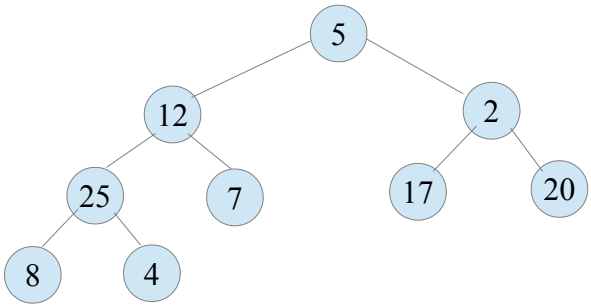
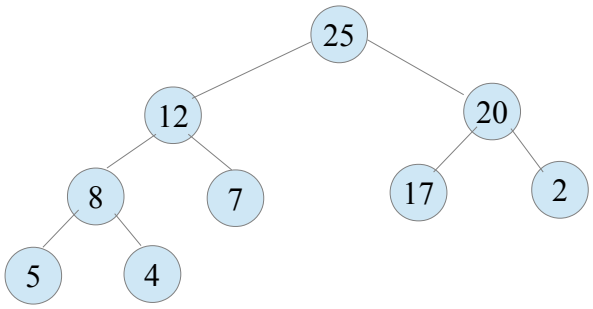
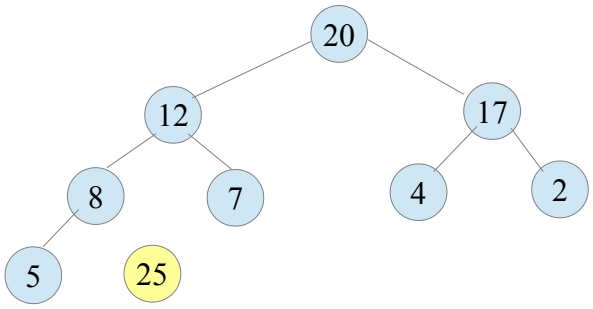
Homework 2's answer

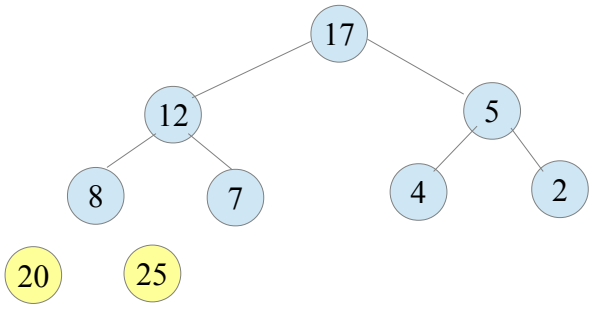
1.

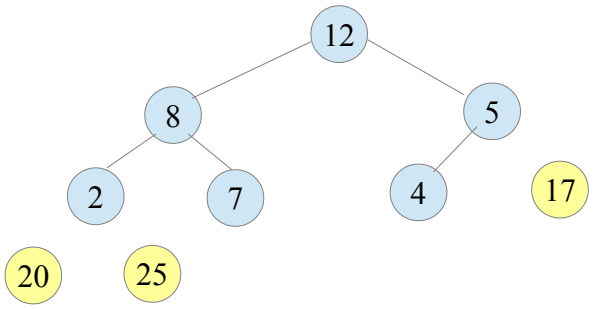
Draw a heap for A={5, 12, 2, 25, 7, 17, 20, 8, 4}.

