Principles of Computer Science II

Abstract Data Types

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 11

Heap Data Structures

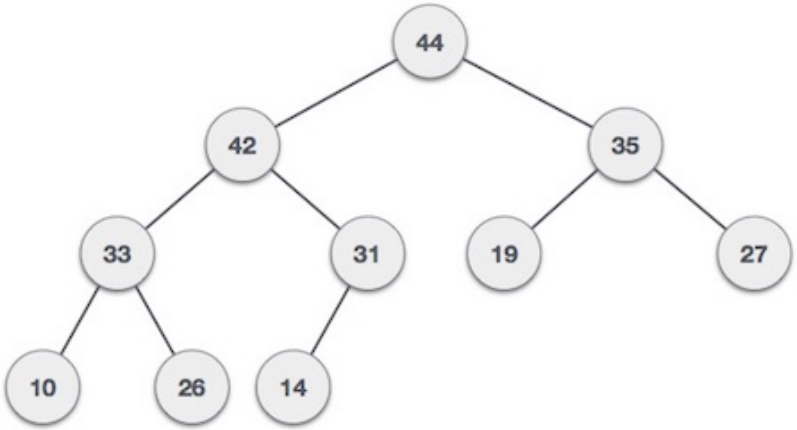
Heap is a special case of balanced binary tree data structure where the root-node key is compared with its children and arranged accordingly.

If α has child node β then

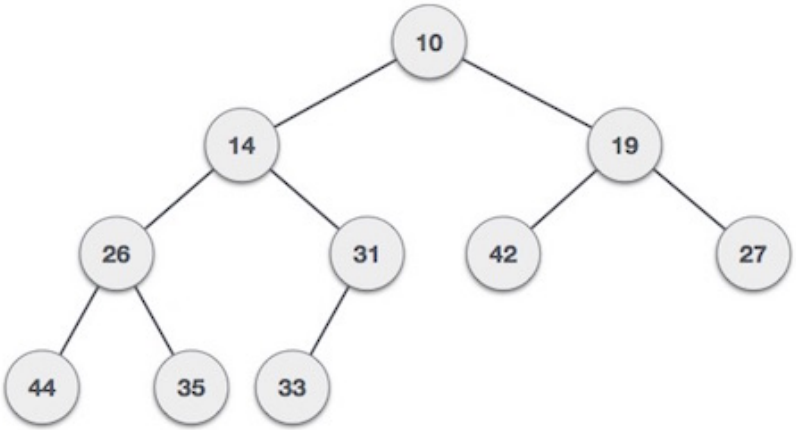
$$key(\alpha) \geq key(\beta)$$

- As the value of parent is greater than that of child, this property generates Max Heap.
- By changing the criterion we can also get a Min Heap.

Max Heap



Min Heap



Max Heap – Add

1. Create a new node at the end of heap.
2. Assign new value to the node.
3. Compare the value of this child node with its parent.
4. If value of parent is less than child, then swap them.
5. Repeat step 3 & 4 until Heap property holds.

Note – In Min Heap construction algorithm, we expect the value of the parent node to be less than that of the child node.

Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input **35** 33 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input **35** 33 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input **35** 33 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 **33** 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 **33** 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 **33** 42 10 14 19 27 44 26 31

Max Heap – Add Example

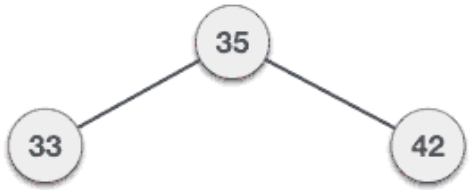
Input 35 **33** 42 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 33 **42** 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 33 **42** 10 14 19 27 44 26 31



Max Heap – Add Example

Input 35 33 **42** 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 33 **42** 10 14 19 27 44 26 31

Max Heap – Add Example

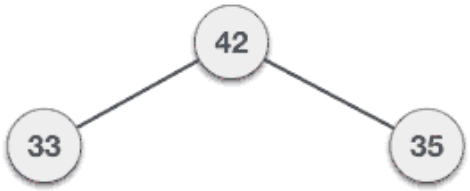
Input 35 33 **42** 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 33 **42** 10 14 19 27 44 26 31

