UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Universidad del Perú, Decana de América

Facultad de Ingeniería de Sistemas e Informática

EJERCICIOS CON COLAS PRIORITARIAS

GRUPO DE TRABAJO 3

ASIGNATURA: Estructura de Datos II

INTEGRANTES:

- ALEJO CARNICA, Bryan Martin 17200256
- CHÁVEZ SILUPÚ, Erick Alberto 17200267
- CUELLO APAZA, Alexander Gabriel 17200269
- CORDOVA SANDOVAL, Rafael Anthony 17200268

DOCENTE: Javier Antonio Prudencio Vidal.

ESCUELA ACADÉMICO-PROFESIONAL: Ingeniería de Software.

LIMA - PERÚ

2022

LÁMINA 30:

- Dibuje el estado del árbol de un montículo mínimo después de agregar estos elementos:
  - 6, 50, 11, 25, 42, 20, 104, 76, 19, 55, 88, 2

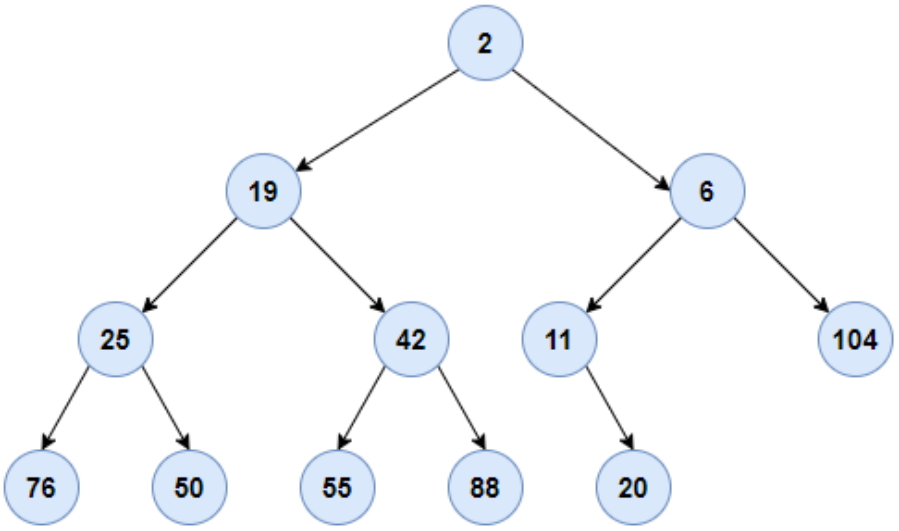
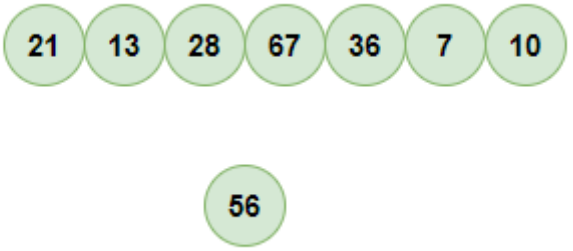


LÁMINA 31:

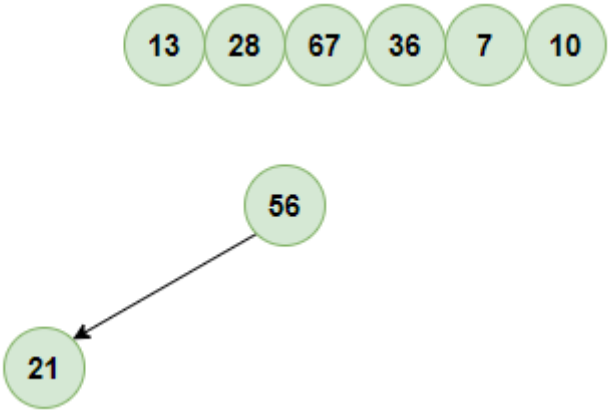
Dado el montículo de la figura, verifique como queda luego de insertar las siguientes claves

56 21 13 28 67 36 07 10

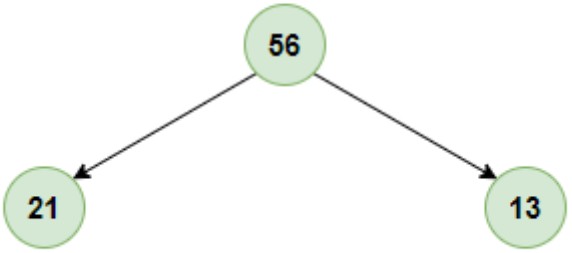
Ubicamos el 56:



