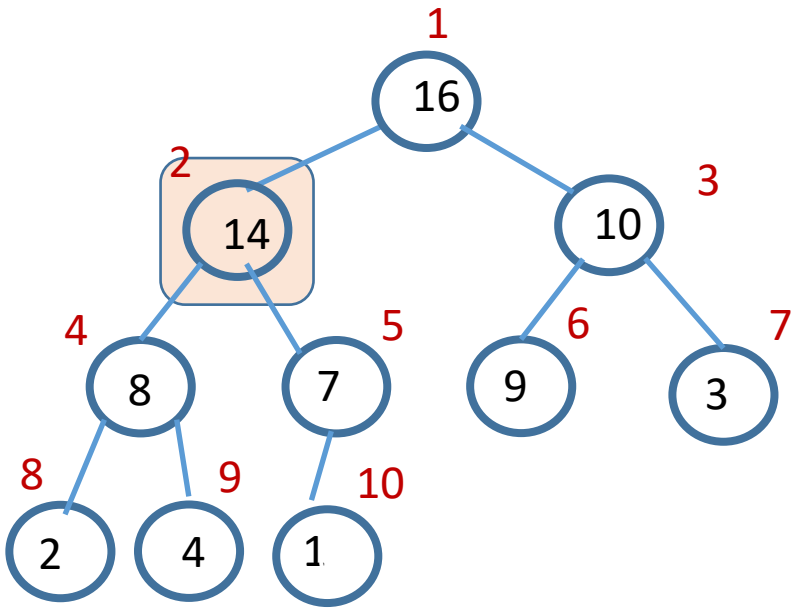
COMP 2611, DATA STRUCTURES
LECTURE 14

HEAPS: MAX-HEAPS

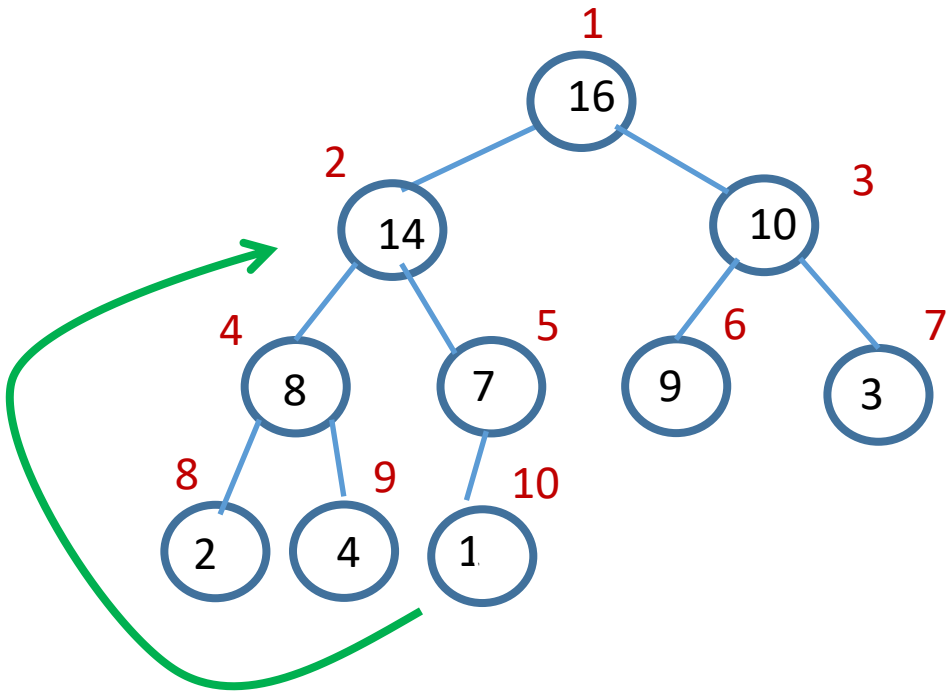
- **Review of deleting an element from a max-heap**
- **Heap sort**
- **Inserting an element in a max-heap**
- **Conclusion of heaps**

Deleting the Value at Index *i* in a Max-Heap



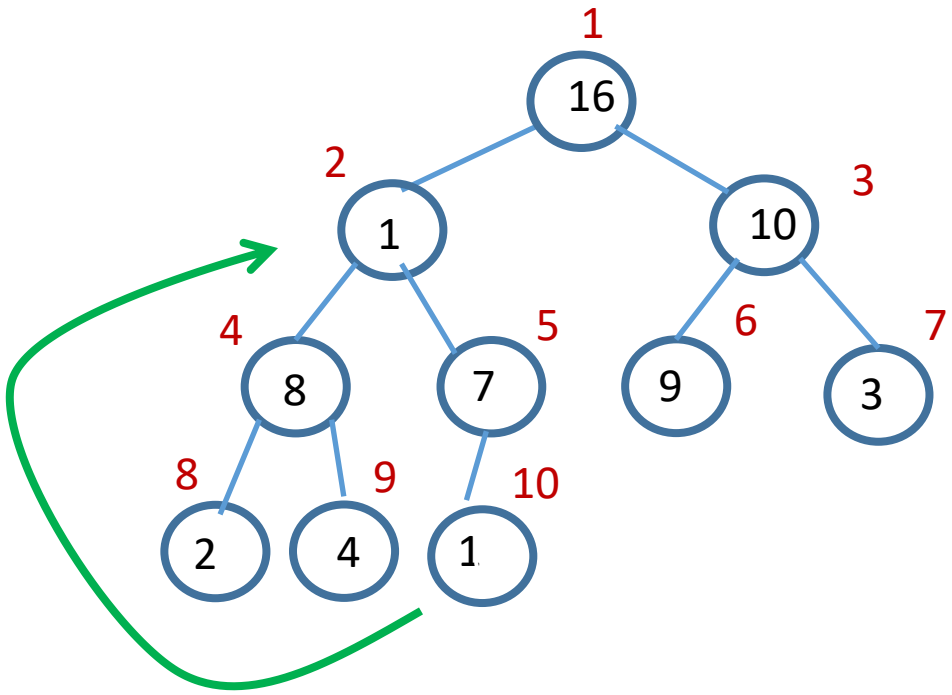
0	1	2	3	4	5	6	7	8	9	10
	16	14	10	8	7	9	3	2	4	1

Deleting the Value at Index *i* in a Max-Heap



0	1	2	3	4	5	6	7	8	9	10
	16	**14**	10	8	7	9	3	2	4	**1**

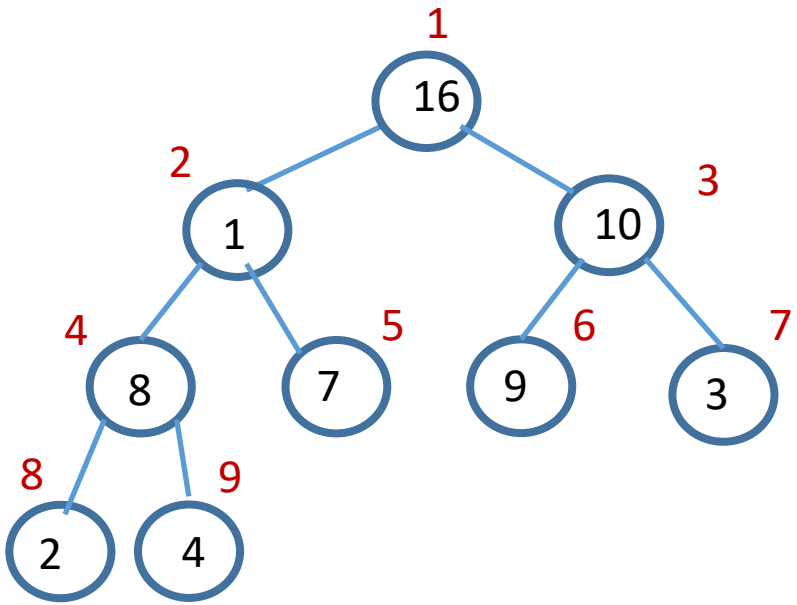
Deleting the Value at Index *i* in a Max-Heap