Max Heap – Add Example

Input 35 33 **42** 10 14 19 27 44 26 31

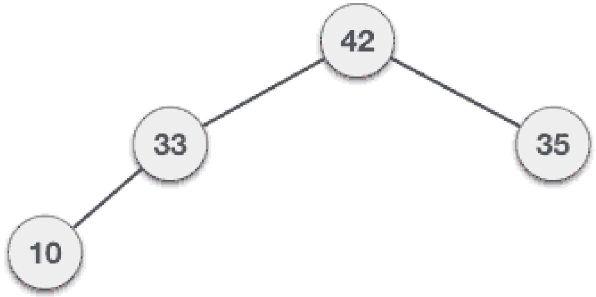


Max Heap – Add Example

Input 35 33 42 **10** 14 19 27 44 26 31

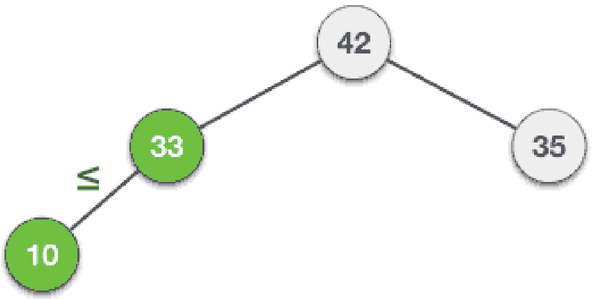
Max Heap – Add Example

Input 35 33 42 **10** 14 19 27 44 26 31



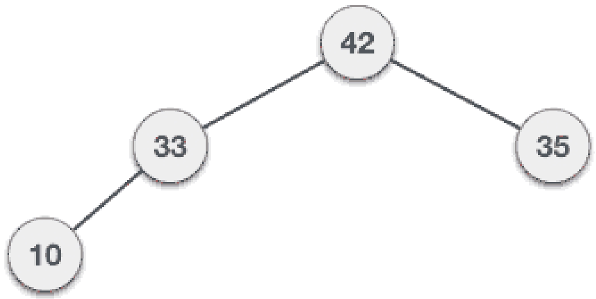
Max Heap – Add Example

Input 35 33 42 **10** 14 19 27 44 26 31



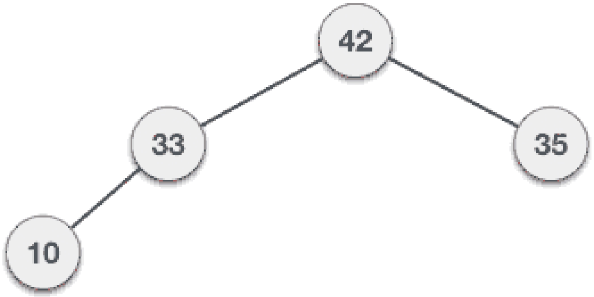
Max Heap – Add Example

Input 35 33 42 **10** 14 19 27 44 26 31



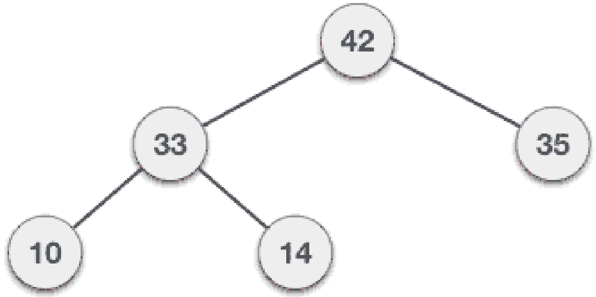
Max Heap – Add Example

Input 35 33 42 10 **14** 19 27 44 26 31



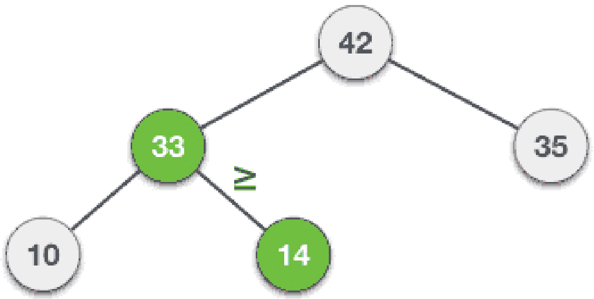
Max Heap – Add Example

