Colas de Prioridad

TDA - Cola de Prioridad

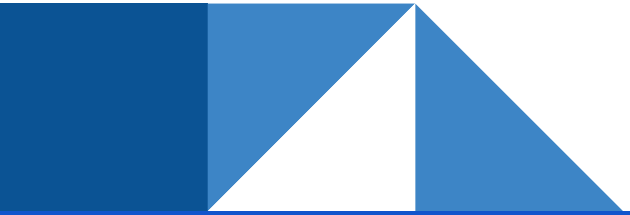
Las dos operaciones básicas que debe soportar una cola de prioridad son:

- Insertar un elemento.
- Devolver el elemento de mayor prioridad.

ED
ESTRUCTURAS DE DATOS

Pregunta

Si la prioridad en la cola es el elemento que lleve más tiempo en ella, ¿cómo la implementarías?



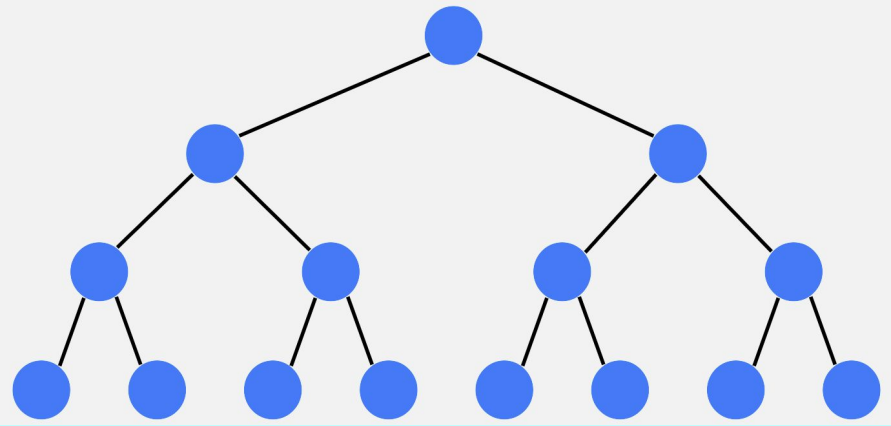
Pregunta

¿Cómo implementarías las filas en los juegos de Six Flags o las filas de un banco?

Árboles binarios llenos

Árbol binario de profundidad *d* con 2^*d* -1 nodos.

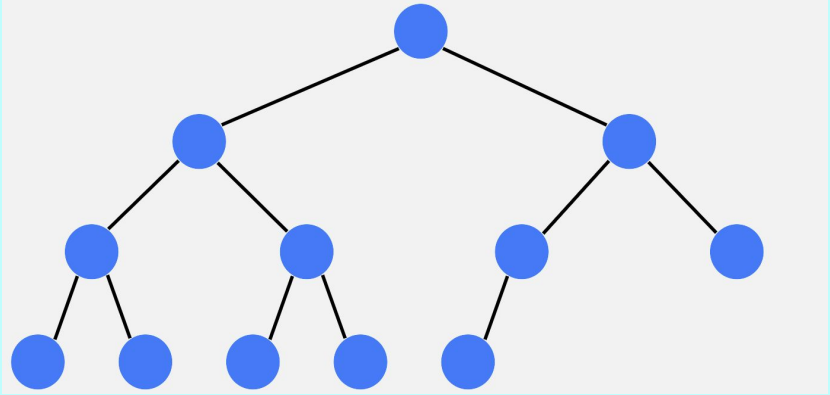
Árbol binario con todos sus niveles llenos.



