento, organiza los datos en el mismo arreglo	O(n log n)
MAX-HEAP-INSERT	Inserta un dato en el arreglo manteniendo la propiedad HEAP	O(log n)
HEAP-EXTRACT-MAX	Retorna y elimina de la colección el valor máximo	O(log n)
HEAP-MAXIMUM	Retorna el valor máximo	O(1)

Operaciones de un HEAP – MAX-EXTRACT-MAX

HEAP-EXTRACT-MAX: retorna y elimina de la colección el valor máximo – equivalente a la operación DEQUEUE

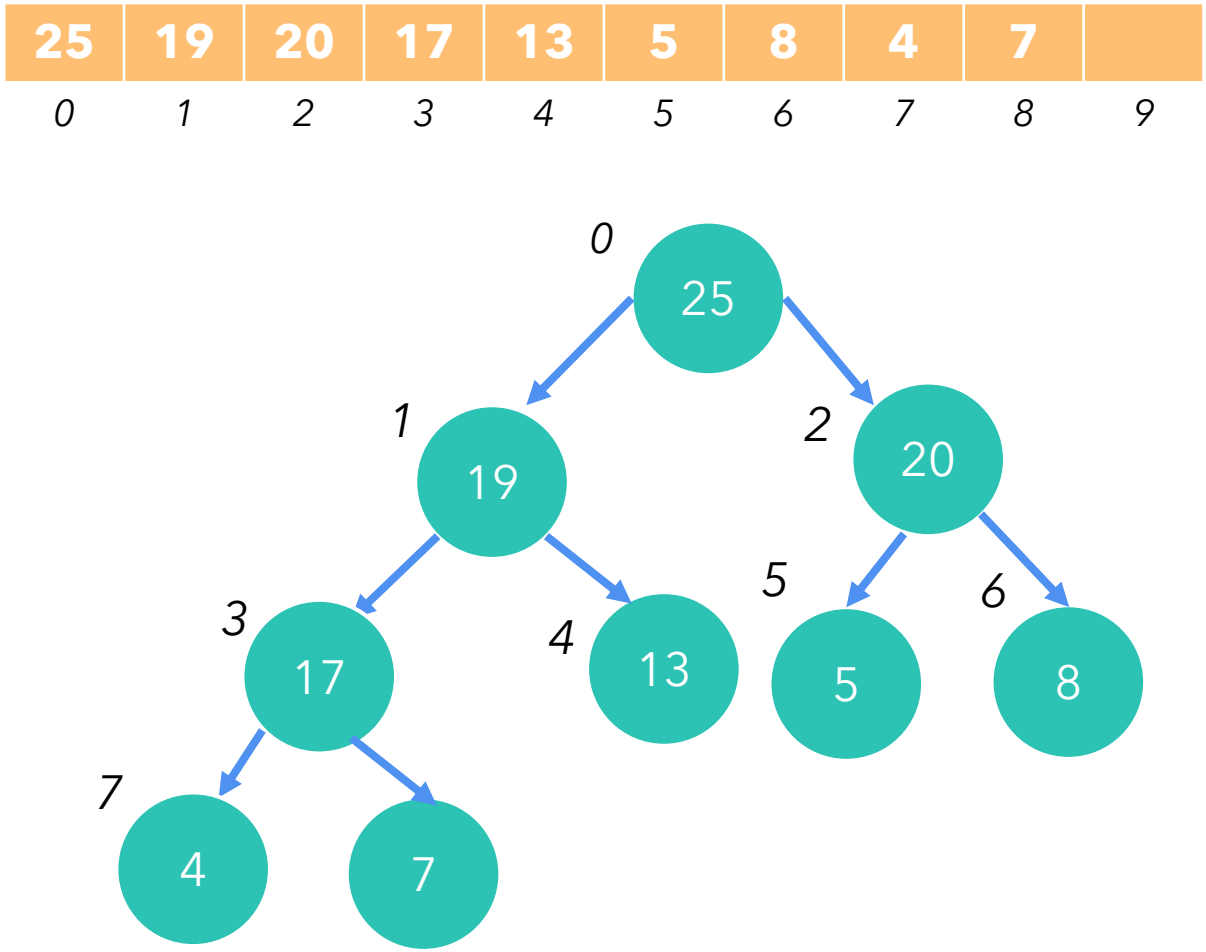
Algoritmo:

1. Intercambia la posición 0 con la ultima posición del HEAP
2. Decrece el tamaño del HEAP en 1
3. Aplica el método HEAPIFY
4. Retorna el valor intercambiado

```
HEAP-EXTRACT-MAX
1. max = A[0]
2. A[0] = A[heap_size]
3. heap_size - -
4. MAX-HEAPIFY(A,0)
5. return max
```

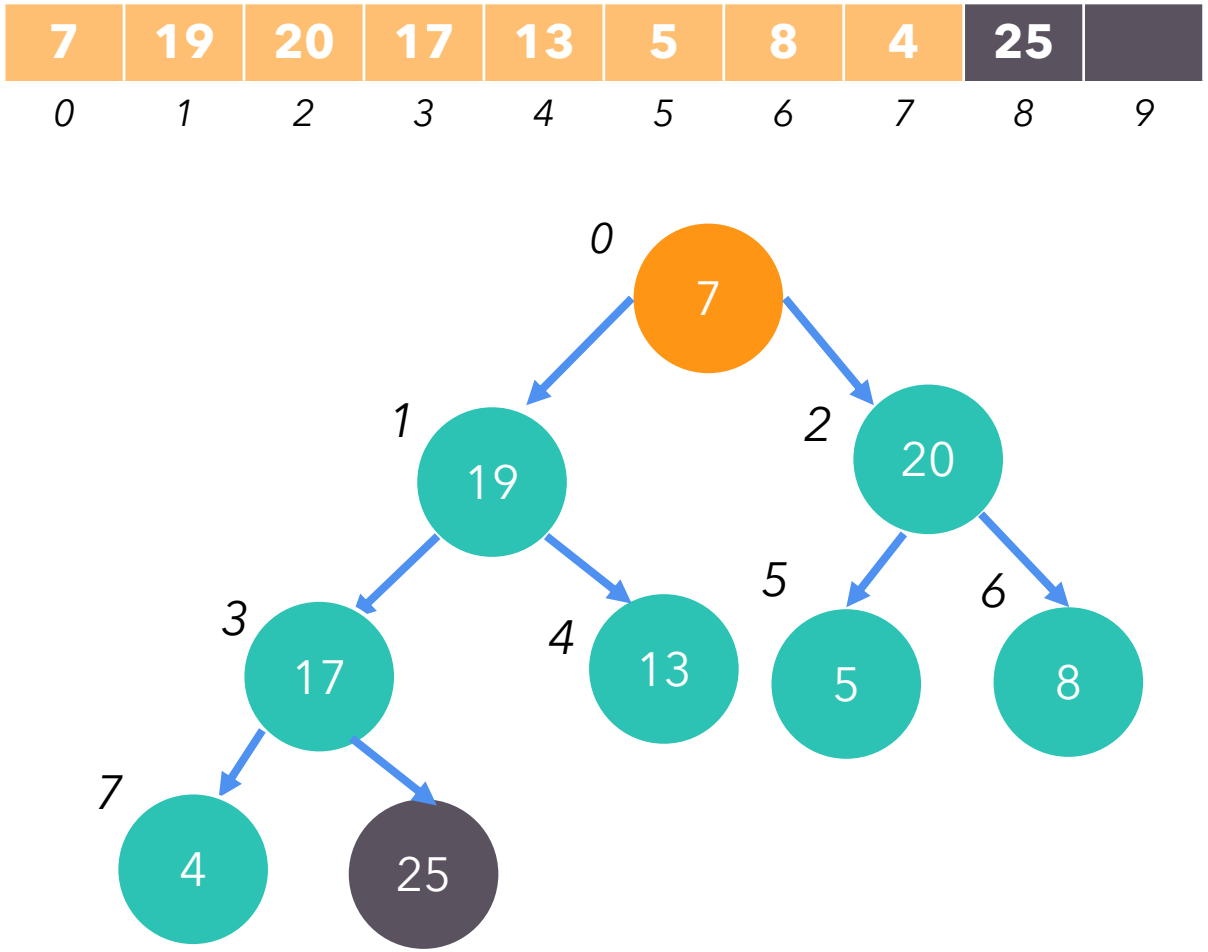
Operaciones de un HEAP – MAX-EXTRACT-MAX

```
HEAP-EXTRACT-MAX
1. max = A[0]
2. A[0] = A[heap_size]
3. heap_size - -
4. MAX-HEAPIFY(A,0)
5. return max
```



