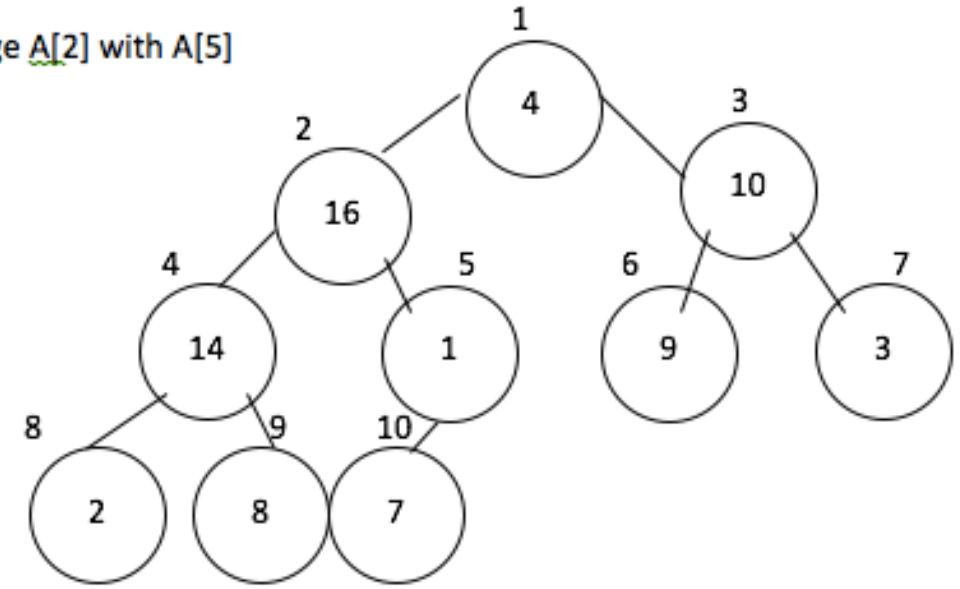
1.) A.) In a heap data structure is in the shape of a complete binary tree which has all levels filled except maybe the last level filled. At the top, also known as the root, is the priority element which can be seen as the highest or lowers. It is seen as not partially ordered and when complete has a logn height.

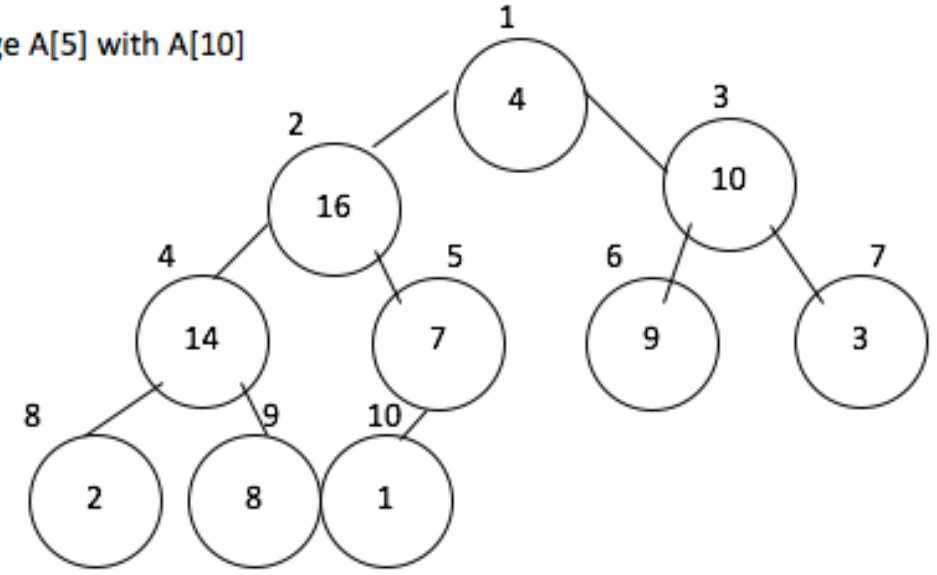
Looking at HEAPSORT algorithm it starts with constructing the heap which runs linear time, or n Θ (n), and then goes into the for loop and performs n-1 times. Max heapify would also be in the for loop. From the book, max-heapify runs Θ (lgn) time. Therfore we get n+n(n-1)logn+O(1). Simplifying to n+nlogn-logn+O(1). Nlogn would be the dominate term therefore it would be Θ(nlogn).