0	1	2	3	4	5	6	7	8	9	10
	16	**1**	10	8	7	9	3	2	4	**1**

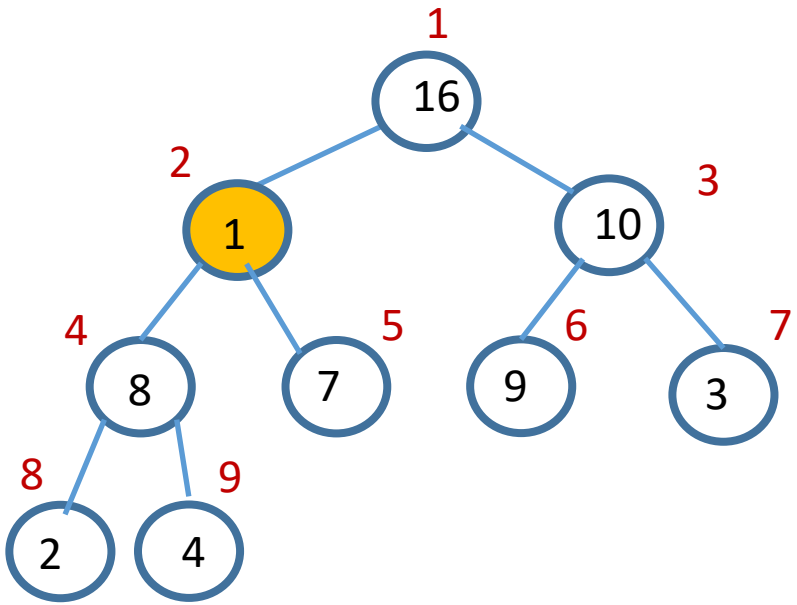
Deleting the Value at Index *i* in a Max-Heap

Now, call maxHeapify at location 2.



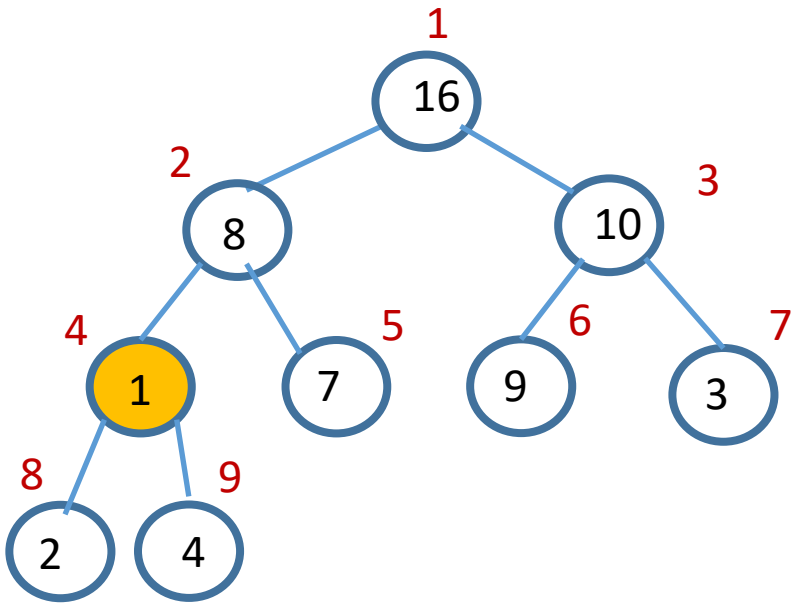
0	1	2	3	4	5	6	7	8	9
	16	1	10	8	7	9	3	2	4

Deleting the Value at Index *i* in a Max-Heap

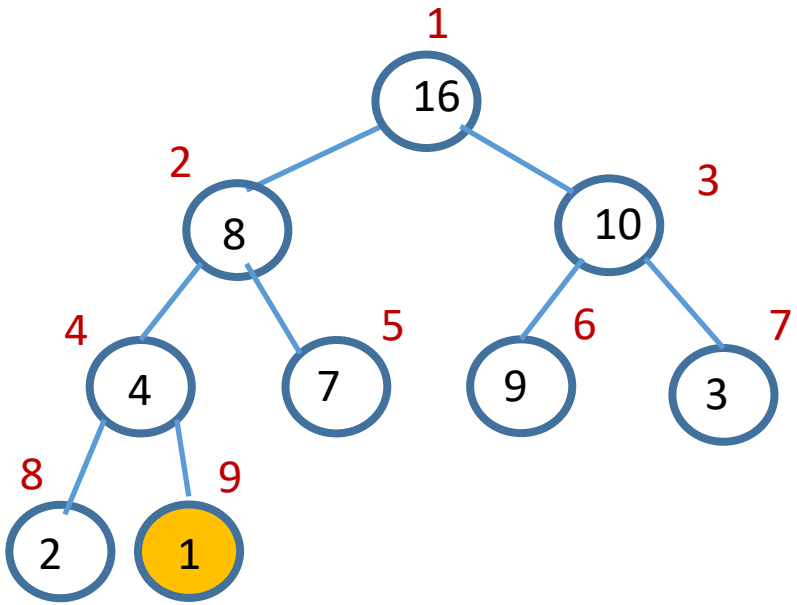


0	1	2	3	4	5	6	7	8	9
	16	1	10	8	7	9	3	2	4

Deleting the Value at Index *i* in a Max-Heap

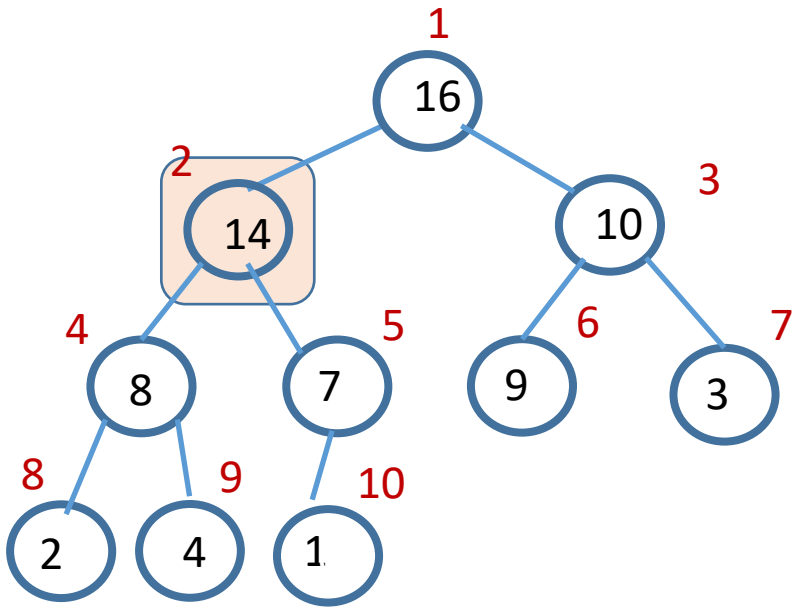


Deleting the Value at Index *i* in a Max-Heap

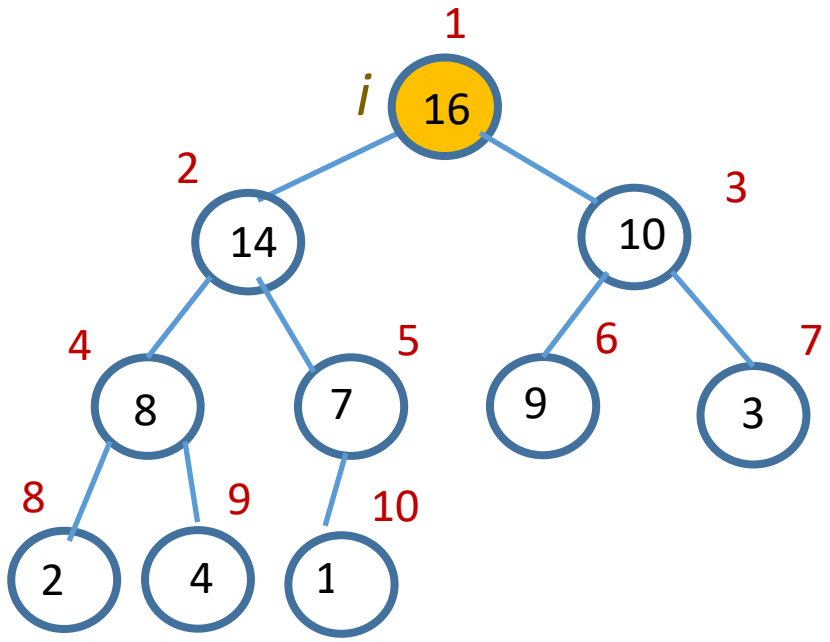


Deleting the Value at Index *i* in a Max-Heap

```
int deleteMaxHeap (MaxHeap * heap, int i) {

   int toDelete = heap->A[i];

   heap->A[i] = heap->A[heap->size];
   heap->size = heap->size - 1;
   maxHeapify (heap, i);

   return toDelete;

}
```



