CS2500 Homework 4

Evan Wilcox

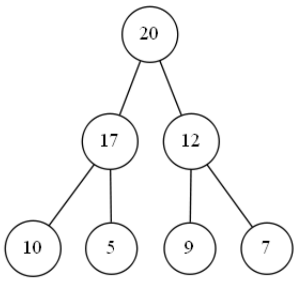
Due March 12, 2019

1. **6.1-6**

Index	1	2	3	4	5	6	7	8	9	10
Value	23	17	14	6	13	10	1	5	7	12

No this array is not a max heap because the right child at the 9th index of the node at the 4th index has a value greater than the 4th node.

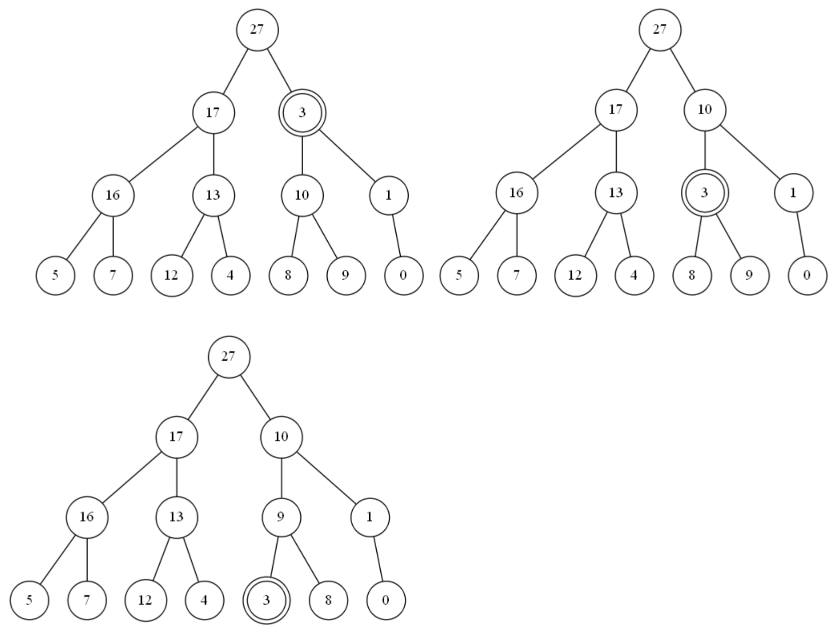
2. **6.1-7**



Index	1	2	3	4	5	6	7
Node	20	17	12	10	5	9	7

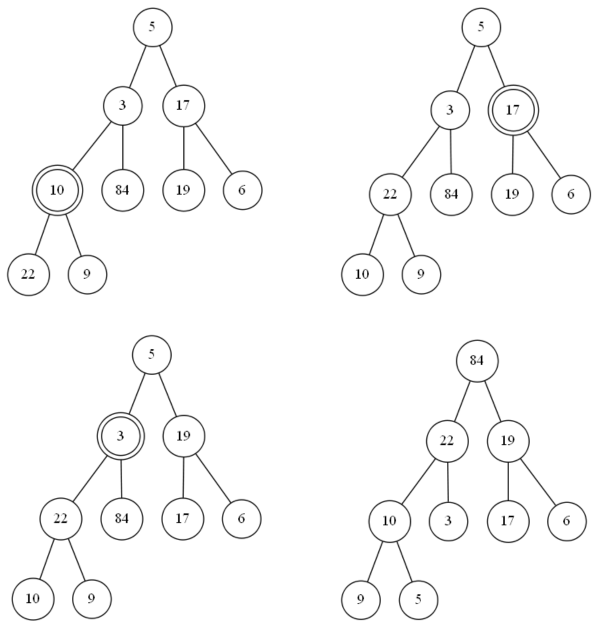
This is a 7 element tree and there are 4 leafs.
The first leaf is at index $\lfloor n/2 \rfloor + 1 = 4$
The second leaf is at index $\lfloor n/2 \rfloor + 2 = 5$
The third leaf is at index $\lfloor n/2 \rfloor + 3 = 6$
The fourth leaf is at index $\lfloor n/2 \rfloor + 4 = 7$