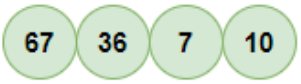
Ahora ubicamos el 21:

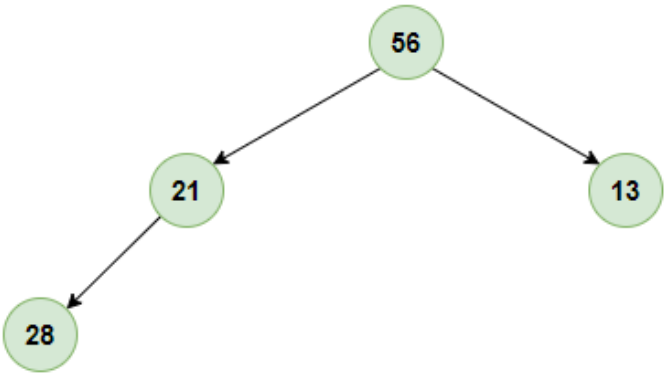


Luego el 13:

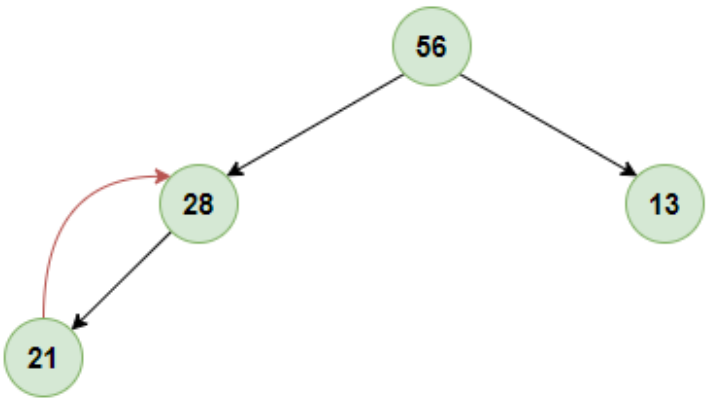


Luego el 28:

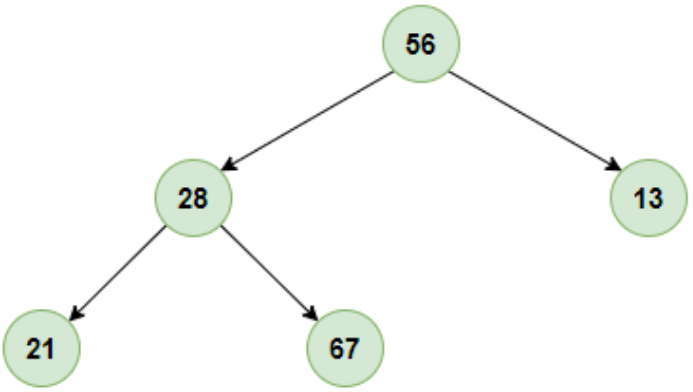




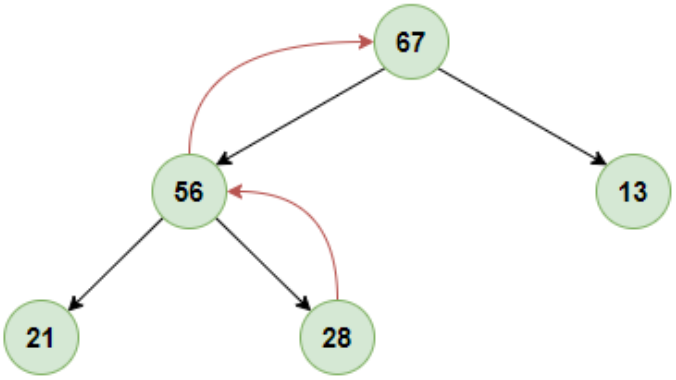
Para que cumpla las propiedades de un árbol max, se ‘burbujea’ el 28 hasta la posición requerida:



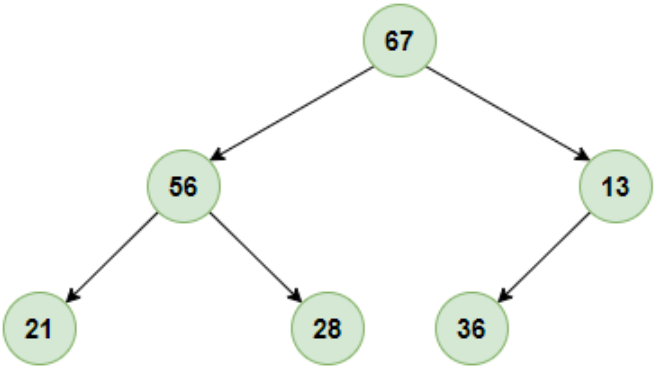
Ahora ubicamos el 67:



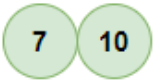
Así como en el caso anterior, burbujeamos el 67 para que cumpla la posición correcta:

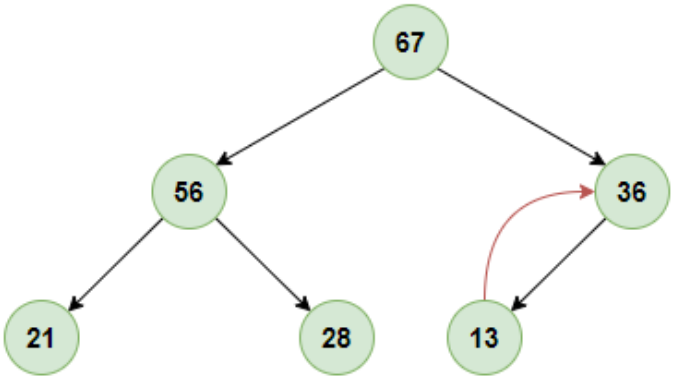


Ahora ubicamos el 36:



Burbujeamos para que cumpla la propiedad:





Ahora con el 7 y por último con el 10:

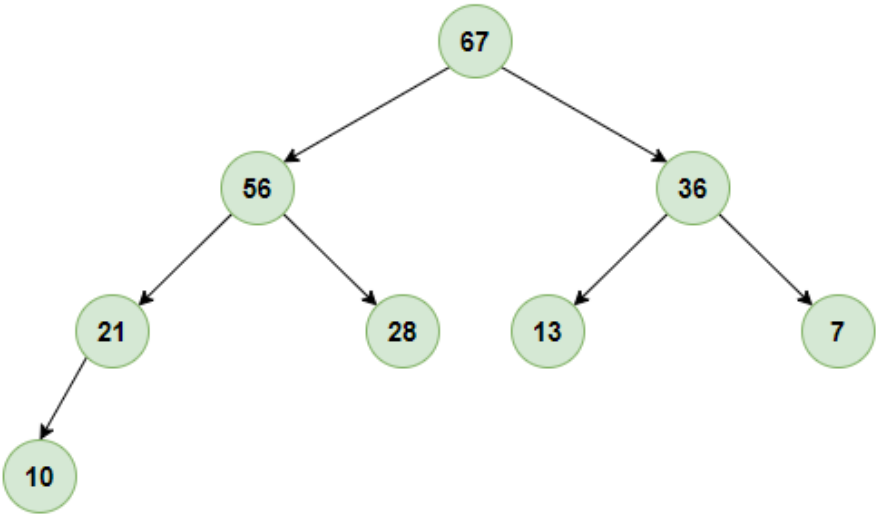
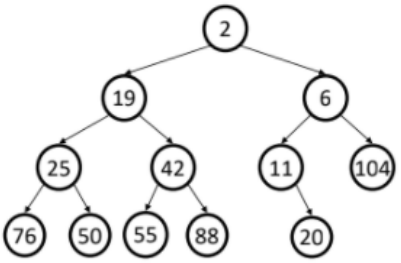
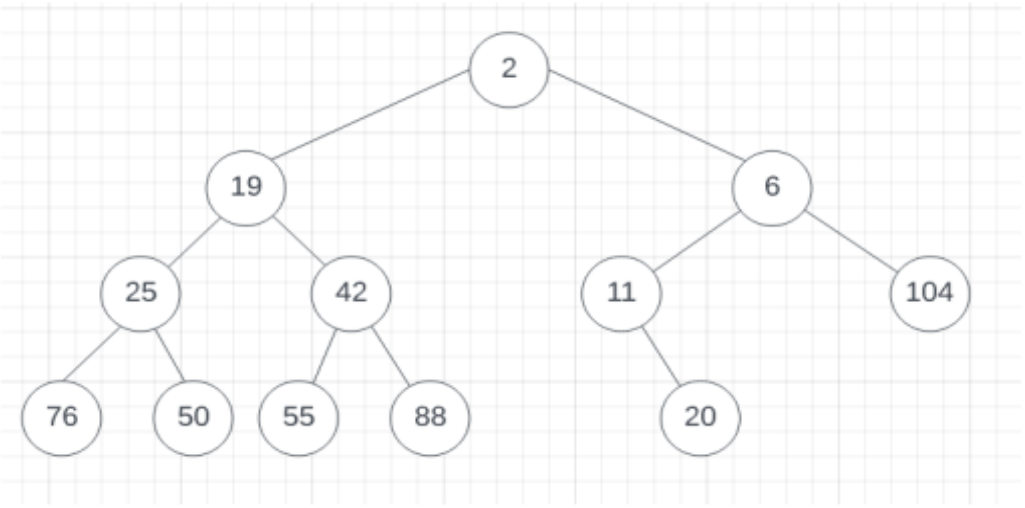