Deleting the Value at Index *1* in Max-Heap

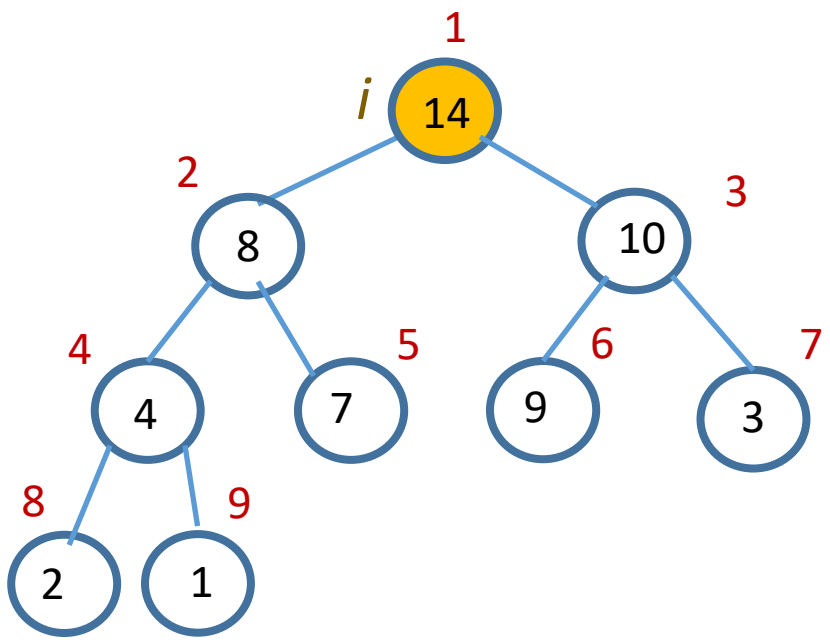


0	1	2	3	4	5	6	7	8	9	10
	16	14	10	8	7	9	3	2	4	1

A

Deleting the Value at Index *1* in Max-Heap

16

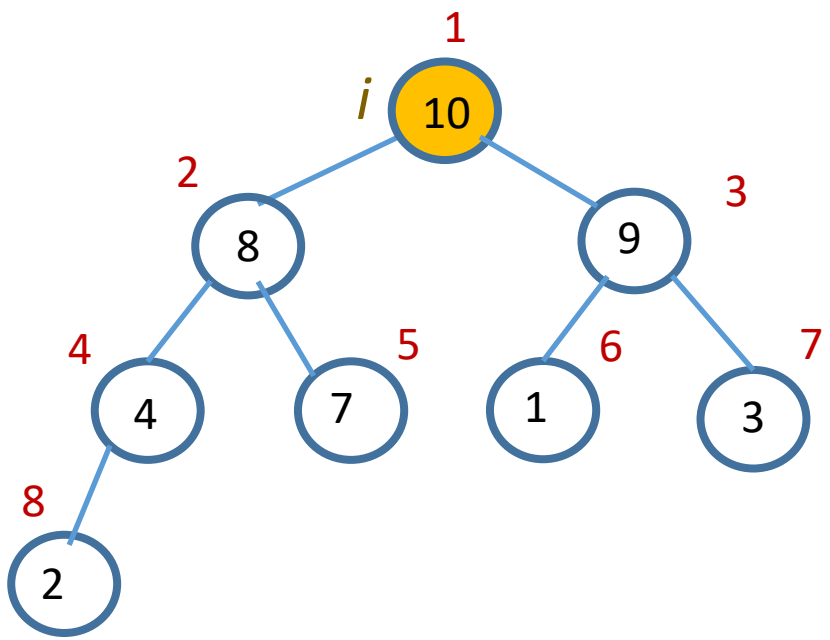


0	1	2	3	4	5	6	7	8	9	10
	14	8	10	4	7	9	3	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

16
14

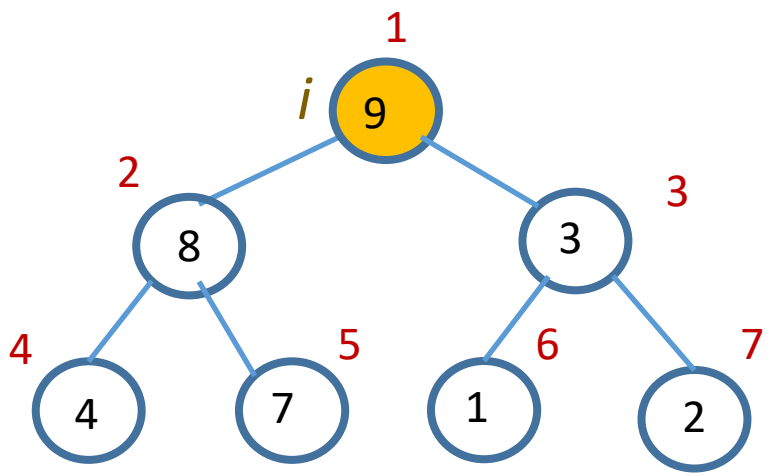


0	1	2	3	4	5	6	7	8	9	10
	10	8	9	4	7	1	3	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

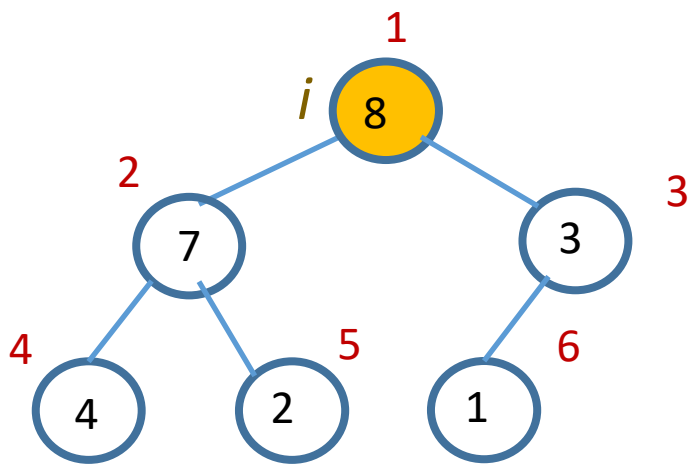
16
14
10



0	1	2	3	4	5	6	7	8	9	10
	9	8	3	4	7	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

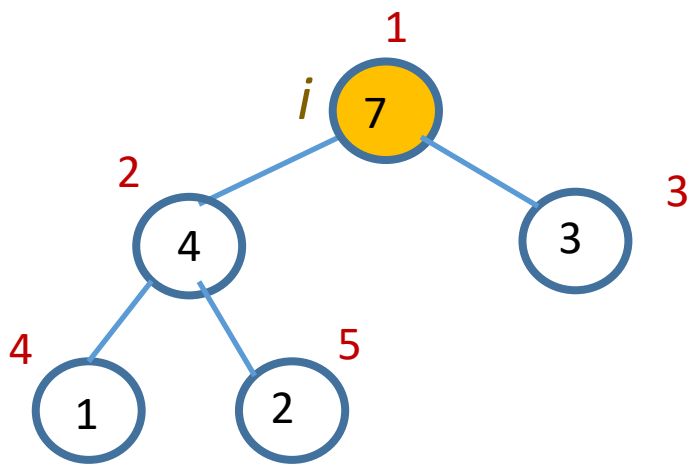


16
14
10
9

0	1	2	3	4	5	6	7	8	9	10
	8	7	3	4	2	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