3. **6.2-1**

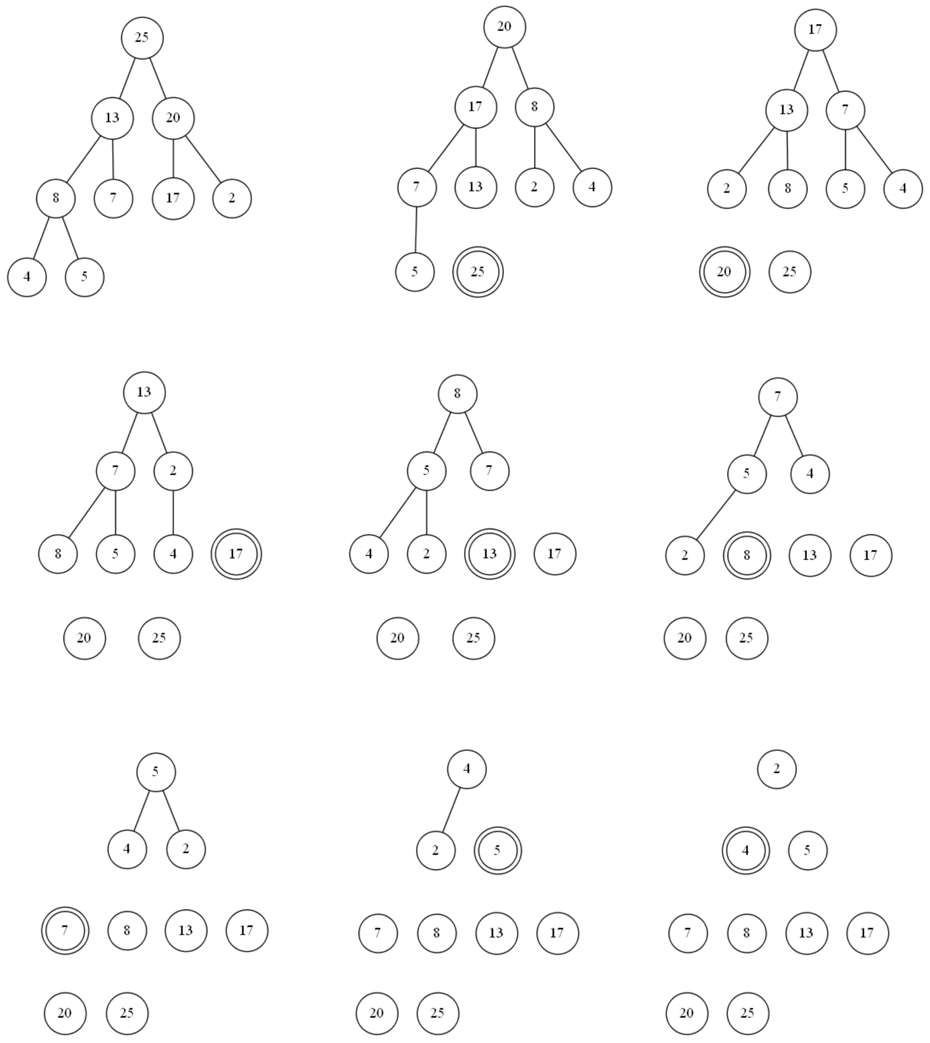


4. **6.2-3**
Calling MAX-HEAPIFY(A, i) when the element $A[i]$ is larger than its children has no effect because that node is already a max heap.

5. **6.3-1**



6. **6.4-1**



A	2	4	5	7	8	13	17	20	25

7. **6.4-2** – show this invariant implemented as a c assert statement

```
for(int j = 1; j <= i; j++)
{
    if(2 * j < heap-size)
    {
        # left child
        assert(A[j] >= A[2 * j]);
    }

    if(2 * j + 1 < heap-size)
    {
        # right child
        assert(A[j] >= A[2 * j + 1]);
    }

    for(int k = j+1;k < A.length; k++)
    {
        # elements at end of array
        assert(A[j] < A[k])
    }
}
```

8. **6.4-3**
   The running time of HEAPSORT on an array A of length n that is already sorted in increasing order is $n\lg n$ because the array must still be converted to a max heap. The running time on an array that is sorted in decreasing order is already a max heap but MAX-HEAPIFY still cost $n\lg n$ so the runtime is $n\lg n$.