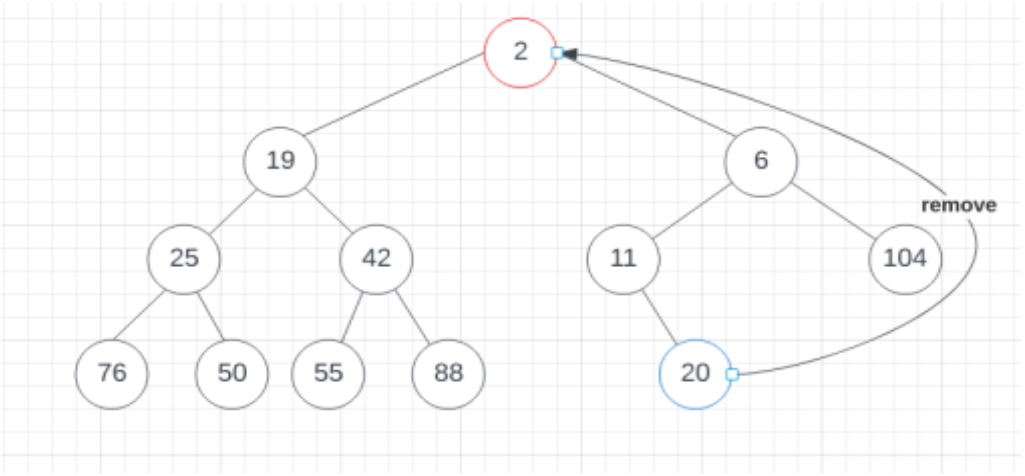
LÁMINA 37

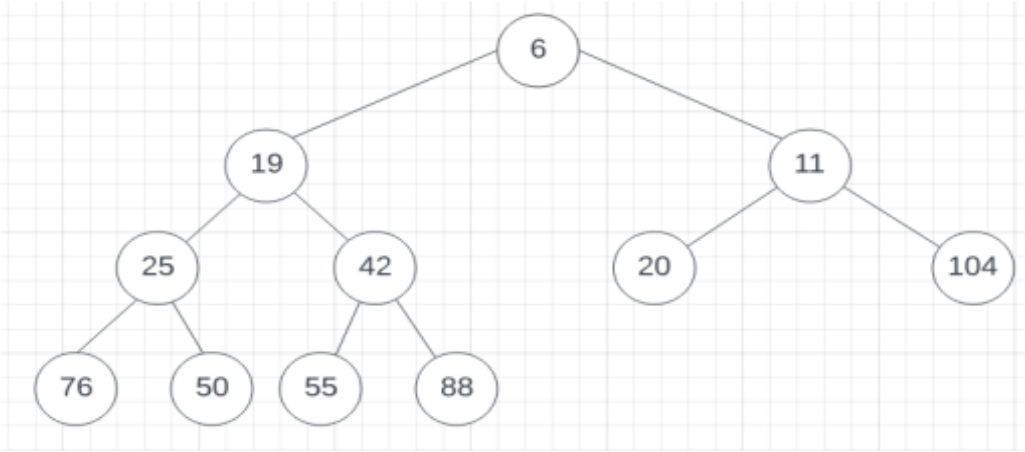
- Supongamos que tenemos el montículo mínimo que se muestra a continuación.
- Muestre el estado del árbol del montículo después de que se haya llamado ejecutado la operación remove 3 veces, y qué elementos son devueltos por la eliminación.