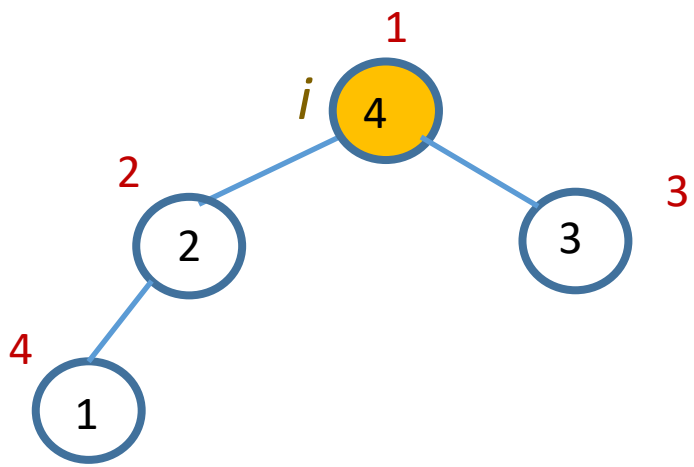


16
14
10
9
8

0	1	2	3	4	5	6	7	8	9	10
	7	4	3	1	2	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

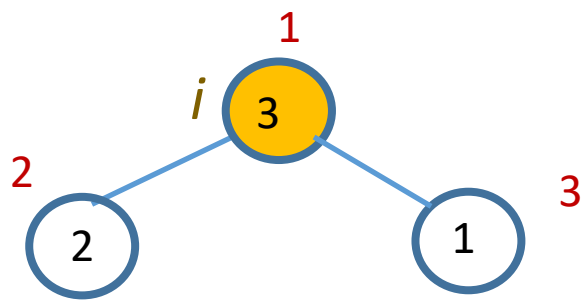


16
14
10
9
8
7

0	1	2	3	4	5	6	7	8	9	10
	4	2	3	1	2	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

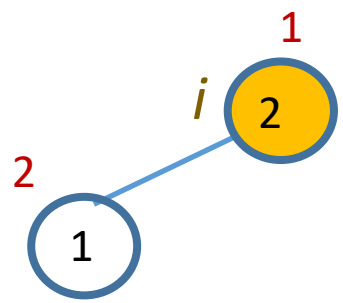


16
14
10
9
8
7
4

0	1	2	3	4	5	6	7	8	9	10
	3	2	1	1	2	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

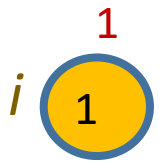


16
14
10
9
8
7
4
3

0	1	2	3	4	5	6	7	8	9	10
	2	1	1	1	2	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap



16
14
10
9
8
7
4
3
2

0	1	2	3	4	5	6	7	8	9	10
	1	1	1	1	2	1	2	2	1	1

A

Deleting the Value at Index *1* in Max-Heap

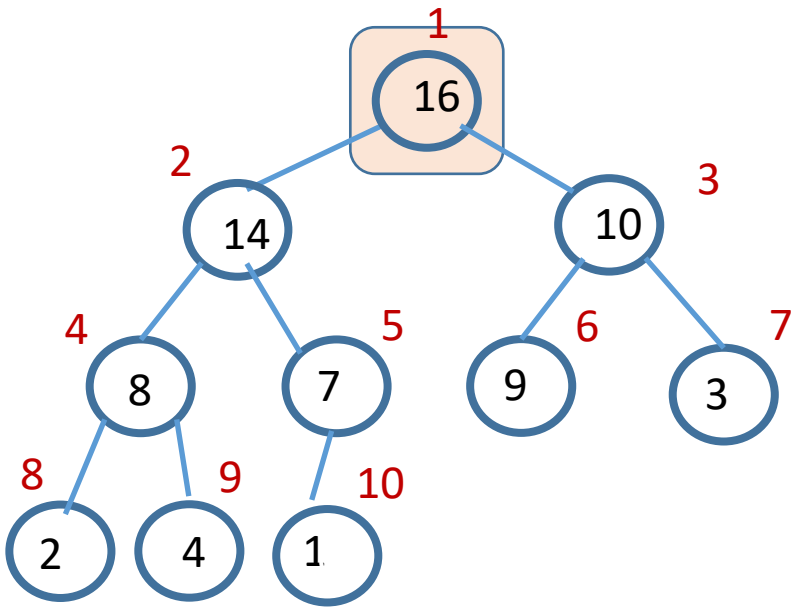
16
14
10
9
8
7
4
3
2
1

0	1	2	3	4	5	6	7	8	9	10
	1	1	1	1	2	1	2	2	1	1

A

Deleting the Values at Index *1* in a Max-Heap

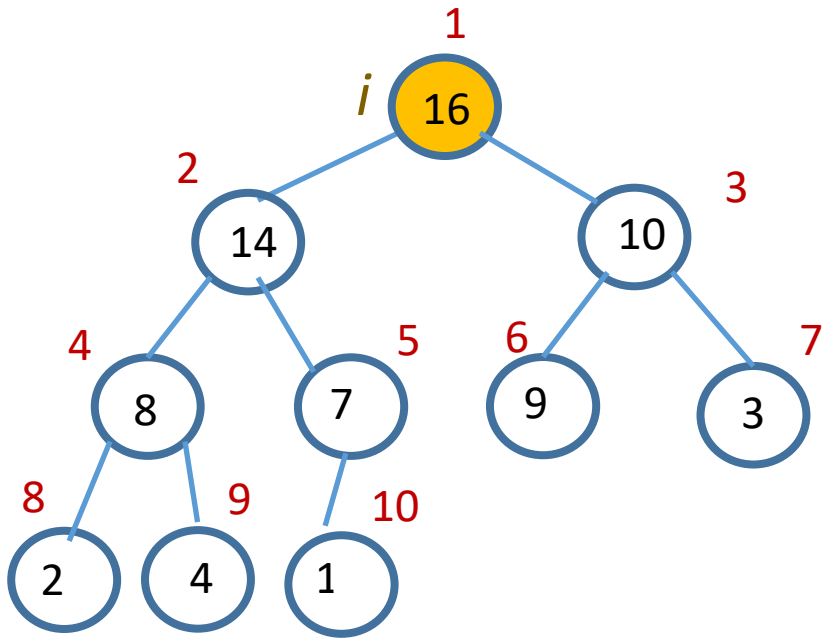
```
void deleteAllMaxHeap (MaxHeap * heap) {

   int deleted;

   for (int i=heap->size; i>=1; i--) {
         deleted = deleteMaxHeap (heap, 1);
         cout << deleted << endl;
   }
}
```



What is the output produced by *deleteAllMaxHeap*?

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[10]



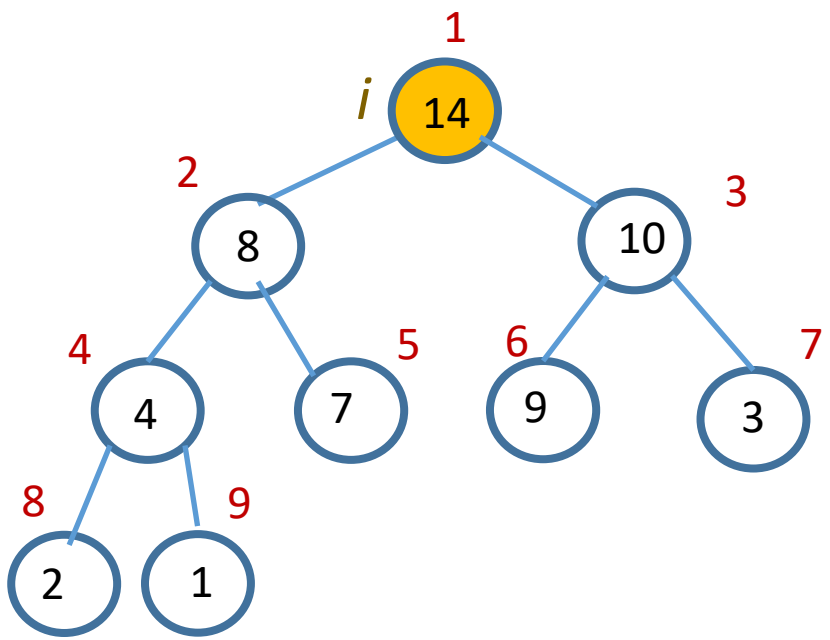
0	1	2	3	4	5	6	7	8	9	10
	16	4	10	14	7	9	3	2	8	1

