Assignment 6

Name : Hrishikesh Rajan
Email : hrishikeshrajan3@gmail.com

Q.1)

1. arr = [1, 3, 7, 9, 12, 10, 8, 16, 18, 22, 27]
   Create a buildHeap method that returns a min heap.

   heapify(arr, n, i):
   // Write your own code

   buildHeap(arr, n):
   //Write your own code
   heapify(arr, n, i)

Ans)

CODE

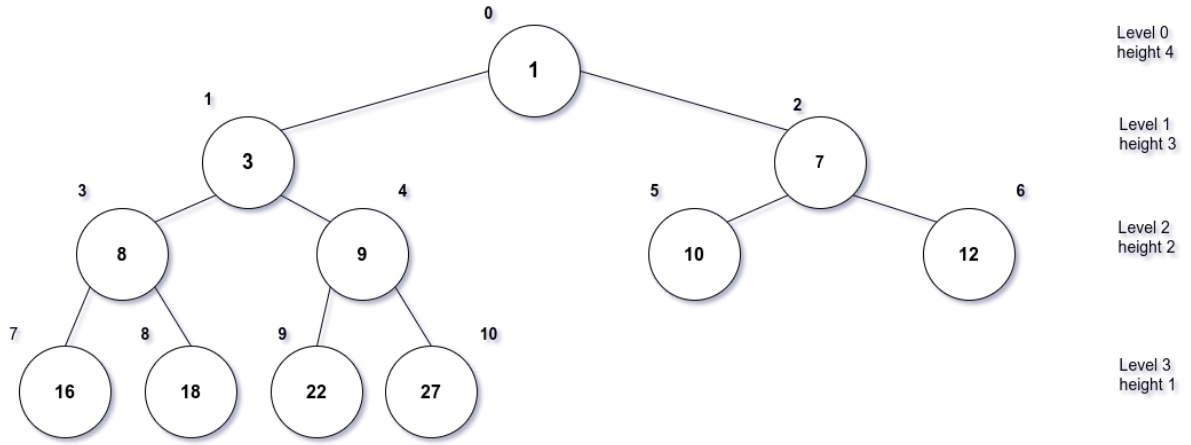
*** Javascript Implementation**

```
//Swaps with smallest child and it's parent
function swap(heap,index,smallest){
   let temp = heap[index];
   heap[index] = heap[smallest];
   heap[smallest] = temp;
   return;
}
//To calculate child indexes of left and right child
function calculateChildIndexs(i){
   const left = Math.floor((2*i + 1));
   const right = Math.floor((2*i + 2));
   return([left,right])
}
// Heapify method
function HEAPIFY_MIN_HEAP(heap,index){

   let [leftChildIndex ,rightChildIndex] = calculateChildIndexs(index);
   let smallest;
```

```
   if(leftChildIndex < heap.length && heap[leftChildIndex] <
heap[index]){
      smallest =  leftChildIndex;
   }else{
       smallest = index
   }
   if(rightChildIndex < heap.length && heap[rightChildIndex] <
heap[smallest]){
       smallest =  rightChildIndex;
   }

   if(smallest != index){
       swap(heap,index,smallest)
       HEAPIFY_MIN_HEAP(heap,smallest)
   }
   if( index < 1){
       return heap;
   }
 }


function BUILD_MIN_HEAP(arr){

   //Calculate the index of last parent
   let parent = Math.floor((arr.length/2)-1);

   //Here we assign i = parent
   //Because our heapification process starts from the last parent
   //Then decrement by 1 till the root node;

   let i = parent;
   while(i>=0){
       HEAPIFY_MIN_HEAP(arr,i)
       i--;
   }
return result;
}


const arr =  [1, 3, 7, 9, 12, 10, 8, 16, 18, 22,27];
const result =  BUILD_MIN_HEAP(arr)
console.log(result);
```

OUTPUT

```
[ 1,  3,  7,  8,  9, 10, 12, 16, 18, 22, 27 ]
```



ANALYSIS

Total No of Nodes = 11;

*Reference

Level =0;

Height = 0;

To find the height of the tree

$N = 2^{h} - 1$

$N + 1 = 2^{h}$

Taking log on both sides

$log_2(N + 1) = log_2(2^{h})$

$h * log_2 2 = log_2(N + 1)$

$h = log_2(N + 1)$

Taking the upper bound of the h

$h = log(11 + 1)$

$h = 4$

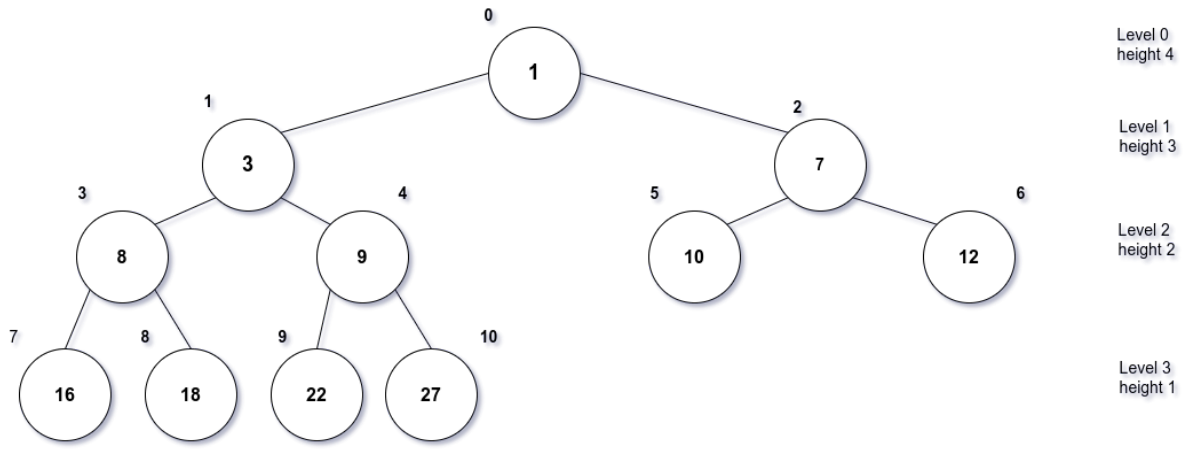
TIME AND SPACE COMPLEXITY

Build Heap

It is assumed that the input array is heap and heapify applied.

First, we need a starting index to begin heapifying
= (n/2)-1 , n = number of total nodes

Let assign this value to variable
So,in this case
= (11/2)-1
By taking the lower bound of this we get, index = 4.That means we are moving from **bottom to top**



From the diagram we can see the index 4 is the last parent node
The heapify process starts from this 4th index and continues to the root node. That means we must heapify each parent node from the last parent index.

Time Complexity of heapifying = O (log (n))

So, here we call heapify method from n/2 till root node, that mean's O(n/2) times approximately equals O(n) times

Total Time Complexity for building the MinHeap = O (n * log(n)).
Here is the catch: Even heapify takes log(n) complexity, the heapify method is not always used
Hence the exact time complexity = O (n)

In the program

> Note:
> During heapify, we calculate the first parent by n/2 and subtract by 1 to get the previous parent and continue to the root node.

Total Time Complexity for **Building MinHeap** = O (n)
Auxiliary Space = O (1) , Since we consider the input array is heap.