ED
ESTRUCTURAS DE DATOS

Árbol binario completo

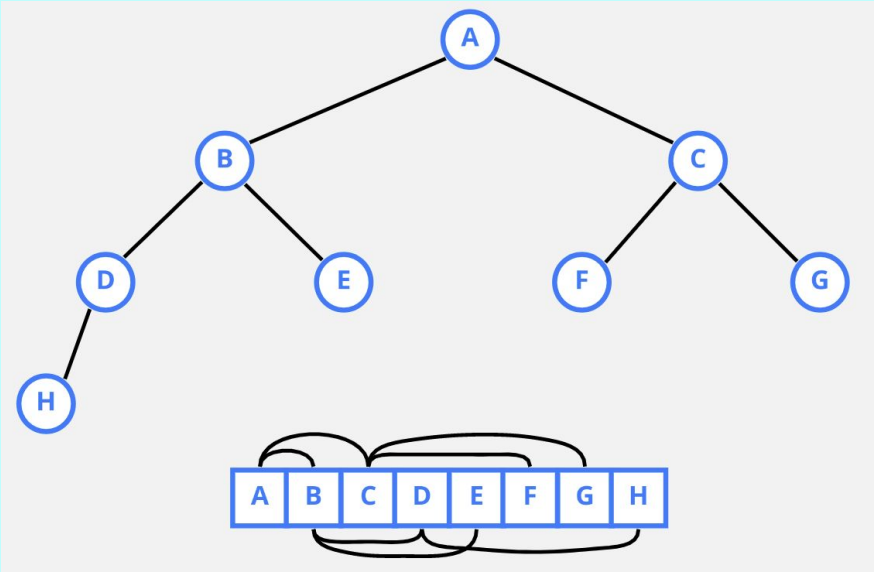
Árbol binario de profundidad *d* en el que al menos los primeros *d* - 1 niveles están llenos y que en el último nivel todos los nodos no vacíos van del lado izquierdo.



ED
ESTRUCTURAS DE DATOS

Implementación de árboles completos

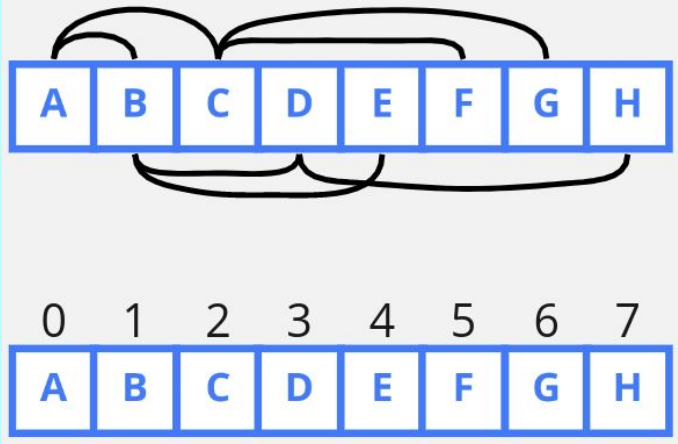
Aunque es un árbol, dado que los elementos se van insertando en un orden establecido, lo más común es implementarlos con arreglos.



ED
ESTRUCTURAS DE DATOS

Para el elemento en el índice *i* del arreglo se tiene que:

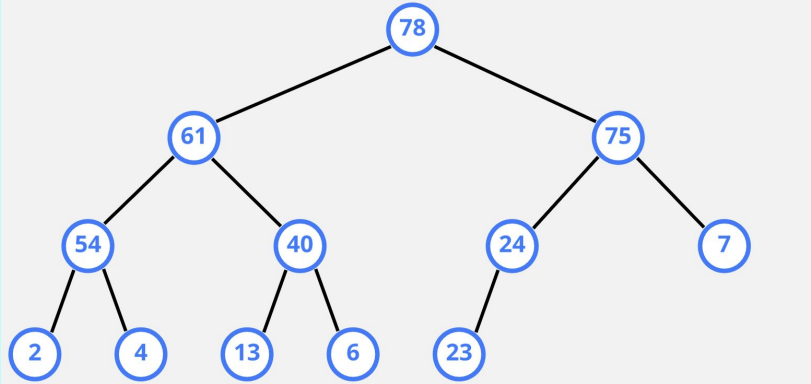
- En el índice 2*i* + 1 está su hijo izquierdo.
- En el índice 2*i* + 2 está su hijo derecho.
- En el índice ⌊ (*i* - 1) / 2 ⌋ está su padre.



ED
ESTRUCTURAS DE DATOS

Montículos máximos (Max-heap)

Árbol binario completo en el que cada nodo es mayor que sus hijos.
Solo está permitido acceder al elemento en la **raíz** del montículo.



Nota que el elemento en la raíz es mayor que cualquier nodo.