Input 35 33 42 10 **14** 19 27 44 26 31



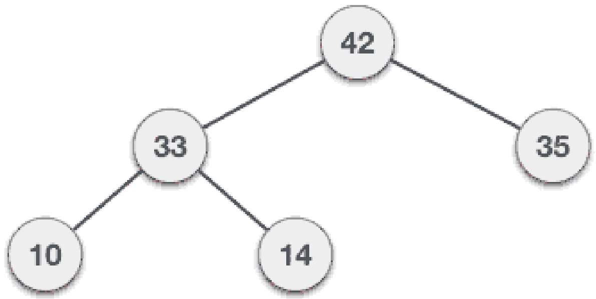
Max Heap – Add Example

Input 35 33 42 10 **14** 19 27 44 26 31



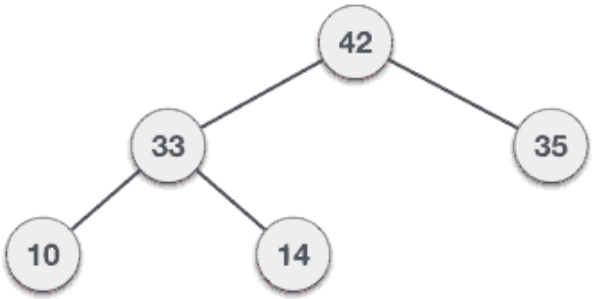
Max Heap – Add Example

Input 35 33 42 10 **14** 19 27 44 26 31



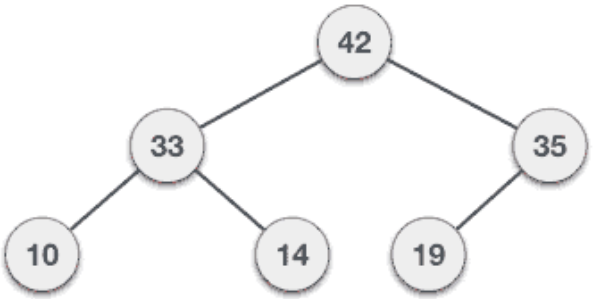
Max Heap – Add Example

Input 35 33 42 10 14 **19** 27 44 26 31



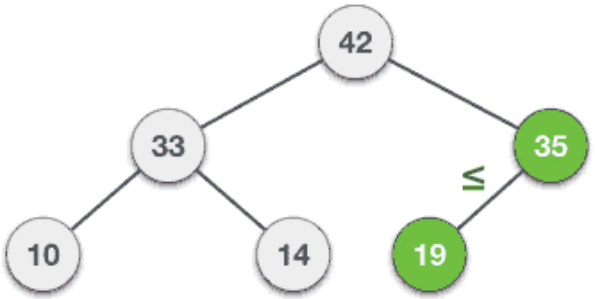
Max Heap – Add Example

Input 35 33 42 10 14 **19** 27 44 26 31



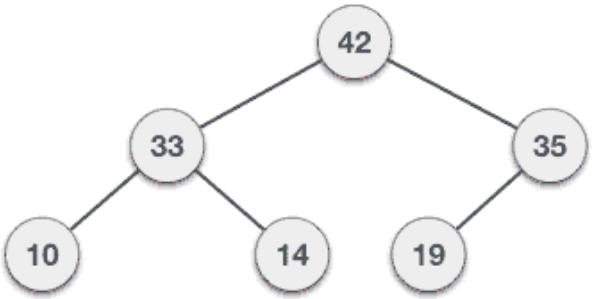
Max Heap – Add Example

Input 35 33 42 10 14 **19** 27 44 26 31



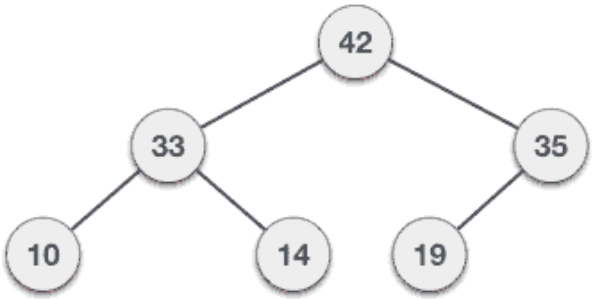
Max Heap – Add Example