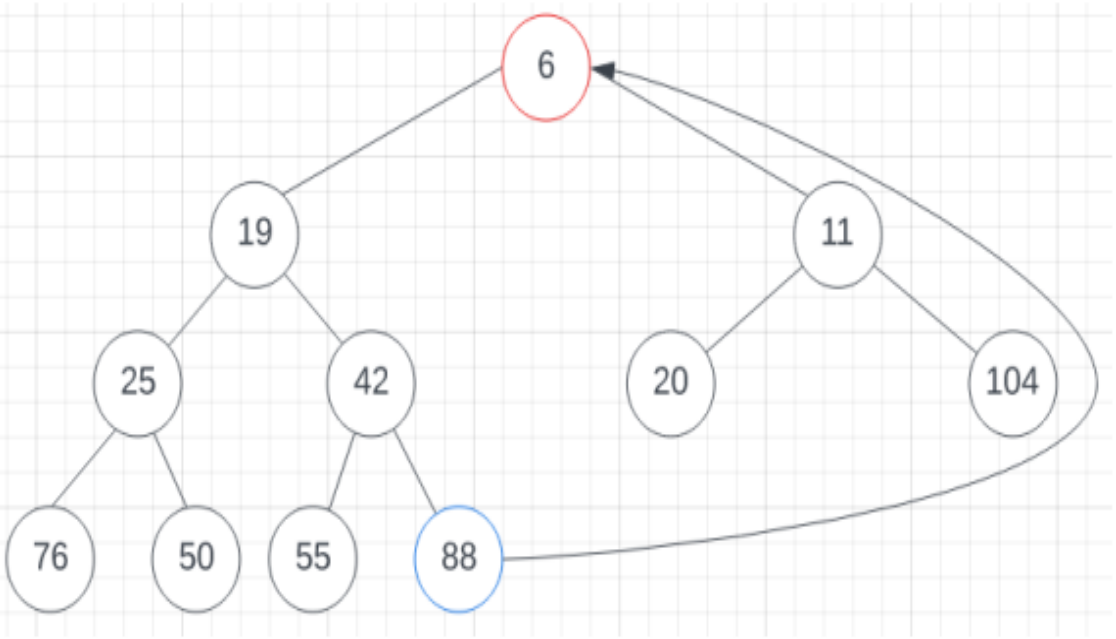

2
19
6
25
42
11
104
76
50
55
88
20

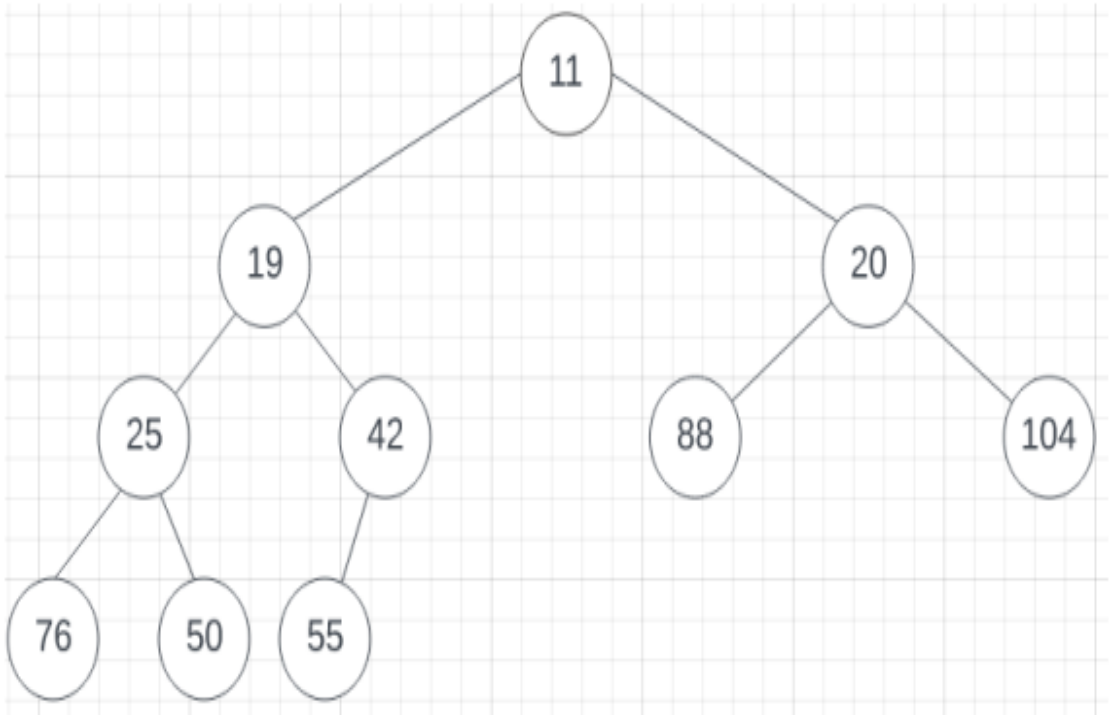
-1er remove


2
19
6
remove
25
42
11
104
76
50
55
88
20

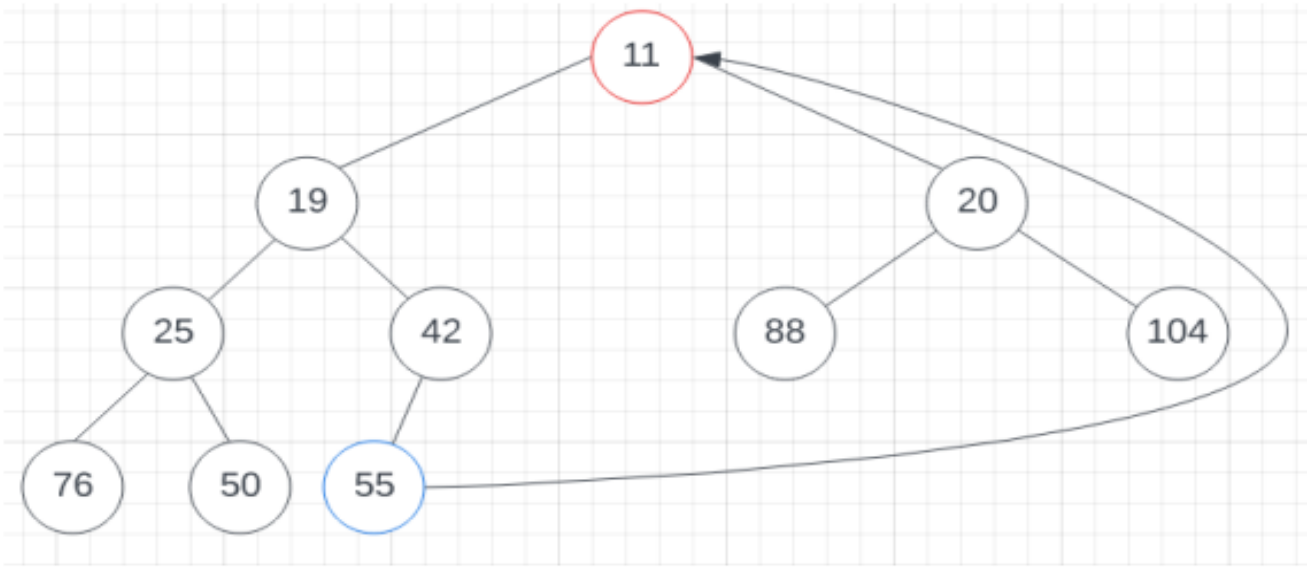

6
19
11
25
42
20
104
76
50
55
88

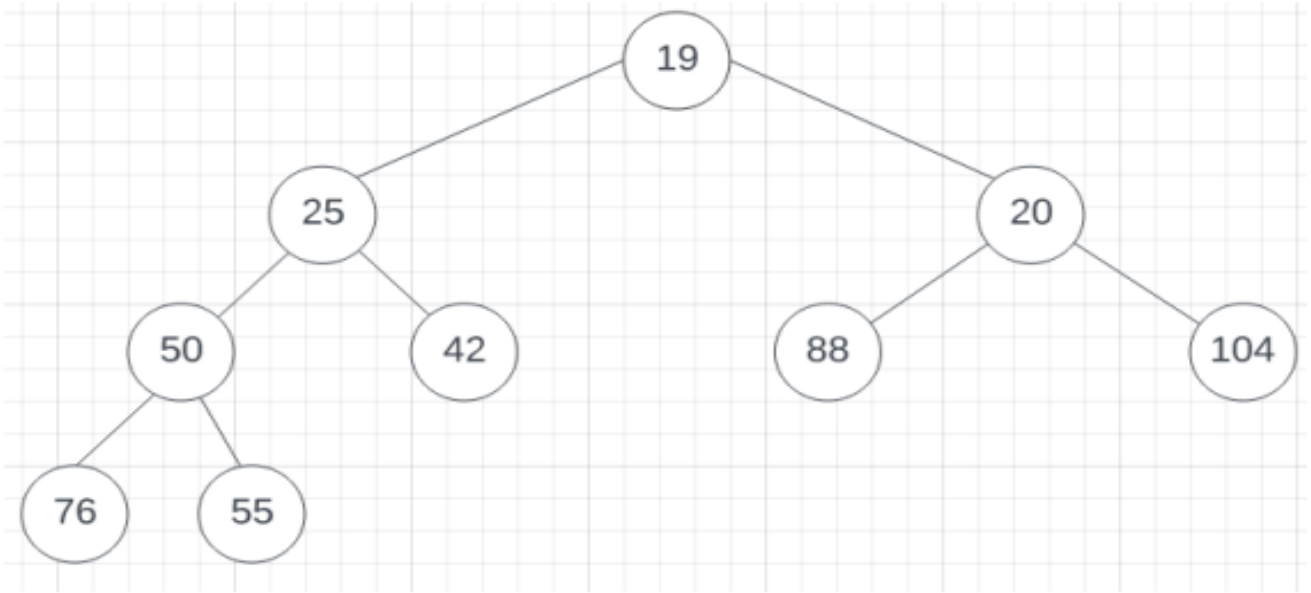
-2do remove





-3er remove





Salida luego del 3er remove:

LÁMINA 70:

- *Idea:* Trátese `a` sí mismo como un montículo máximo, cuyos datos comienzan en 0 (no en 1).
  - `a` no está realmente en orden de montículo.
  - Pero si "burbujea hacia abajo" repetidamente cada nodo que *no sea hoja* , comenzando desde el último, eventualmente tendrá un montículo adecuado.

- Ahora que `a` es un montículo máximo válido:
  - Llame a `remove` repetidamente hasta que el montículo esté vacío.
  - Pero hágalo de modo que cuando se "elimine" un elemento, se mueva al final del arregloz en lugar de desalojarlo por completo del arreglo.
  - Cuando termines, ¡voilá! El arreglo está ordenado.

Clase HeapPriorityQueue

```
public class HeapPriorityQueue {