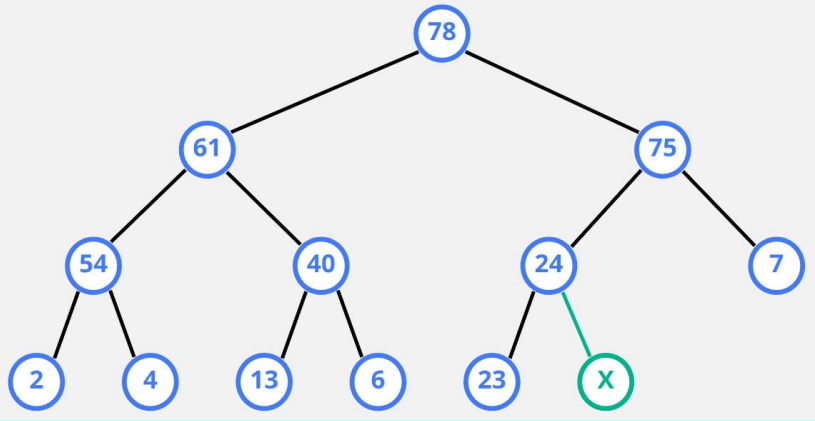
No es un árbol binario de búsqueda

ED
ESTRUCTURAS DE DATOS

Algoritmo para insertar en un Max-heap

Para insertar un nuevo elemento **X** en un max-heap:

1. Lo insertamos en la siguiente posición libre del árbol.
2. Reacomodamos **X** hacia arriba.

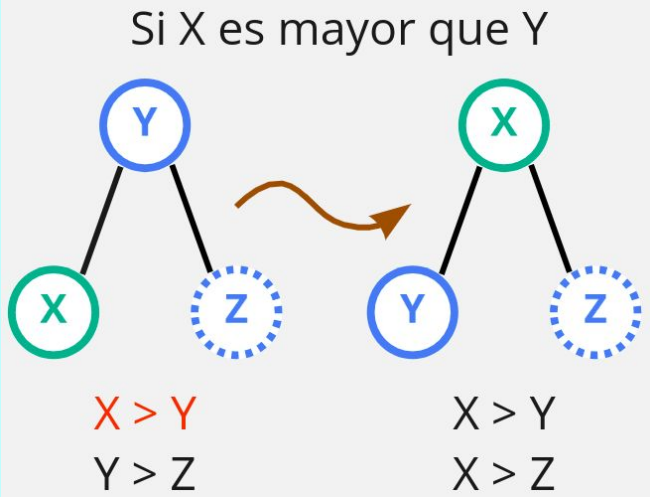


ED
ESTRUCTURAS DE DATOS

Reacomodar hacia arriba en un Max-heap (heapify)

Para reacomodar hacia arriba un elemento X en un max-heap:

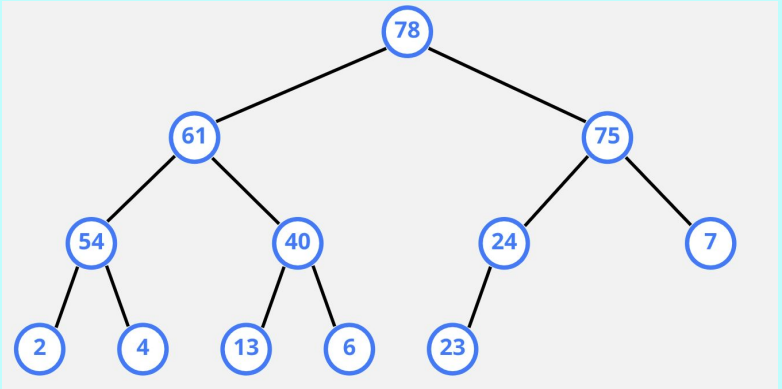
1. Si no tiene padre, terminamos.
2. Comparamos X con su padre Y:
   a. Si X < Y, terminamos.
   b. Si X > Y, intercambiamos a X y Y de posición.
   c. Seguimos reacomodando a X (Volvemos al paso 1).