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[9]

16



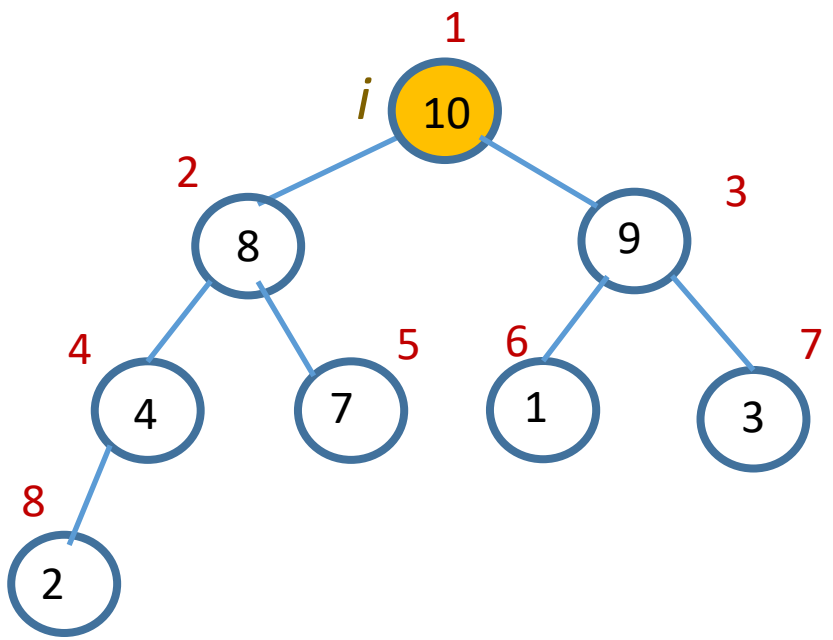
0	1	2	3	4	5	6	7	8	9	10
	14	8	10	4	7	9	3	2	1	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[8]

16
14



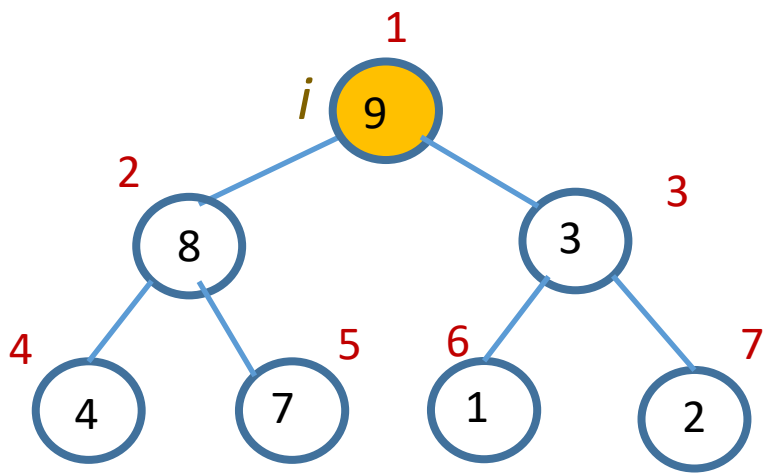
0	1	2	3	4	5	6	7	8	9	10
	10	8	9	4	7	1	3	2	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[7]

16
14
10

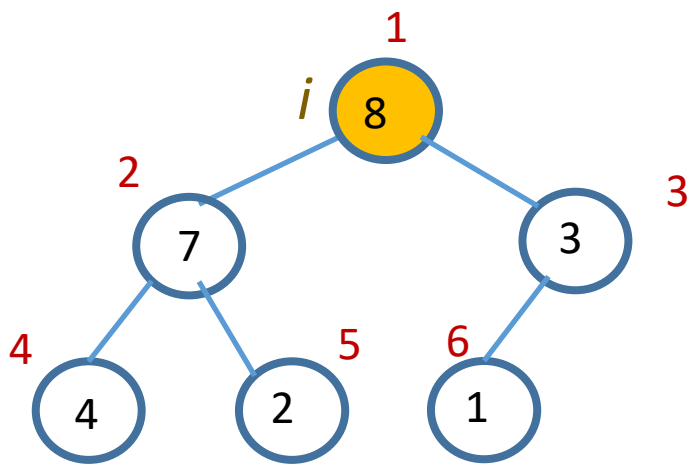


0	1	2	3	4	5	6	7	8	9	10
	9	8	3	4	7	1	2	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[6]



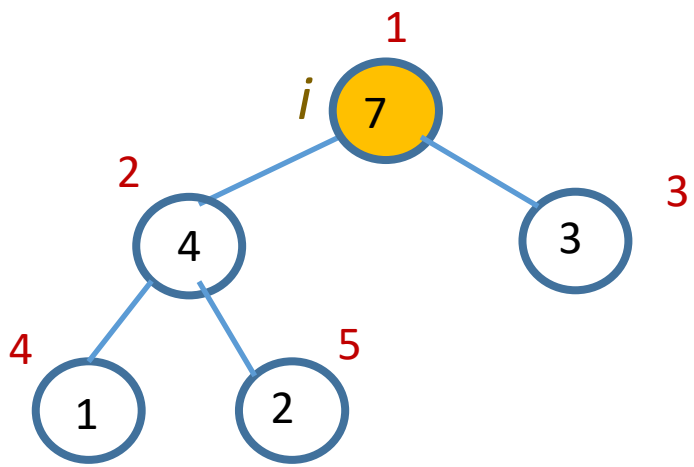
16
14
10
9

0	1	2	3	4	5	6	7	8	9	10
	8	7	3	4	2	1	9	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[5]



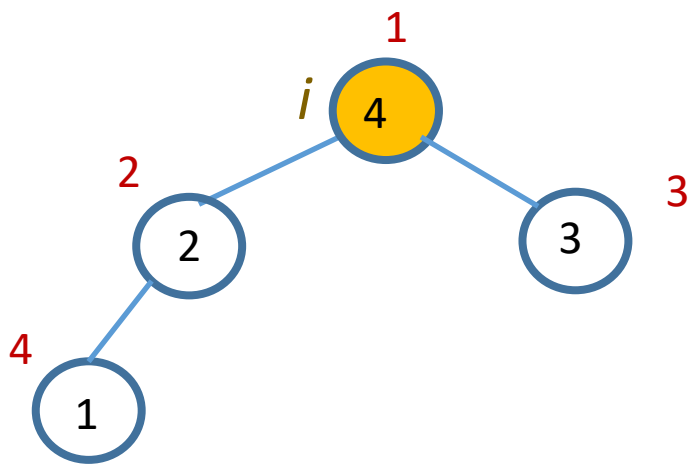
16
14
10
9
8

0	1	2	3	4	5	6	7	8	9	10
	7	4	3	1	2	8	9	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[4]



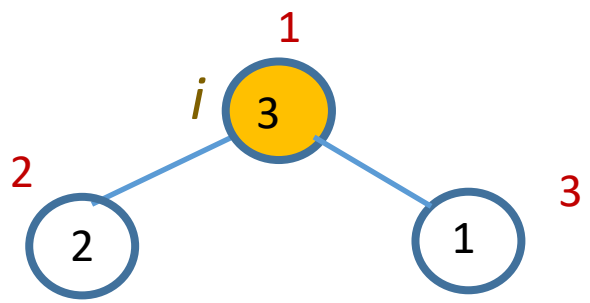
16
14
10
9
8
7

0	1	2	3	4	5	6	7	8	9	10
	4	2	3	1	7	8	9	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[3]



