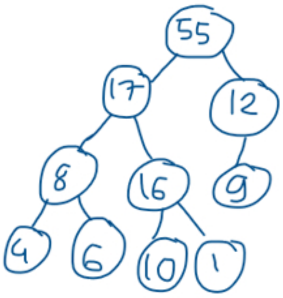
Heap:

- parent is > children   max
- parent is < chidren   min



n	level (height)	$\log_2(n+1)-1$
1	0	0
3	1	1
7	2	2
15	3	3

↓

Calculate how many level in a tree using n= elements

[55, 17, 12, 8, 16, 9, 4, 6, 10, 1]

parents (i): $\left(\frac{i-1}{2}\right)$

left (i): $2i+1$

right (i): $2i+2$

- minimum node (when there is 1 element on the lowest level)

  $2^{h}+1-1=2^{h}$

- maximum node (when the lowest level is full)

  $2^{h+1}-1$

- height = log n

prove: $2^h \le n \le 2^{h+1}-1$

$h \le \log n \le h+1$

$\Rightarrow h = \log n$

add:

insert next availabe space
heapify up

remove:

remove the root
replace with the last element
heapify down: ↓

swap with the larger of the children

```
int lastposition;

E[] array = (E[]) new Object[size];

public void add(E obj) {
        array[++lastposition] = obj;
        trickle Up (last positio);
}
```

```
public void Swap(int from, int to){
        E tmp= array[from];
        array[from]= array[to];
        array[to] = tmp;
 }

public void trickleUp (int position){
      if (position == 0)
          return;
      int parent = (int) Math.floor((position -1)/2)

      if (array[position].compareTo (array[parent]) >0){
          swap(position, parent);
          trickleUp(parent);
      }
}

public E remove(){
      E tmp= array[0]
      swap ( 0, lastposition--);
      trickleDown(0);
       return tmp;
}
```

```
public void trickleDown(int parent){
    int left= 2* parent +1;
    int right = 2* parent +2;
    if (left == last position && array[parent] < array[left]){
        swap( parent , left);
        return;
    }
    if(right== last position && array[parent]< array[right]{
        swap (parent, right);
        return;
```

Deal with the last leaf :

```
}
 if( left >= last position || right >= last position)   → out of range
          return ;
 if( array [left] > array [right] &&
        array [parent] < array [left]){
      swap (parent, left)
      trickle Down (left);                              } Deal with middle
   }
 if (array[parent] < array[right]) {
        Swap( parent, right);
        trickle down (right);
 }
```

Heap Sort: