## Explanation of the method make Heap ()

```
public static int[] makeHeap3(int[] a){
    for(int i =a.length/2-1;i>=0;i--){
public static void heapify(int[] a,int size,int root){ -
    int minimum = root; Track the min data index (Assume @root)
    int left = 2*root+1;
    if(left<size && a[left]<a[minimum]){</pre>
                                . If the minimum is not at root
                                 make it so.
        heapify(a, size, minimum); <
```

An Example

Array = [7, 4, 13, 3, 10, 5, 12] The binary tree representation



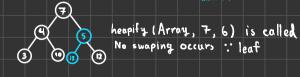
First for loop starts at index 2

Array = [7, 4, 13, 3, 10, 5, 12]



heapify (Array, 7, 2) is called

since 13 > 12 and 12 is not at the root, a swap occurs



i is decremented an heapify (Array, 7, 1) is called Array - [7, 4, 5, 3, 10, 13, 12]



> This for loop iterates through half the array starting at the half point, because we know that the last half of the array are leaf nodes

-> This heapify uses the binary tree form of heap to help sorting.

> If any child node has less value data change the minimum's index

> A reccursive call since this index no longer holds the minimum data a recursive call is made to ensure heap property is preserved

since 4>3 a Swap occurs



heapify (Array, 7, 3) is called No swapping occurs

i is decremented

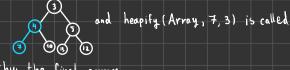
an heapify (Array, 7,0) is called



Since 7>3 Swapping occurs



heapity (Array, 7, 1) is called and swapping occur



and thus the final array