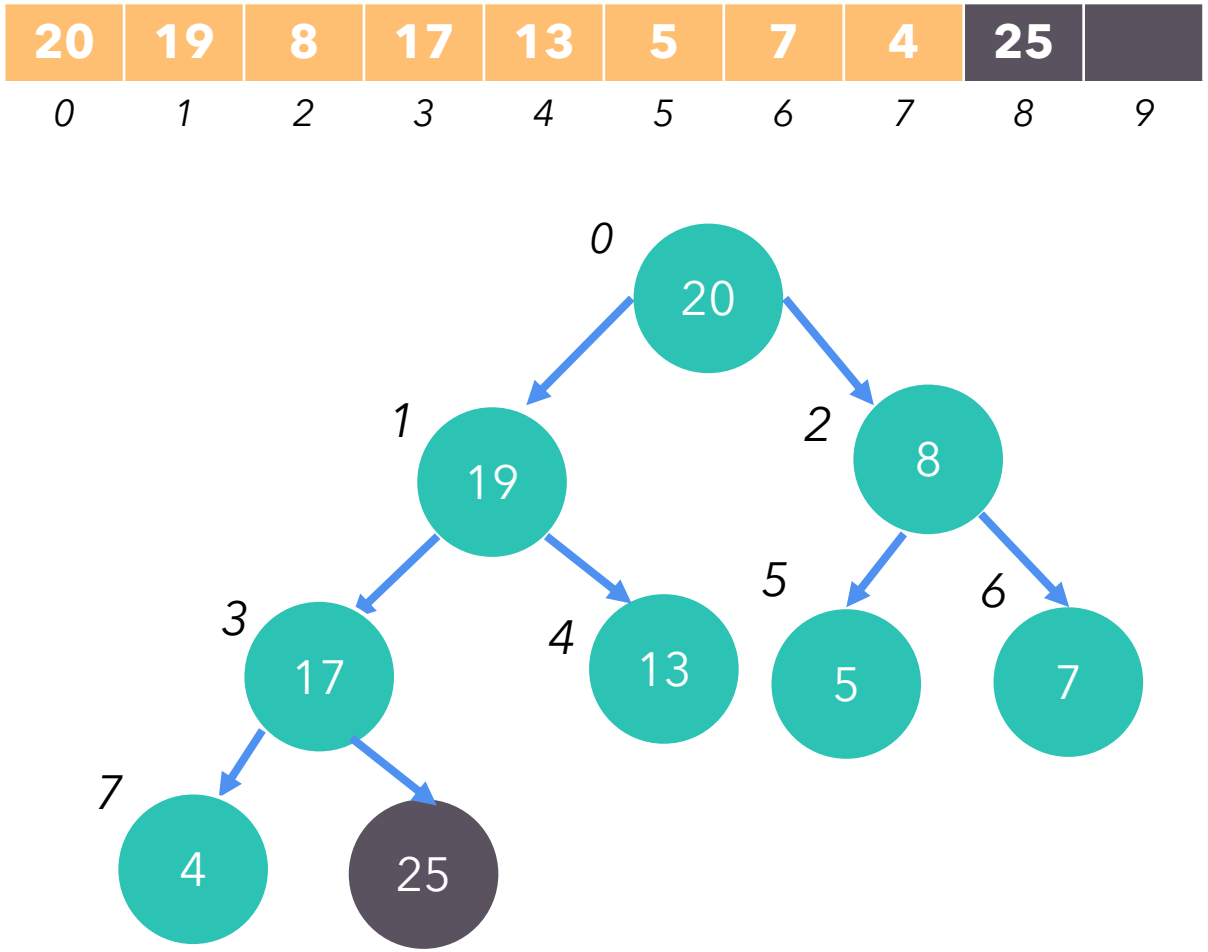
Operaciones de un HEAP – MAX-EXTRACT-MAX

```
HEAP-EXTRACT-MAX
1. max = A[0]
2. A[0] = A[heap_size]
3. heap_size - -
4. MAX-HEAPIFY(A,0)
5. return max
```



Operaciones de un HEAP – MAX-EXTRACT-MAX

```
HEAP-EXTRACT-MAX
1. max = A[0]
2. A[0] = A[heap_size]
3. heap_size - -
4. MAX-HEAPIFY(A,0)
5. return max
```



Operaciones de un HEAP – MAX-EXTRACT-MAX

```
HEAP-EXTRACT-MAX
1. max = A[0]
2. A[0] = A[heap_size]
3. heap_size - -
4. MAX-HEAPIFY(A,0)
5. return max
```

O(1)

O(log n)

Operaciones de un HEAP

Operación	Descripción	Complejidad
MAX-HEAPIFY	Mantiene la propiedad heap	O(log n)
BUILD-MAX-HEAP	Construye un heap a partir de un arreglo no ordenado	O(n log n)
HEAPSORT	Algoritmo de ordenamiento, organiza los datos en el mismo arreglo	O(n log n)
MAX-HEAP-INSERT	Inserta un dato en el arreglo manteniendo la propiedad HEAP	O(log n)
HEAP-EXTRACT-MAX	Retorna y elimina de la colección el valor máximo	O(log n)
HEAP-MAXIMUM	Retorna el valor máximo	O(1)

Operaciones de un HEAP – MAXIMUM

HEAP-MAXIMUM: retorna el valor máximo – equivalente a la operación FIRST

- El valor máximo siempre esta en la posición 0

```
HEAP-MAXIMUM
1. return A[0]
```