    private static int N;

    public static void HeapPriorityQueue(int arr[]) {
        N = arr.length - 1;
        for (int i = N / 2; i >= 0; i--) {
            maximo(arr, i);
                                    }
                                                }

    public static void maximo(int arr[], int i) {

        int izquierda = 2 * i;
        int derecho = 2 * i + 1;
        int maximo = i;
        if (izquierda <= N && arr[izquierda] > arr[i]) {
            maximo = izquierda;
                                                }
        if (derecho <= N && arr[derecho] > arr[maximo]) {
            maximo = derecho;
                                                    }
        if (maximo != i) {
            cambio(arr, i, maximo);
            maximo(arr, maximo);
                    }
                                                        }

    public static void cambio(int arr[], int i, int j) {
        int tmp = arr[i];
        arr[i] = arr[j];
        arr[j] = tmp;
                                                            }
```

```
    public void heapSort(int arr[]) {
        HeapPriorityQueue(arr);
        for (int i = N; i > 0; i--) {
            cambio(arr, 0, i);
            N = N - 1;
            maximo(arr, 0);
                                }
                                    }
}
```

Metodo main

```
public class monticulo {
    public static void main(String[] args) {

        HeapPriorityQueue obj = new HeapPriorityQueue();
        int []arreglo={12,4,8,80,1};


        System.out.println("\033[32mARREGLO ORIGINAL\033[37m");
        mostrarArr(arreglo);
        obj.heapSort(arreglo);
        System.out.println("\n\033[32mARREGLO CON HEAPSORT\033[37m");
        mostrarArr(arreglo);
                                    }

    private static void mostrarArr(int []m){
        for(int i: m){
           System.out.print(i+" ");  }
            System.out.println();     }

}
```

Ejecución

```
ARREGLO ORIGINAL
12 4 8 80 1

ARREGLO CON HEAPSORT
1 4 8 12 80
BUILD SUCCESSFUL (total time: 5 seconds)
```