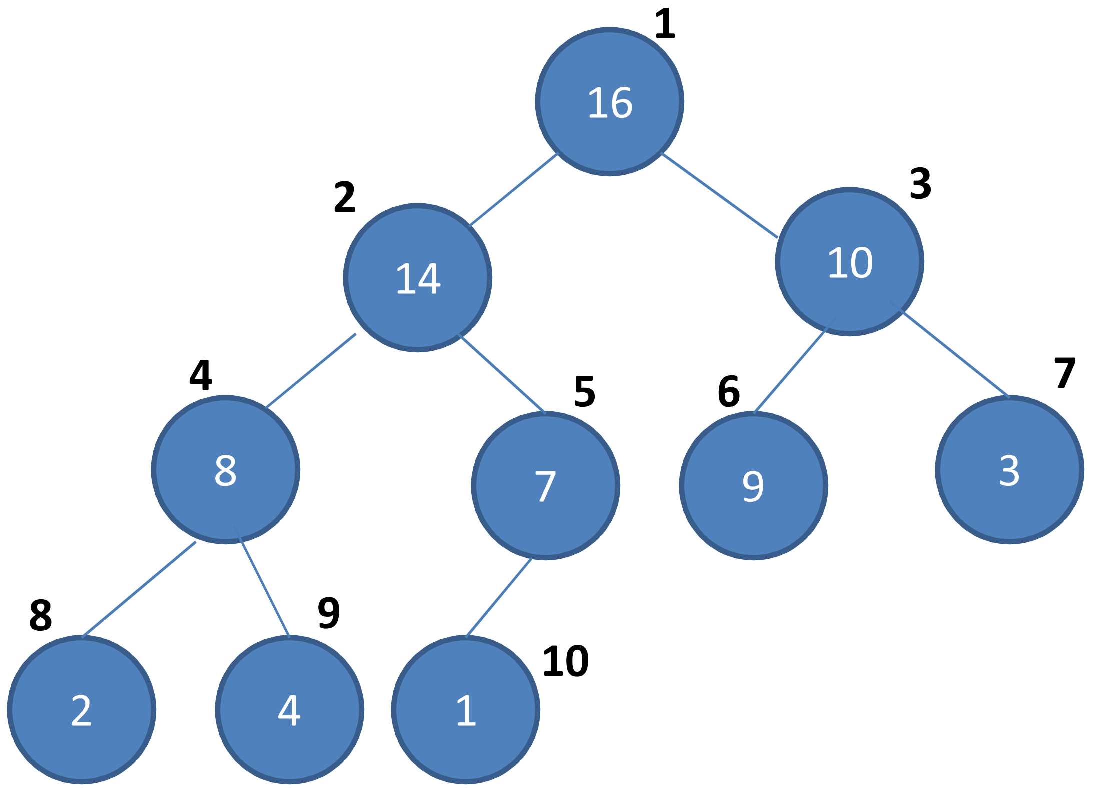
Max-Heap



index	1	2	3	4	5	6	7	8	9	10
value	16	14	10	8	7	9	3	2	4	1

Heap Properties

- An array A that represents **heap** is an object with two attributes:
  - A.length (No. of elements in the array)
  - A.heap-size (how many elements in the heap stored)
- Compute indices:
  - Parent (i) { return (i/2)}
  - Left (i) {return (2i)}
  - Right (i) {return (2i+1)}