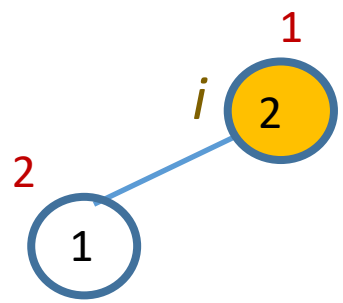
16
14
10
9
8
7
4

0	1	2	3	4	5	6	7	8	9	10
	3	2	1	4	7	8	9	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[2]



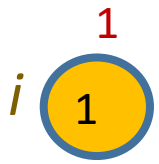
16
14
10
9
8
7
4
3

0	1	2	3	4	5	6	7	8	9	10
	2	1	3	4	7	8	9	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

- Store in *A*[1]



16
14
10
9
8
7
4
3
2

0	1	2	3	4	5	6	7	8	9	10
	1	2	3	4	7	8	9	10	14	16

A

Deleting the Value at Index *1* in Max-Heap

16
14
10
9
8
7
4
3
2
1

0	1	2	3	4	5	6	7	8	9	10
	1	2	3	4	7	8	9	10	14	16

A

Deleting the Values at Index *1* in a Max-Heap

```
void deleteAllMaxHeap (MaxHeap * heap) {

   for (int i=heap->size; i>=1; i--)
         heap->A[i] = deleteMaxHeap (heap, 1);

}
```

initMaxHeapFromArray Function

```
MaxHeap * initMaxHeapFromArray (int A[], int numElements) {

   MaxHeap * heap;

   heap = new MaxHeap;

   for (int i=0; i<numElements; i++) {
         heap->A[i+1] = A[i];
   }

   heap->size = numElements;

   return heap;
}
```

Heap Sort

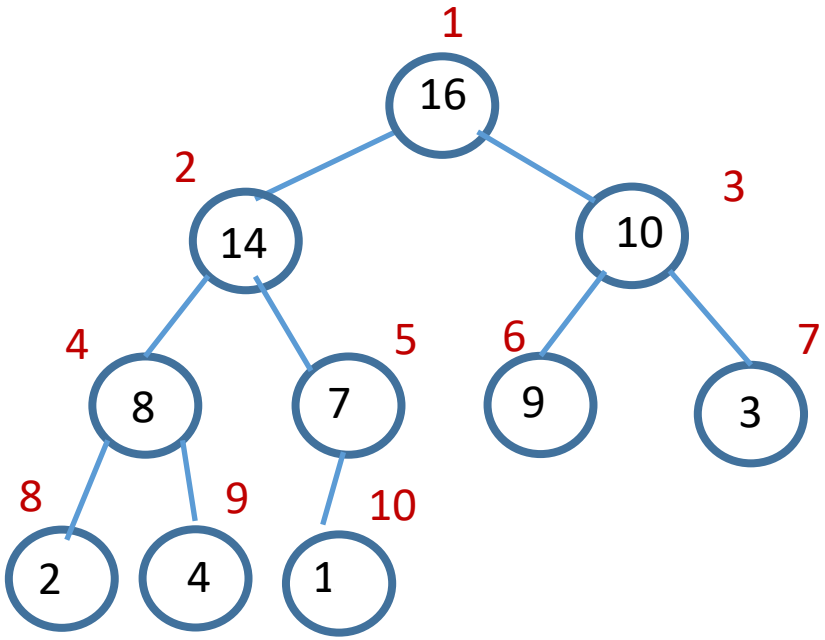
```
void heapSort (int A[], int numElements) {

   MaxHeap * heap = initMaxHeapFromArray (A, numElements);

   buildMaxHeap (heap);
   deleteAllMaxHeap (heap);

   for (int i=1; i<=numElements; i++)
        A[i-1] = heap->A[i];
}
```

Heap Sort (Version 2)

```
void heapSort (int A[], int numElements) {

    MaxHeap * heap = initMaxHeapFromArray (A, numElements);

    buildMaxHeap (heap);
    for (int i=heap->size; i>=2; i--) {
        int temp = heap->A[i];
        heap->A[i] = heap->A[1];
        heap->A[1] = temp;
        heap->size = heap->size - 1;
        maxHeapify(heap, 1);
    }

    for (int i=1; i<=numElements; i++)
          A[i-1] = heap->A[i];
}
```

Code to delete biggest element (at location 1)