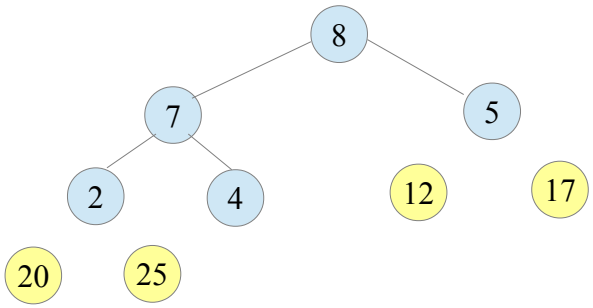


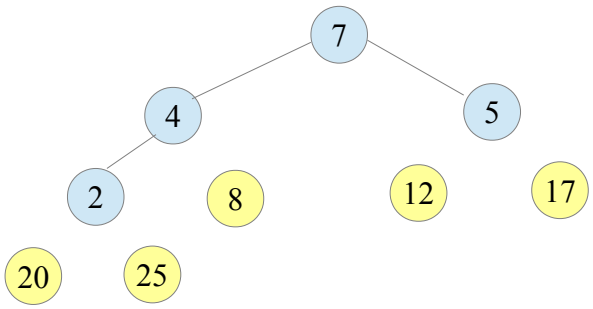


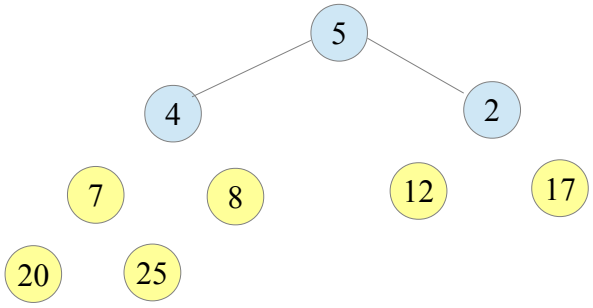


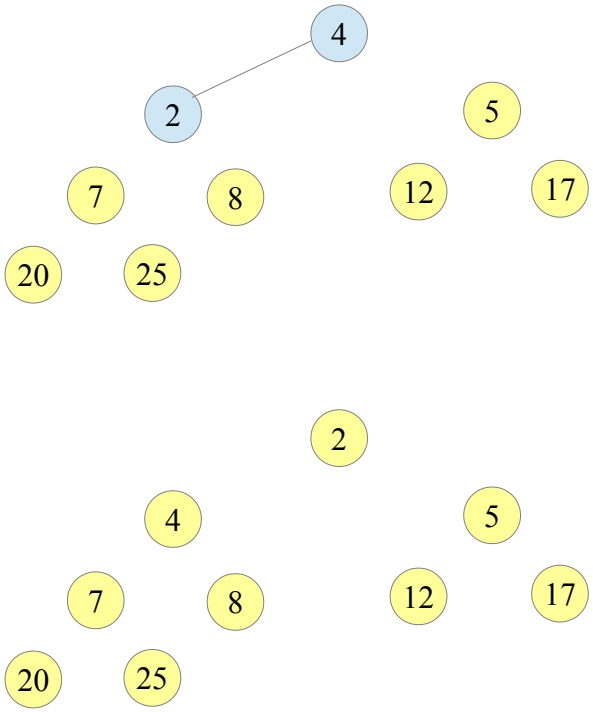



12
8
5
2
7
4
17
20
25


8
7
5
2
4
12
17
20
25


7
4
5
2
8
12
17
20
25


5
4
2
7
8
12
17
20
25



The result is {2, 5, 6, 7, 8, 12, 17, 20, 25}.

2.1

Divide-And-Conquer-Max (list s, n)

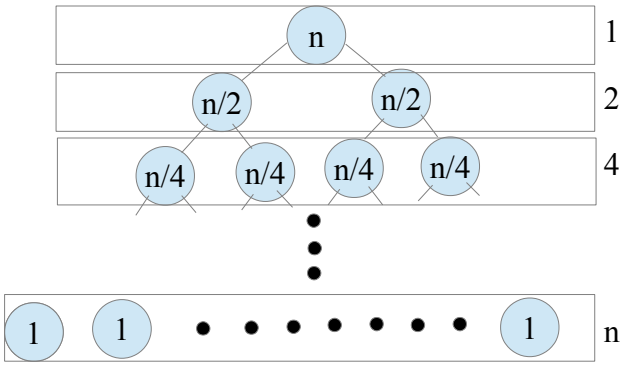
```
if(n == 1) {
        return s[0];
}
int half = n/2;
list s1 = first half members;
list s2 = last half members
int ans1 = Divide-And-Conquer-Max(s1, half);
int ans2 = Divide-And-Conquer-Max(s2, half);
if ans1 > ans2
        return ans1
return ans2
```

Divide-And-Conquer-Min (list s, n)

```
if(n == 1) {
        return s[0]
}
int half = n/2;
list s1 = first half members;
list s2 = last half members
int ans1 = Divide-And-Conquer-Min(s1, half);
int ans2 = Divide-And-Conquer-Min(s2, half);
if ans1 > ans2
        return ans2
return ans1
```

2.2

The algorithm works like a binary tree until the size of set is 1 and each node has the constant running time (O(1)) as below



Therefore, the running time is (1+2+4+8+16+....+n) = $1+2+4+8+...+2^k$

= $2^{(k+1)} - 1$

= $2*2^k - 1$

= $2n - 1$

= $O(n)$