Heap Types

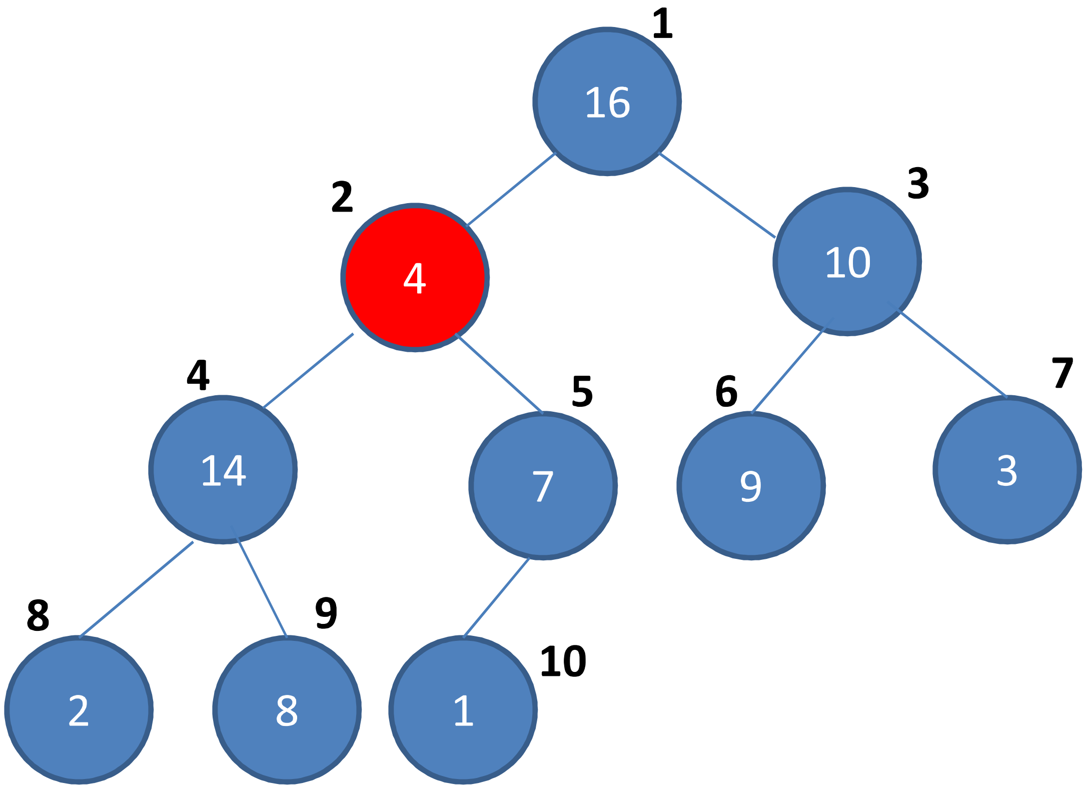
- There are two types of binary heap:
  - Max-heaps { A [Parent(i)] >= A[i] }
    - Largest element at the root
  - Min-heaps { A [Parent(i)] <= A[i] }
    - Smallest element at the root

Maintaing Heap Property

- MAX_HEAPIFY procedure

```
Max-Heapify(A,i)
        l=left (i)
        r=right (i)
        if l <= A.heap-size and A[l]>A[i]
              largest = l
        else  largest = r
        if r<= A.heap-size and A[r]>A[largest]
              largest = r
        if largest <>  I
              exchange A [i] with A[largest]
        Max-Heapify (A, largest)
```

Maintaining Heap Property - Example



Max-Heapify (A, 2)

A.heap-size = 10
l = 4
r= 5
4 <= 10 and 14 > 4
largest = 4
5 <= 10 and 7 > 14
Nothing
4 <> 2
exchange 4 with 14

Max-heapify (A, 4)

Building a Heap

- BUILD_MAX_HEAP procedure (goes half of the nodes and runs Max-Heapify on each one)

```
Build-Max-Heap(A)
        A.length = A.heap-size
        for i = [A.length / 2] down to 1
            Max-Heapify (A,i)
```