Inserting an Element in a Max-Heap

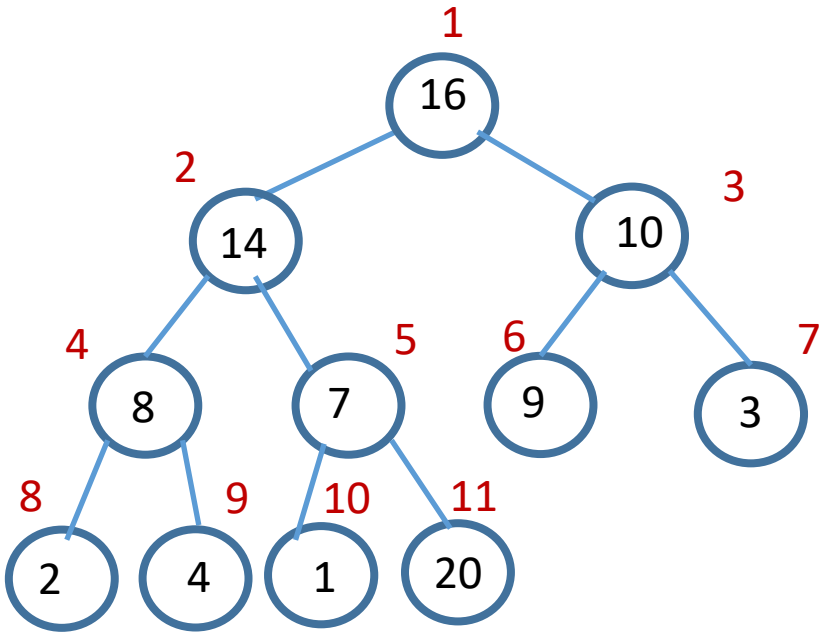


Insert 20

0	1	2	3	4	5	6	7	8	9	10
	16	14	10	8	7	9	3	2	4	1

A

Inserting an Element in a Max-Heap

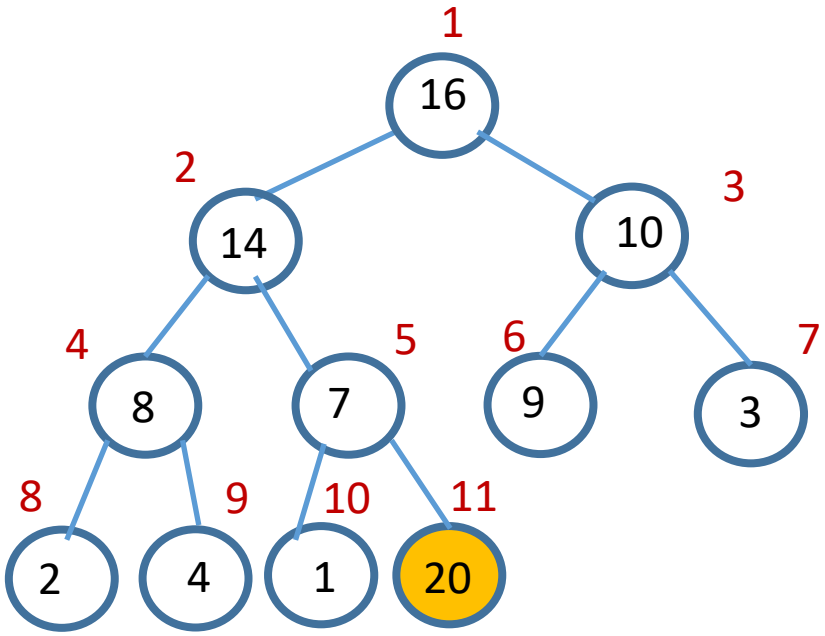


Insert 20

0	1	2	3	4	5	6	7	8	9	10	11
	16	14	10	8	7	9	3	2	4	1	**20**

A

Inserting an Element in a Max-Heap

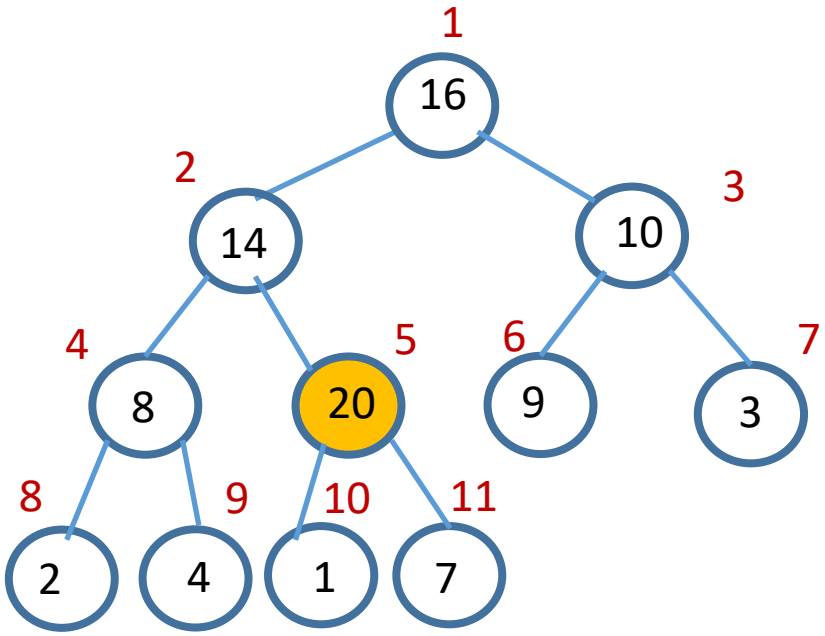


Insert 20

0	1	2	3	4	5	6	7	8	9	10	11
	16	14	10	8	7	9	3	2	4	1	**20**

A

Inserting an Element in a Max-Heap

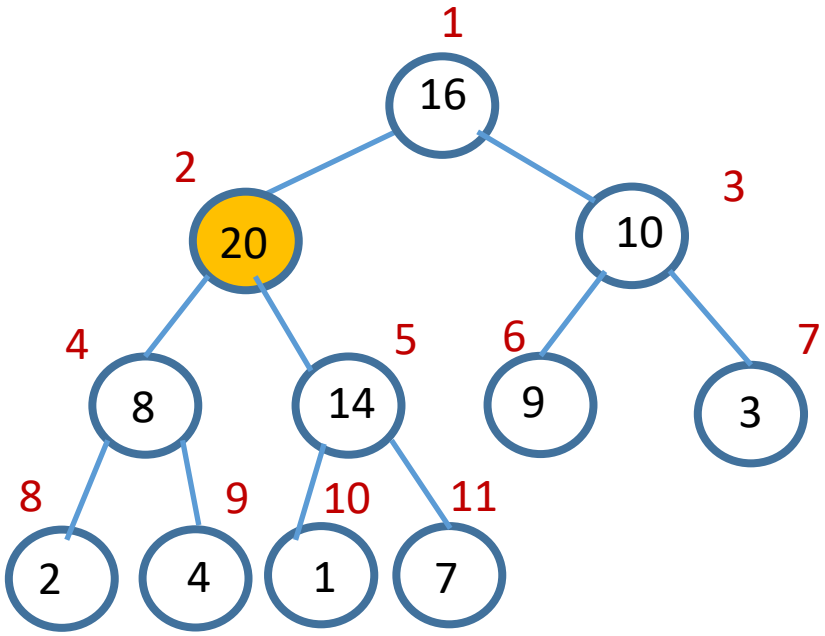


Insert 20

0	1	2	3	4	5	6	7	8	9	10	11
	16	14	10	8	**20**	9	3	2	4	1	**7**

A

Inserting an Element in a Max-Heap

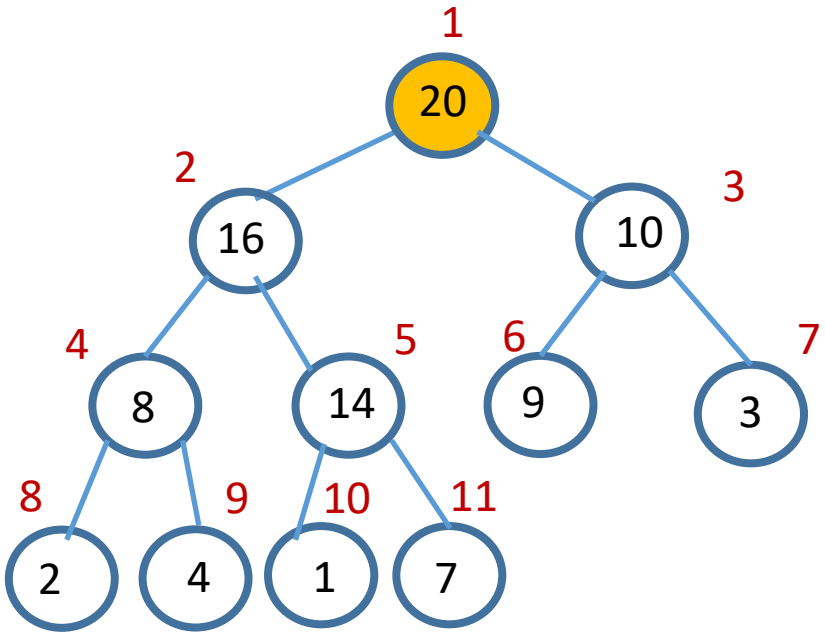


Insert 20

0	1	2	3	4	5	6	7	8	9	10	11
	16	**20**	10	8	**14**	9	3	2	4	1	7

A

Inserting an Element in a Max-Heap

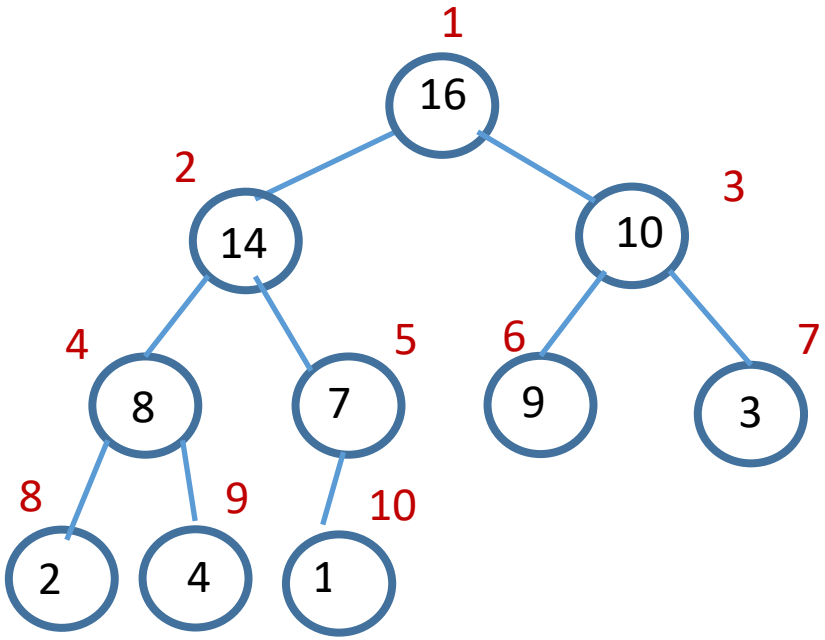


Insert 20

0	1	2	3	4	5	6	7	8	9	10	11
	20	**16**	10	8	14	9	3	2	4	1	7

A

Inserting an Element in a Max-Heap

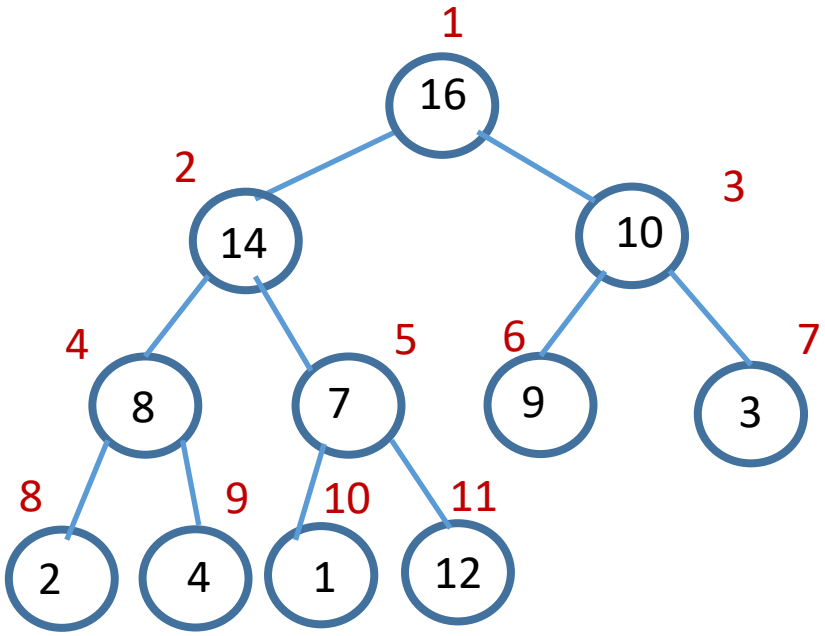


Insert 12

0	1	2	3	4	5	6	7	8	9	10
	16	14	10	8	7	9	3	2	4	1

A

Inserting an Element in a Max-Heap

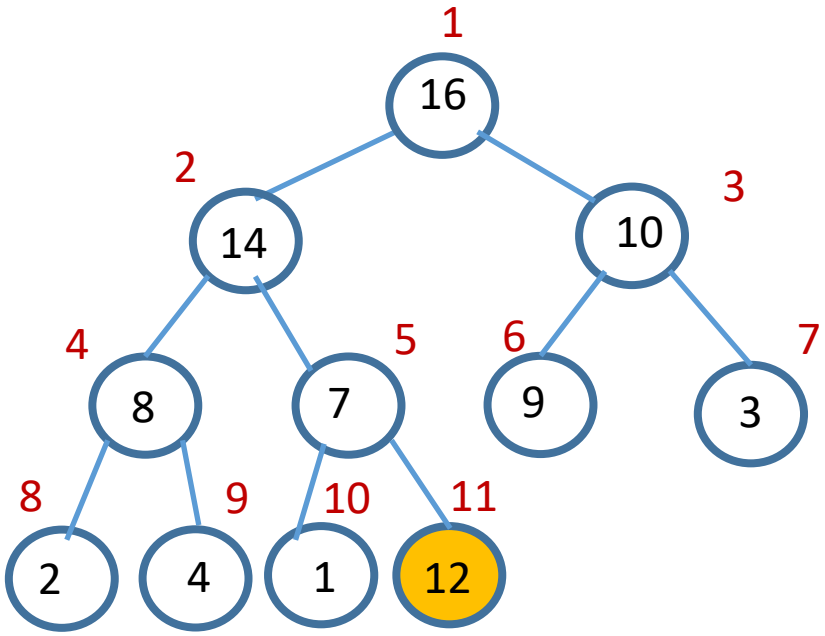


Insert 12

0	1	2	3	4	5	6	7	8	9	10	11
	16	14	10	8	7	9	3	2	4	1	12

A

