Assignment 6

Name : Hrishikesh Rajan
Email : hrishikeshrajan3@gmail.com

Q.3)

1. Find the **k closest points** to the origin.
   Points = [[1, 3], [-2, 2]]
   K = 1
   Output = [-2,2]

Ans)

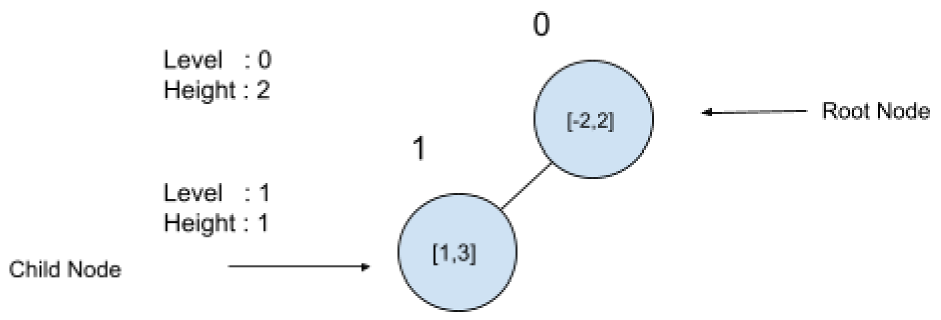
* javascript implementation

CODE:

```
function MIN_MIN_HEAP(heap){
   let temp = heap[0];
   heap[0] = heap.pop();
   return ([temp,heap]);
}

function calculatePoints(arr){
   let point_1 = arr[0];
   let point_2 = arr[1];
   return ((point_1**2) + (point_2**2))
}

//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])
}

function HEAPIFY_MIN_HEAP(heap,index){
   let [leftChildIndex ,rightChildIndex] = calculateChildIndexs(index);
   let smallest;


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

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

   //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--;
   }
   //Here we delete k times from the head node
   let m = 0;
   while(m<k){
     let  [head,heap] = MIN_MIN_HEAP(arr);
     HEAPIFY_MIN_HEAP(heap,0)
     kthCoordinates.push(head)
       m++;
   }
return kthCoordinates;
}

const arr = [[1,3],[-2,2]]
const result =  BUILD_MIN_HEAP(arr,1)
console.log(result);
```

OUTPUT :

```
[[-2,2]]
```



ANALYSIS:

Total No of Nodes = 2;

*Reference
Level =0;
Height = 1;

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

$h = log_2(2 + 1)$

$h = 2$

Height of the tree = 2

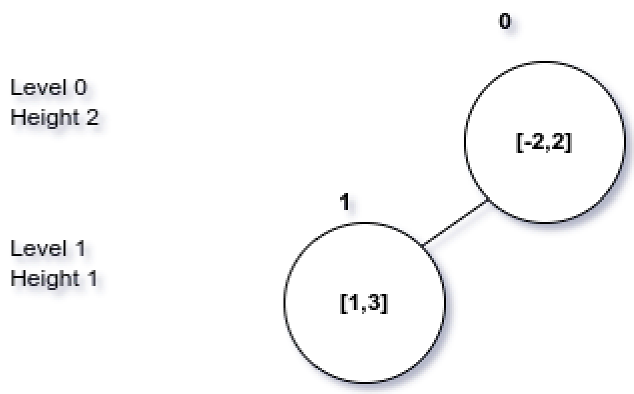
EXPLANATION:

Build Heap

It is assumed that the input array is heap and we need to only do the heapify process.

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
= (2/2)-1
By taking the lower bound of this we get, index = 0.That means we are moving from **bottom to top,** Both in this both model is fine because of small numbers of input



From the diagram we can see the index of the parent node is 0.The heapify process starts from the 0th index itself . That means only need to heapify with one child node.But the given input from the question belongs to the best case category. But we always consider the **worst case.**

Time Complexity of heapifying = O (log_2 (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_2(n)).
But theoretically proves that even heapify takes log(n) complexity, the exact total time complexity will be O (n)

Total Time Complexity for Building MinHeap = 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.But here the code flow is almost constant

Deletion

The deletion process is used to get the values out of the heap whether is min-heap or max-heap.The deletion process undertaken by several steps which are given below.

1) Copy the first node to temporary variable
2) Copy the last node value to root node and delete the last node
   2.1) At this stage the heap tree can be violate the property,In here it's min-heap property
3) Apply heapify procedure to maintain the min heap property

Here we need to consider how many values we should take from the min heap.This will be provided in the question itself,here it is 1 which is represented by " K ". So that means we need to take the value " K " times from the min-heap head. Also, for each deletion we need to call the heapify method as well.

Hence,

The time complexity of Deletion = K * O(log (n))

Auxiliary Space = O (n), New array is created for storing the closest points