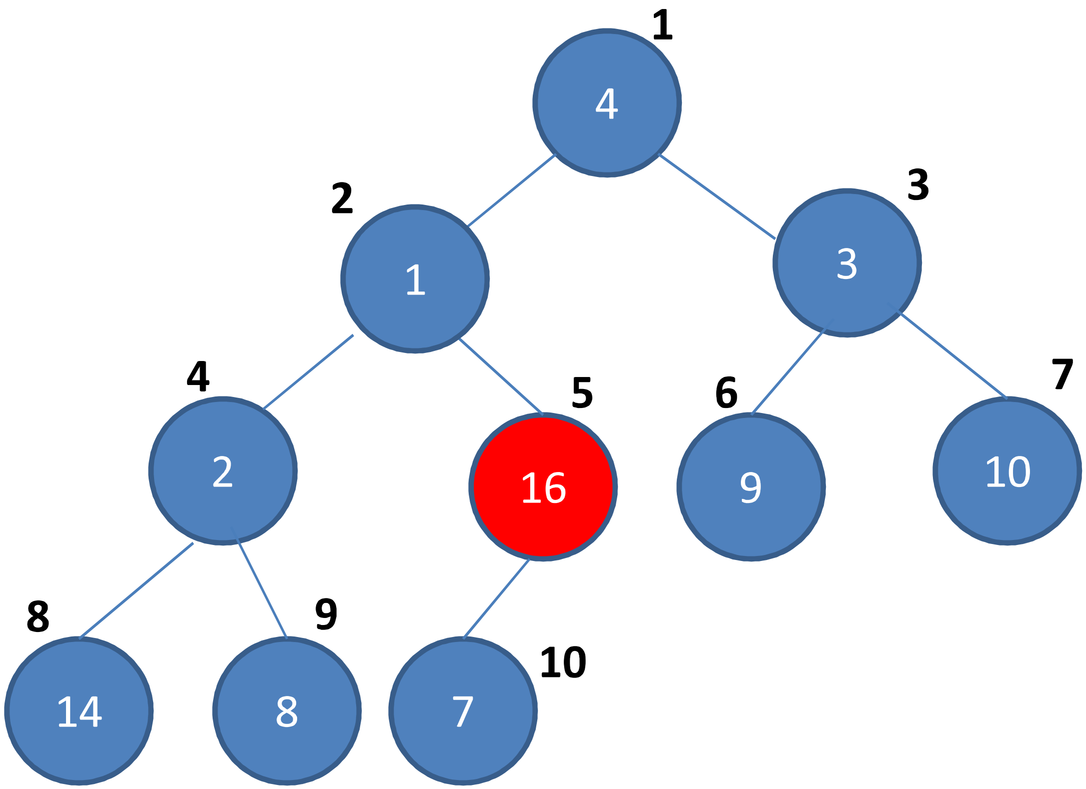
****

Note:

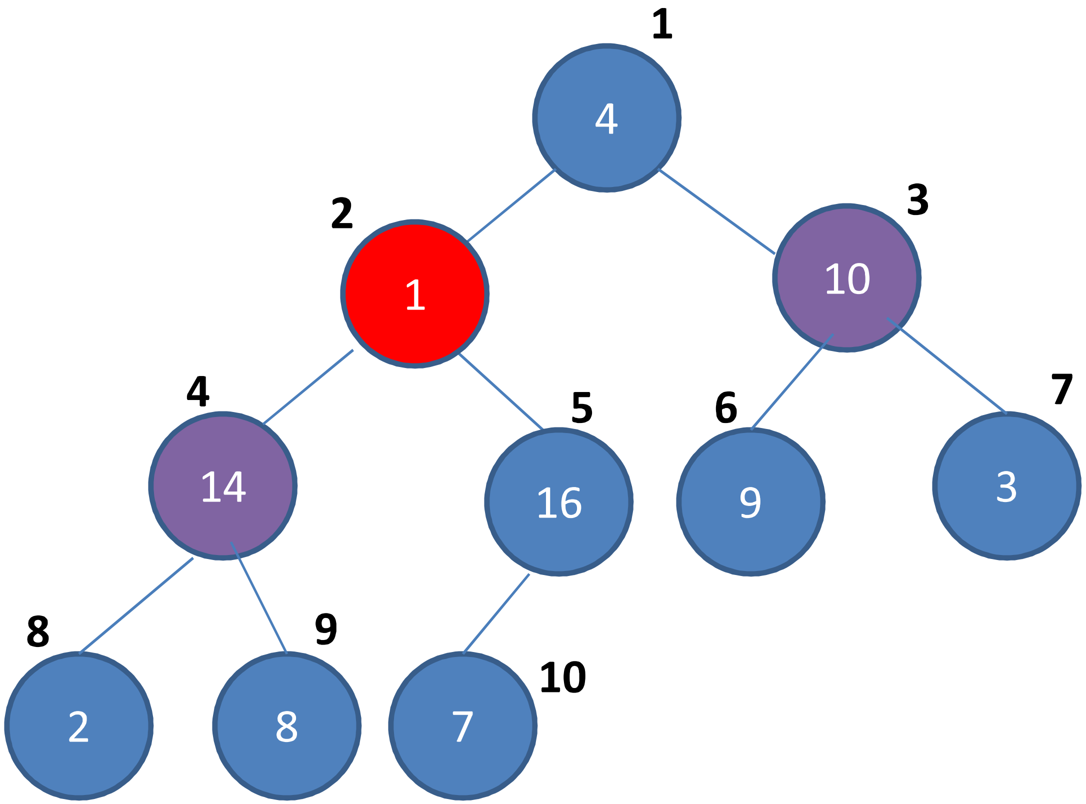
In binary tree we know that **A.length/2** elements from starting are always internal nodes, not leave nodes, so we start our procedure from A.length/2