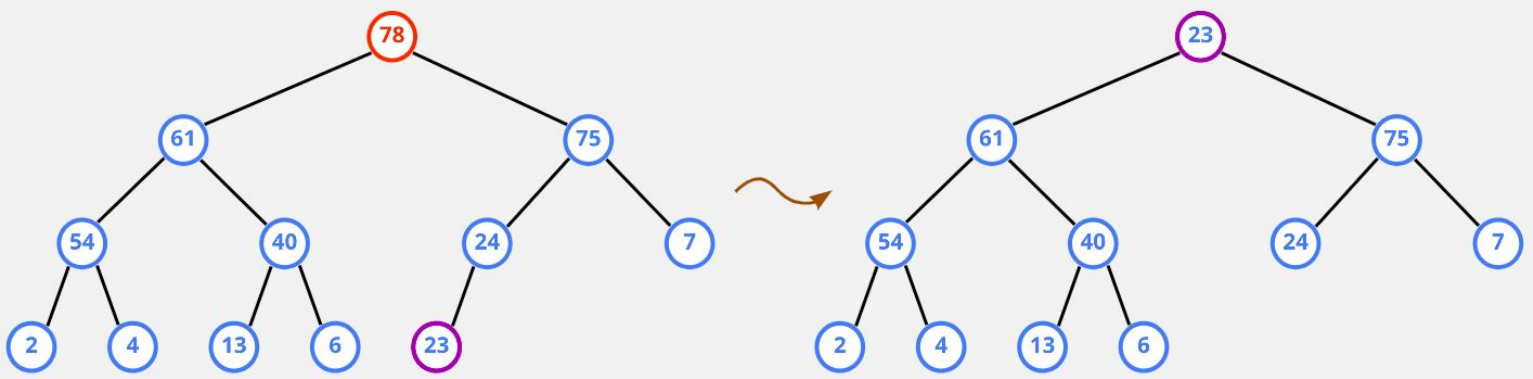
ED
ESTRUCTURAS DE DATOS

Ejercicio

Inserta el número 77 en el siguiente max-heap.



Algoritmo para eliminar de un Max-heap



Para eliminar el máximo elemento de un max-heap:

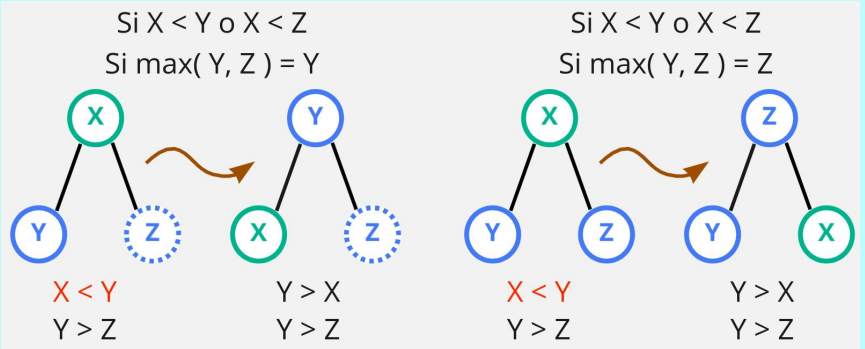
1. Reemplazamos la raíz del árbol con el elemento en el índice **n** - 1.
2. Vaciamos el índice **n** - 1.
3. Reacomodamos la raíz hacia abajo.

ED
ESTRUCTURAS DE DATOS

Reacomodar hacia abajo en un Max-heap (heapify)

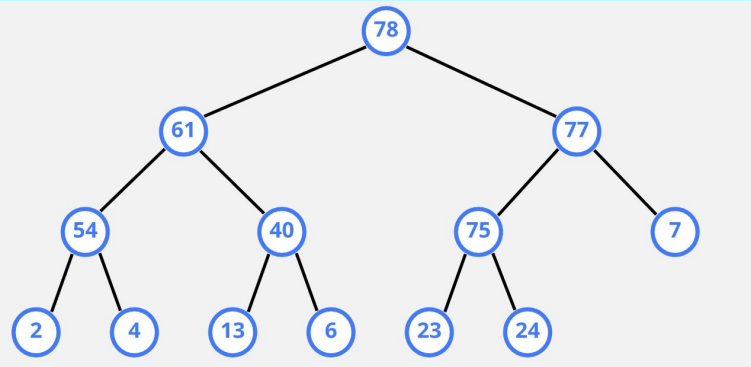
Para reacomodar hacia abajo un elemento **X** en un max-heap:

1. Si **X** no tiene hijos, terminamos.
2. Comparamos **X** con sus hijos **Y** y **Z**:
   a. Si **X > Y** y **X > Z**, terminamos.
   b. Si **X < Y** o **X < Z**, intercambiamos a **X** con el hijo que valga max(**Y**, **Z**).
   c. Seguimos reacomodando a **X** (Volvemos al paso 1).



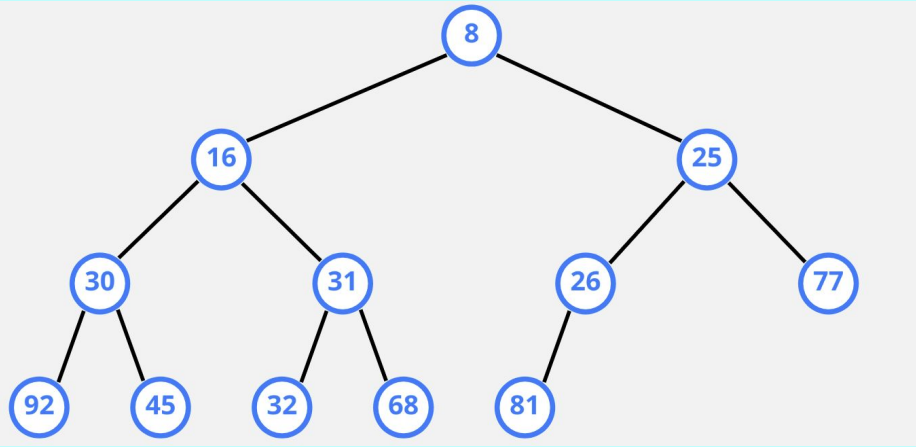
Ejercicio

Elimina el máximo del siguiente max-heap.



Montículos mínimos (Min-heap) - 1964 Williams

Árbol binario completo en el que cada nodo es menor que sus hijos.
Solo está permitido acceder al elemento en la **raíz** del montículo.



Nota que el elemento en la raíz es menor que cualquier nodo.

ED
ESTRUCTURAS DE DATOS

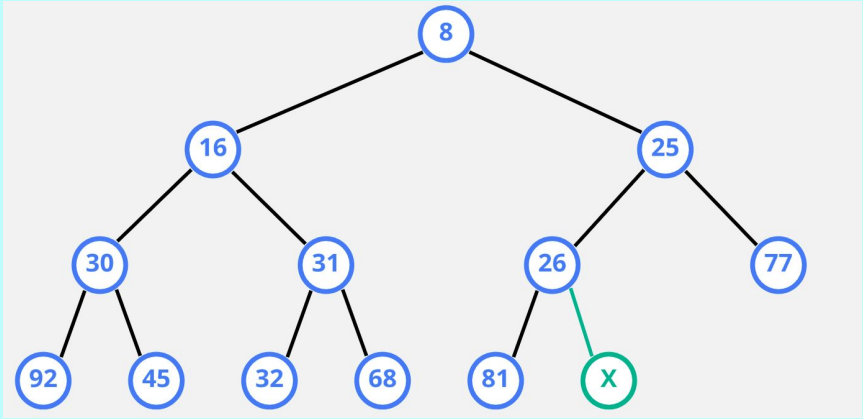
Simulador de Min-heaps

Pedro Ulises Cervantes González - Facultad de Ciencias

Algoritmo para insertar en un Min-heap

Para insertar un nuevo elemento **X** en un min-heap:

1. Lo insertamos en la siguiente posición libre del árbol.
2. Reacomodamos **X** hacia arriba.



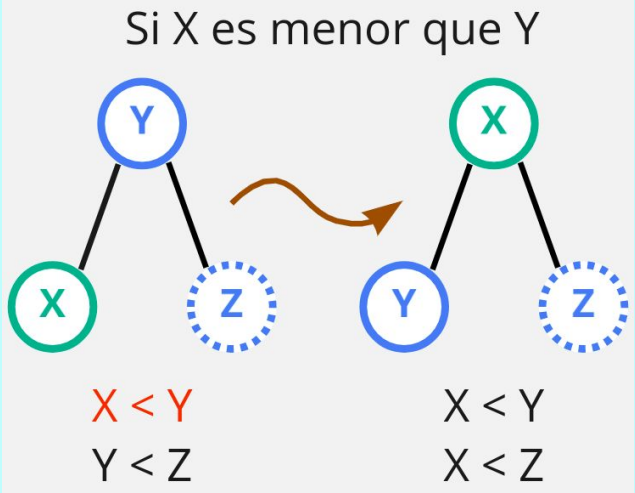
Lo único que cambia respecto a un Max-heap es el algoritmo para reacomodar hacia arriba.

