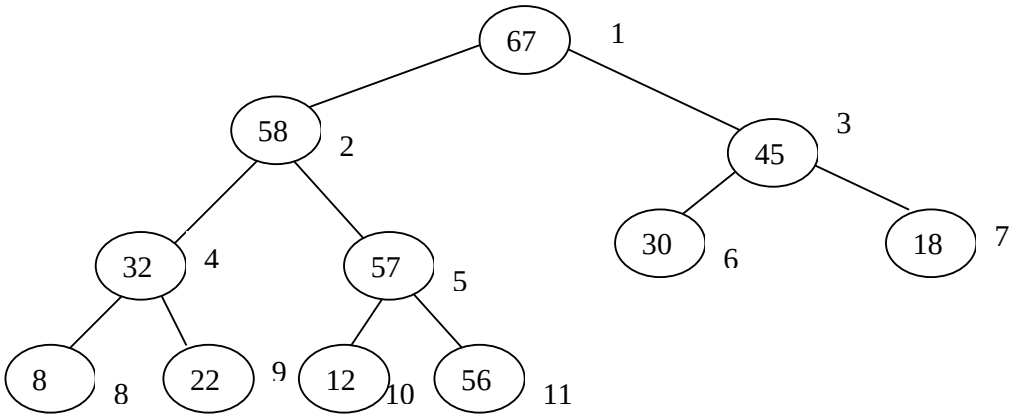
Binary Heap o Montículo Binario

Un montículo es un árbol binario semicompleto (completamente lleno excepto el último nivel) que satisface las siguientes condiciones:

- cada nodo debe tener mayor o igual prioridad a la de sus hijos (si tiene alguno).
- el nodo de mayor prioridad es la raíz.
- cualquier subárbol es un montículo binario.

La propiedad de ordenación parcial es clave: para cualquier camino directo entre una hoja y la raíz, los valores aparecen en orden creciente. Por otro lado, un árbol binario semicompleto es sumamente regular, presenta la ventaja de que puede ser implementado mediante un arreglo sin necesidad de enlaces. Esto es: para cualquier elemento i, el hijo izquierdo está en la posición 2*i, el hijo derecho en la posición (2*i+1) y el padre en la posición (i div 2), con div: división entera.



0	1	2	3	4	5	6	7	8	9	10	11
-	67	58	45	32	57	30	18	8	22	12	56

Un heap se va llenando por niveles: primero el nivel 1, luego el 2, etc. Todos los niveles deben estar llenos excepto el último. El nivel i tendrá 2^i nodos.

Algoritmos sobre Heaps:

Los algoritmos básicos que operan sobre montículos tienen las siguientes características:

- operan a lo largo de algún "camino" desde la raíz hasta el fondo del montículo (excepto al *unir* dos o más montículos).
- comienzan haciendo una simple modificación estructural que potencialmente viola las condiciones del montículo, luego lo recorren y modifican para asegurar que dichas condiciones son reestablecidas.

Inserción:

1. agregar el elemento al final del arreglo, lo cual potencialmente viola la condición de montículo.
2. hacer que el elemento recién agregado suba el montículo por el camino hacia la raíz de forma tal que ocupe la posición que le corresponde y en consecuencia, se restaure la condición de montículo.

Eliminación:

1. ubicar la posición i del elemento a eliminar.
2. ubicar al elemento N del heap en la posición i (con esto se elimina al elemento).
3. decrementar el total de elementos N del arreglo.
4. "bajar" el valor desde la posición i por el camino hacia una hoja.

Ejercicios:

1.- Suponiendo la implementación basada en arrays:

a) escribir el algoritmo para insertar un nuevo elemento en un heap.
b) escribir un algoritmo para eliminar el elemento más pequeño del heap.
c) escribir un algoritmo para eliminar el elemento más grande del heap.
d) escribir un algoritmo para eliminar cualquier elemento del heap (si existe).
e) escribir un algoritmo que verifique la condición de heap de un árbol binario.

2.- Describir una representación para heaps tree basada en listas lineales enlazadas:

a) establecer ventajas y desventajas respecto a la implementación en arrays.
b) repetir los ejercicios del punto 1, esta vez para implementación dinámica.

3.- Con la lista de claves: 20, 73, 42, 11, 80, 39, 72, 30, 100, 46, 88, 32, 21
Generar:

a) un árbol binario lleno por amplitud
b) un ABB
c) un AVL
d) un Heap.