B.)

Exchange A[2] with A[5]

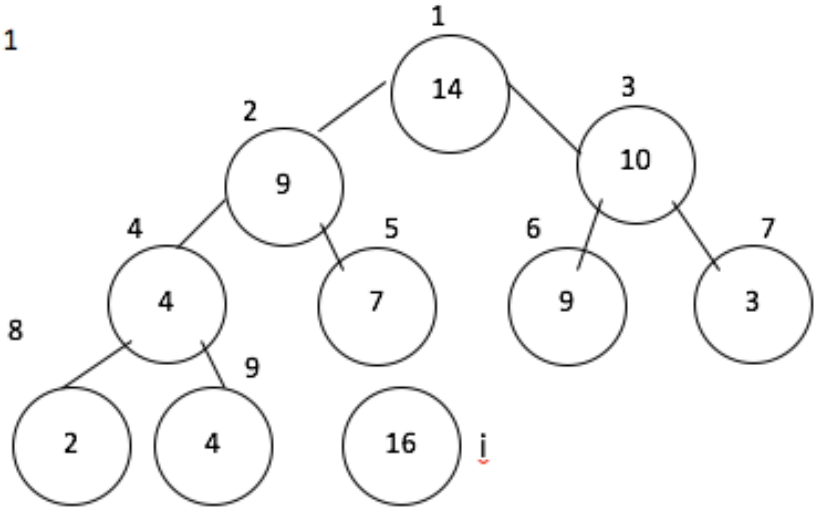


Exchange A[5] with A[10]

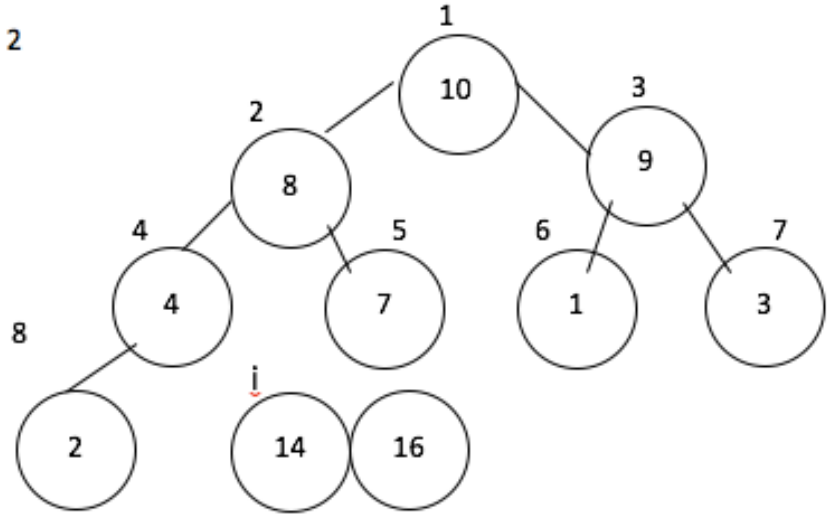


C.)

Iteration 1



Iteration 2



2)A.)

```
HEAVIER(A,P,R)
1 size=(p+r)/3
2 groupOne=A[1….size]
3 groupTwo=A[size+1….2(size)]
4 groupThree=A[2(size)…..r]
5 if(weight(groupOne)>weight(groupTwo))   //balance one
6         one=A[1]
7         two=A[2]
8         three=A[3]
9 else if(weight(groupOne)<weight(groupTwo)
10        one=A[4]
11        two=A[5]
12        three=A[6]
13 else
14        one=A[7]
15        two=A[8]
16        three=A[9]
17 if (weight(one)>weight(two))        //balance two
18        heaviest=one
19 else if (weight(one)<weight(two))
20        heaviest=two
21 else
22        heaviest=three
```

O(nlogn)

B) Asumming that n is a power of 3 and dividing it into 3 parts would run $O(n^3)$ time