ED
ESTRUCTURAS DE DATOS

Reacomodar hacia arriba en un Min-heap (heapify)

Para reacomodar hacia arriba un elemento X en un min-heap:

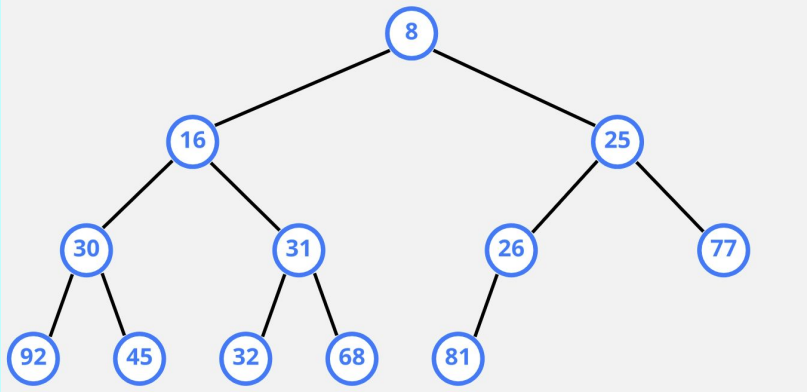
1. Si no tiene padre, terminamos.
2. Comparamos X con su padre Y:
   a. Si X > Y, terminamos.
   b. Si X < Y, intercambiamos a X y Y de posición.
   c. Seguimos reacomodando a X (Volvemos al paso 1).



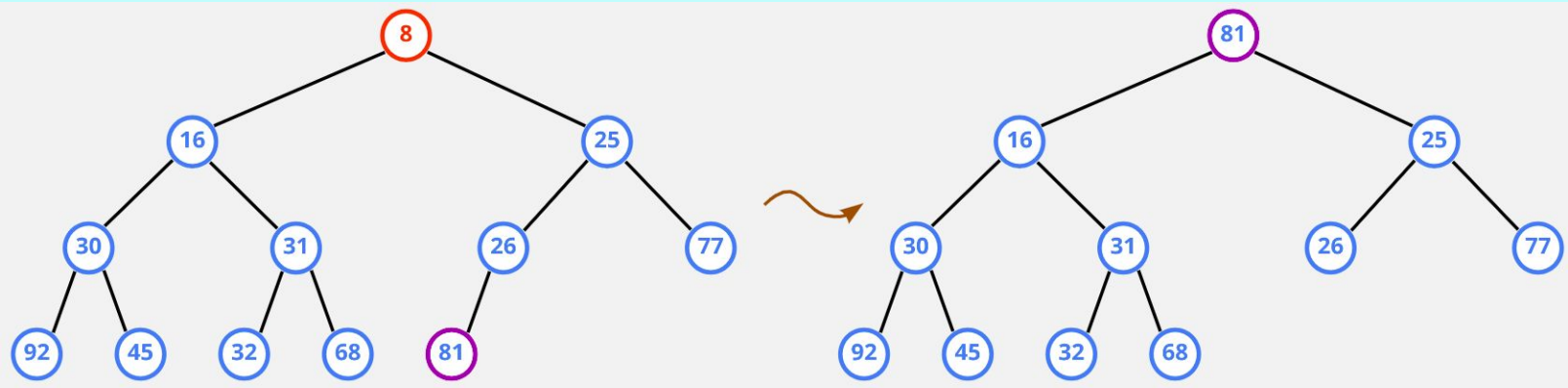
ED
ESTRUCTURAS DE DATOS

Ejercicio

Inserta el número 4 en el siguiente min-heap.



