## Actividad 3.2 - Árbol Heap: Implementando una fila priorizada

## **Push**

```
int parent(int i){ return (i - 1) / 2; }

/*Agregue un dato a la fila priorizada*/
void priority_queue::push(int dato){

   datos.push_back(dato); // Se hace un append a la lista de datos
   int index = datos.size() - 1;

   while(index > 0 && datos[parent(index)] < datos[index]){
        // Cambiamos los elementos de posición.
        swap(datos[index], datos[parent(index)]);

        index = parent(index); // la POSICIÓN del padre pasa a ser nodo hijo. Es su posicion correcta
   }
}</pre>
```

```
Adiided 3.2
void push (int dota)
   dobs. push-bock (dots) (
     index= dotos. size()-1 (2)
   while index > o and do bs [povent (index)]
 (484-1 SUSP (datos [index), datos (parent (index)])
         index= povent(index) ( {
 T(n)= (+ (2+(3logstj + (4 log(tj-1)+(5 log(Etj-1)
T(n) = (,+(2-(y-(5+
                        109 2+
```

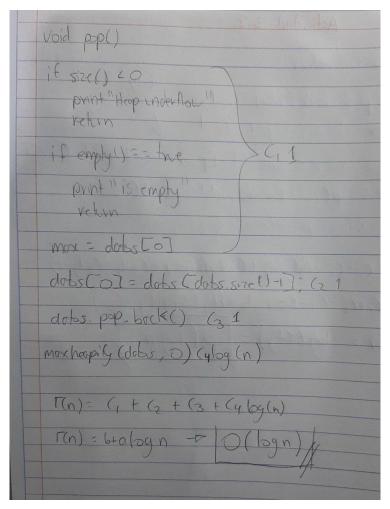
## Pop

```
void priority_queue::pop(){
    //cout << heapExtractMax(datos) << endl;
    if (size() < 0){
        cout << "Heap underflow" << endl;
        return;
    }
    if (empty() == true){
        cout << "La lista ya esta vacia" << endl;
        return;
    }

    // El más grande está siempre al inicio.
    int max = datos[0];

    // el último (menor valor) pasalo al frente para reorganizar todo el árbol y conservar propiedad.
    datos[0] = datos[datos.size() - 1];

    datos.pop_back(); // ya pasaste el más pequeño al frente. Borra el que está al final que es el mismo valor
    maxHeapify(datos, 0); // reorganizas el árbol
}</pre>
```



## Top, Empty & Size

```
/*Regresa el valor del dato que esta con mayor prioridad en la fila priorizada.*/
int priority_queue::top(){
    if (datos.size() >= 1){
        return datos[0];
    }
    cout << "No hay elementos en la lista" << endl;
    return -1;
}

/*Regresa un valor boleando diciendo si la fila priorizada esta vacía o tiene datos.*/
bool priority_queue::empty(){
    return (datos.size() == 0) ? true : false;
}

/*Regresa la cantidad de datos que tiene la fila priorizada*/
int priority_queue::size(){
    return datos.size();
}</pre>
```

int top()
if (dots. size () >=1) (, 1
refun dotos CO3 (2)
print" no hoy elements " (3)
retun-1 (g)
4
T(n)= (+ (2+(3+(4=0(1)))
Tury Criss of Line
bal empty ()
return (dotos. size == 0)? true: folse; (, )
To= G = O(1)
int Size ()
return dotas. Size () (1)
T(n) = (1 = 0(1))
The state of the s