Inserting an Element in a Max-Heap

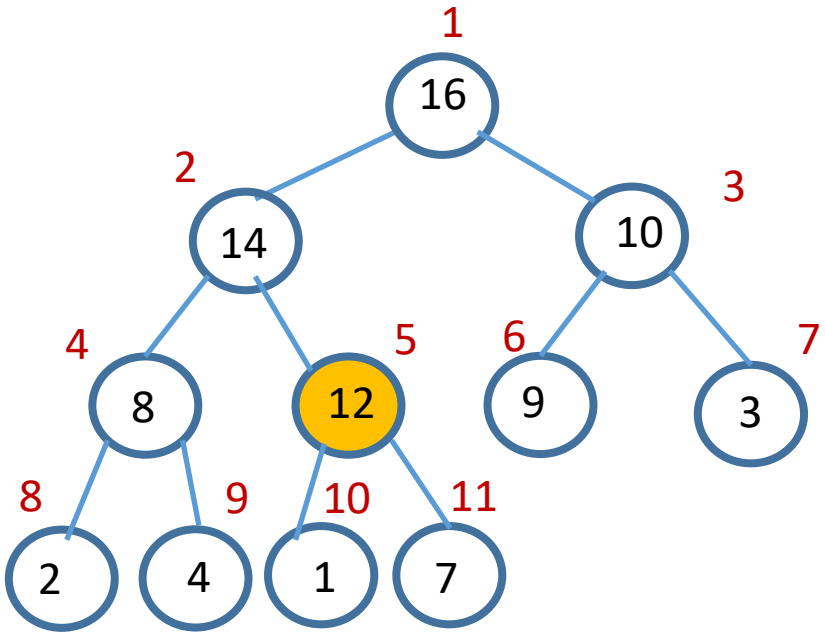


Insert 12

0	1	2	3	4	5	6	7	8	9	10	11
	16	14	10	8	7	9	3	2	4	1	12

A

Inserting an Element in a Max-Heap



Insert 12

0	1	2	3	4	5	6	7	8	9	10	11
	16	14	10	8	12	9	3	2	4	1	7

A

Insert Function

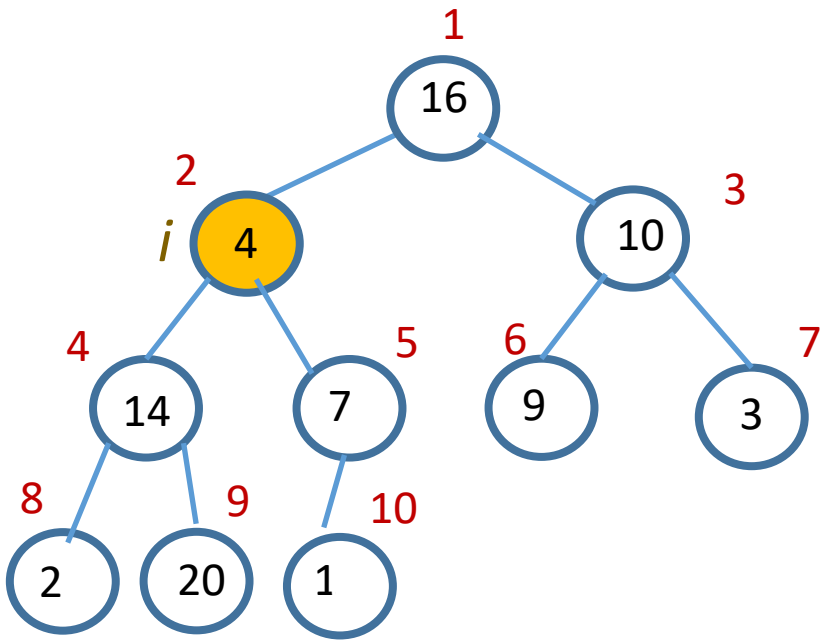
```
void insertMaxHeap (MaxHeap * heap, int data) {
  heap->size = heap->size + 1;   // one more, so add 1
  heap->A[heap->size] = data;    // insert at end

  int i = heap->size;
```

A[Parent (i)] < A (i]

```
  while (i > 1 && heap->A[i/2] < heap->A[i]) {
     int temp = heap->A[i/2];    // swap with parent
     heap->A[i/2] = heap->A[i];
     heap->A[i] = temp;

     i = i / 2;                  // i is now parent
  }
}
```

Revisiting Heaps: Maintaining the Max-Heap Property

Heapify Node 2



Can't Heapify Node 2 since its left subtree is not a max-heap

Operations on a Max-Heap

- Building a max-heap from a random set of values.
- Finding the biggest element in a max-heap.
- Deleting an element from a max-heap.
- Inserting an element in a max-heap.
- Sorting the elements of an array in ascending order using a max-heap.

How to find out if a max-heap contains a particular value?