Algoritmo para eliminar de un Min-heap



Para eliminar el mínimo elemento de un min-heap:

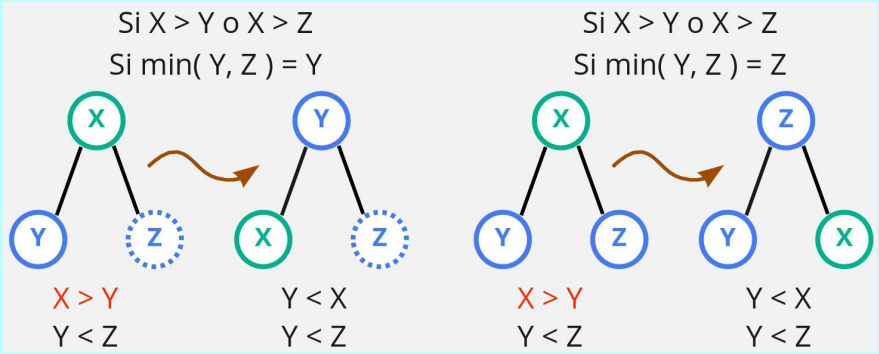
1. Reemplazamos la raíz del árbol con el elemento en el índice **n** - 1.
2. Vaciamos el índice **n** - 1.
3. Reacomodoamos la raíz hacia abajo.

ED
ESTRUCTURAS DE DATOS

Reacomodar hacia abajo en un Min-heap (heapify)

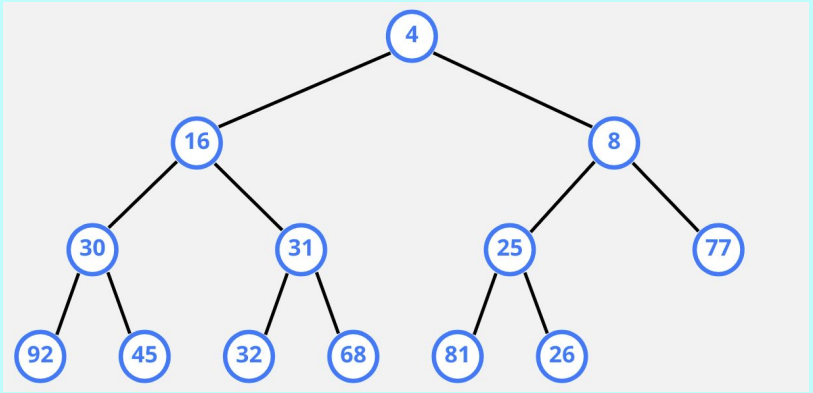
Para reacomodar hacia abajo un elemento X en un min-heap:

1. Si X no tiene hijos, terminamos.
2. Comparamos X con sus hijos Y y Z:
   a. Si X < Y y X < Z, terminamos.
   b. Si X > Y o X > Z, intercambiamos a X con el hijo que valga min(Y, Z).
   c. Seguimos reacomodando a X (Volvemos al paso 1).



Ejercicio

Elimina 3 veces el mínimo del siguiente min-heap.



Construir un Montículo a partir de un arreglo

Si tenemos un arreglo A lleno con tamaño **n** y lo queremos convertir en un min-heap o max-heap, hay dos opciones:

Desde el índice **i** = 1 hasta el índice **n** - 1:
    Reacomodar A[i] hacia arriba.

→ **O(n log n)**

Desde el índice **i** = [**n** / 2] - 1 hasta el índice 0:
    Reacomodar A[i] hacia abajo.

→ **O(n)**

Los reacomodos deben corresponder con el tipo de heap.

ED
ESTRUCTURAS DE DATOS

Obtener los k elementos de mayor prioridad en una colección

Algoritmo	Tiempo de ejecución
Iterar k veces la colección sin ordenar.	O(kn)
Ordenar la colección y devolver los primeros k.	O(nlogn + k)
Insertarlos en un BST balanceado y devolver los primeros k de un recorrido inorden.	O(nlogn + logn + k)
Insertarlos en un heap y devolver los primeros k.	O(n + klogn)

ED
ESTRUCTURAS DE DATOS

Heapsort

1 Convertir el arreglo en un max-heap.
2 Mientras el heap tenga elementos:
3 Sacar el máximo.
4 Reducir el tamaño del heap.
5 Colocar el máximo en la posición que liberó el heap.

¿Qué tiempo toma ejecutar este algoritmo?

ED
ESTRUCTURAS DE DATOS

Simulador de Heapsort

Pedro Ulises Cervantes González - Facultad de Ciencias