Build Heap - Example

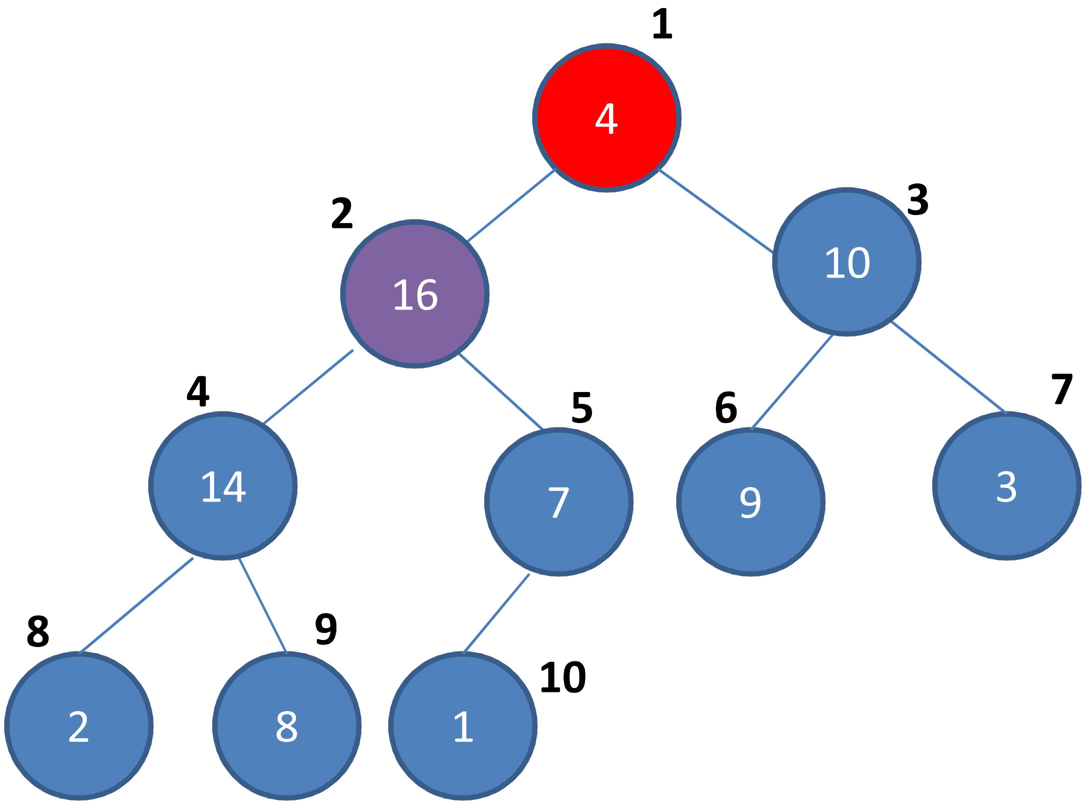
index	1	2	3	4	5	6	7	8	9	10
value	4	1	3	2	16	9	10	14	8	7



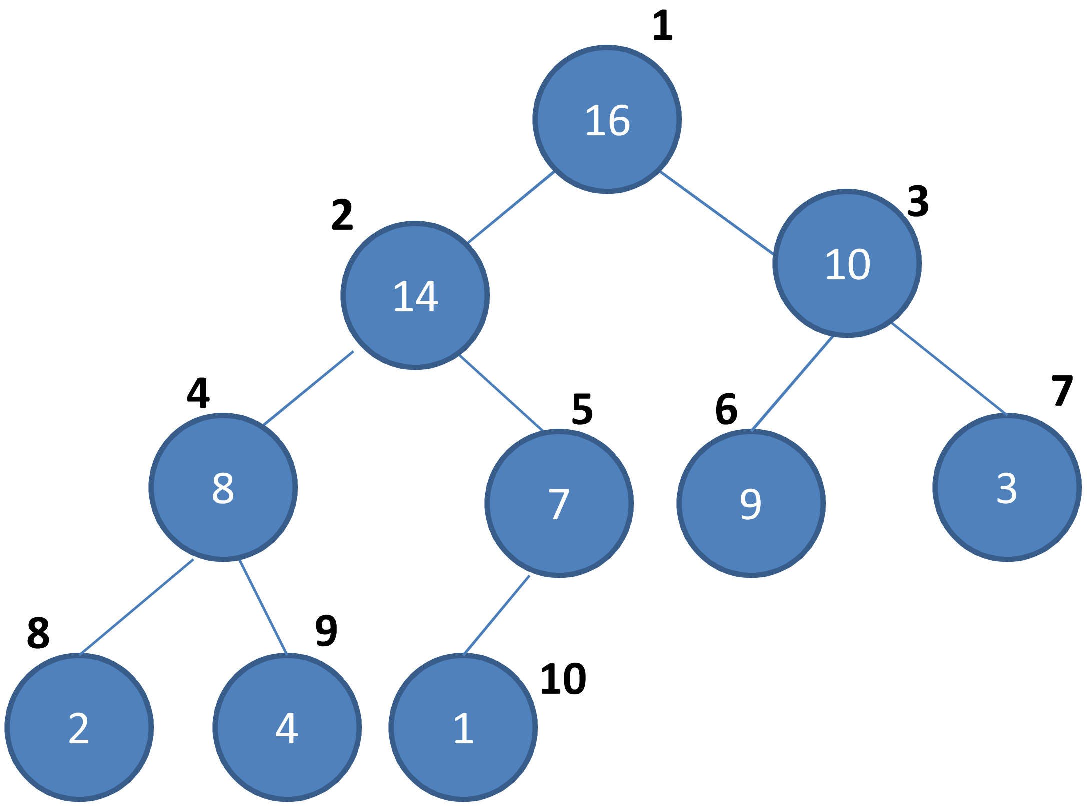
Build Heap – Example cont.



Build Heap – Example cont.



Build Heap – Example cont.



index	1	2	3	4	5	6	7	8	9	10
value	16	14	10	8	7	9	3	2	4	1

Heap Sort Algorithm

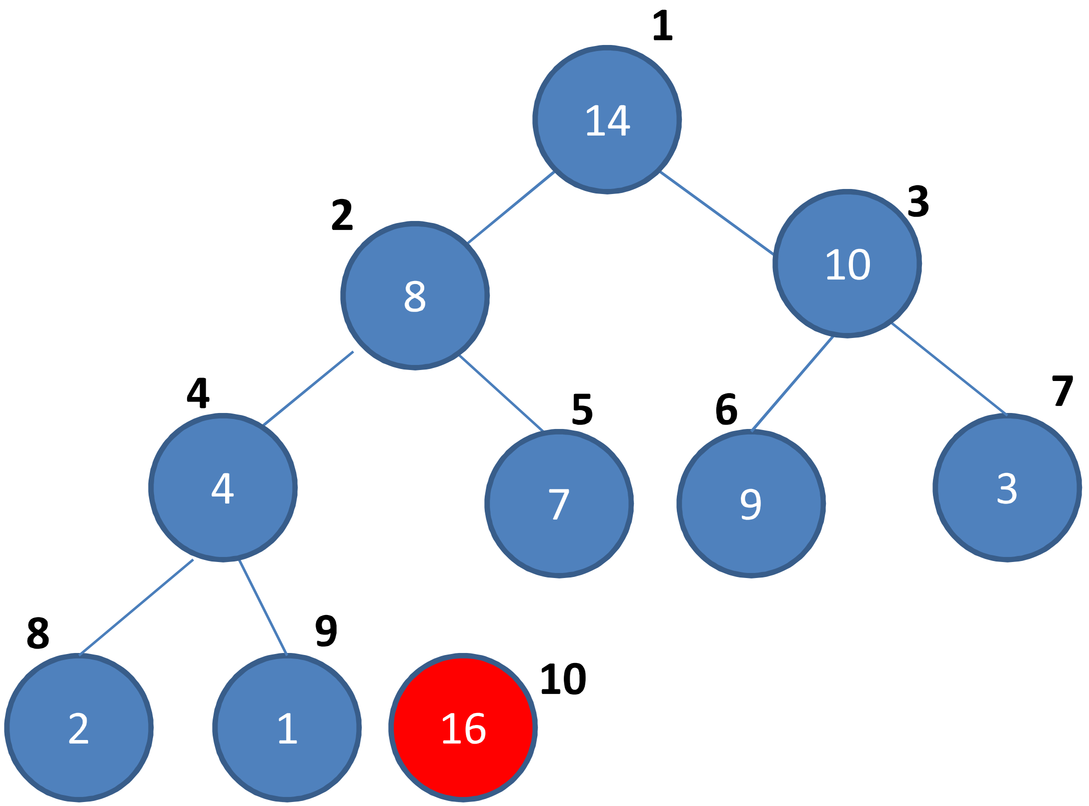
- HEAPSORT procedure

```
HeapSort(A)
        Build-Max-Heap(A)
        for i = A.length down to 2
                exchange A[1] with A[i]
                A.heap-size = A.heap-size  - 1
                Max-Heapify (A,1)
```

****

HEAPSORT procedure takes **O (n log n)** running time.
Build-Max-Heap procedure takes **O (n)** running time.
Max-Heapify procedure takes **O (log n)** running time.

Heap Sort Algorithm - Example



Heap Sort Algorithm - Example

index	1	2	3	4	5	6	7	8	9	10
value	16	14	10	8	7	9	3	2	4	1

index	1	2	3	4	5	6	7	8	9	10
value	1	2	3	4	7	8	9	10	14	16