Input 35 33 42 10 14 **19** 27 44 26 31



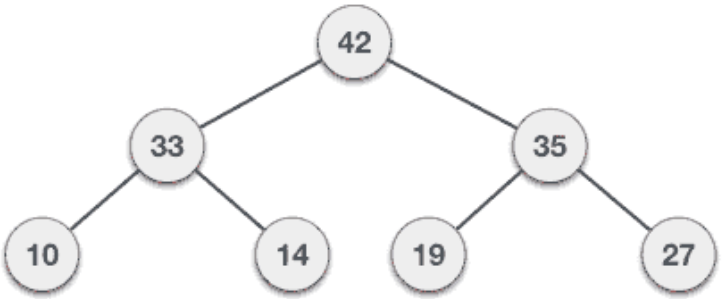
Max Heap – Add Example

Input 35 33 42 10 14 19 **27** 44 26 31



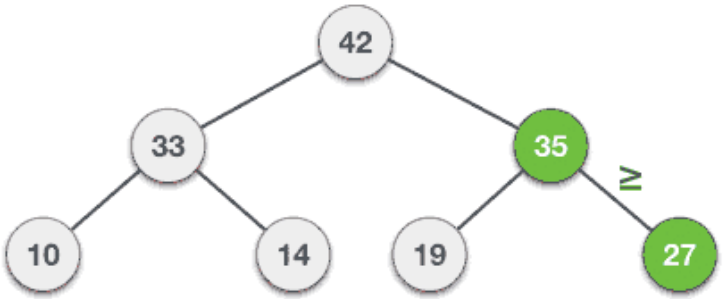
Max Heap – Add Example

Input 35 33 42 10 14 19 **27** 44 26 31



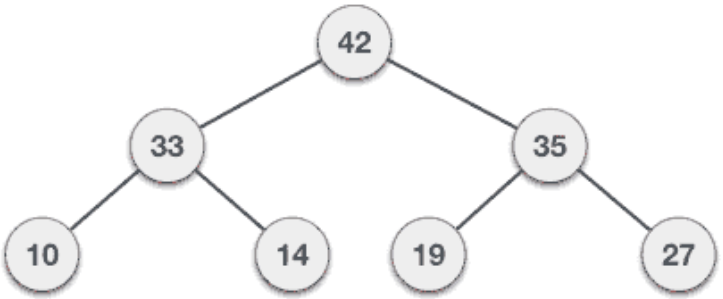
Max Heap – Add Example

Input 35 33 42 10 14 19 **27** 44 26 31



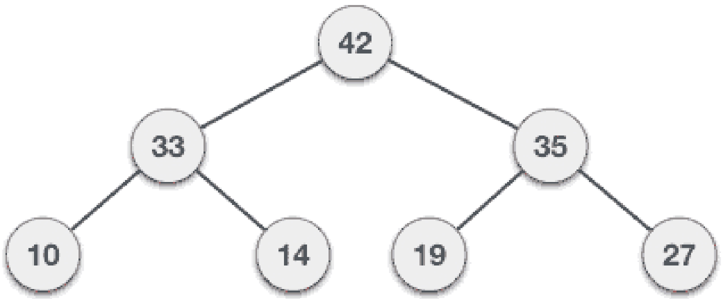
Max Heap – Add Example

Input 35 33 42 10 14 19 **27** 44 26 31



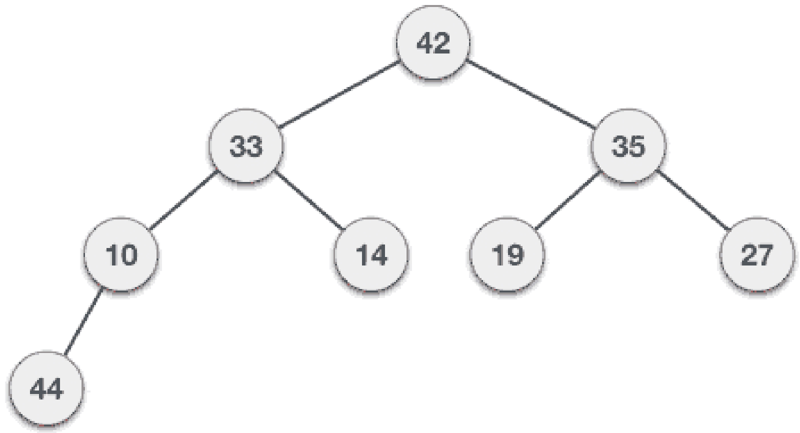
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



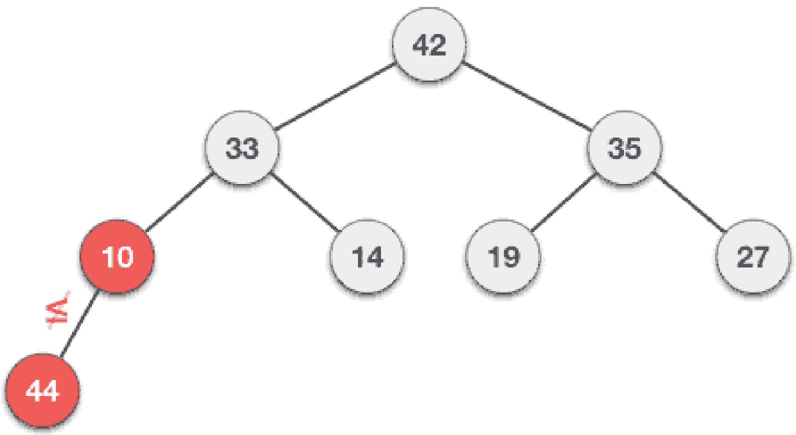
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



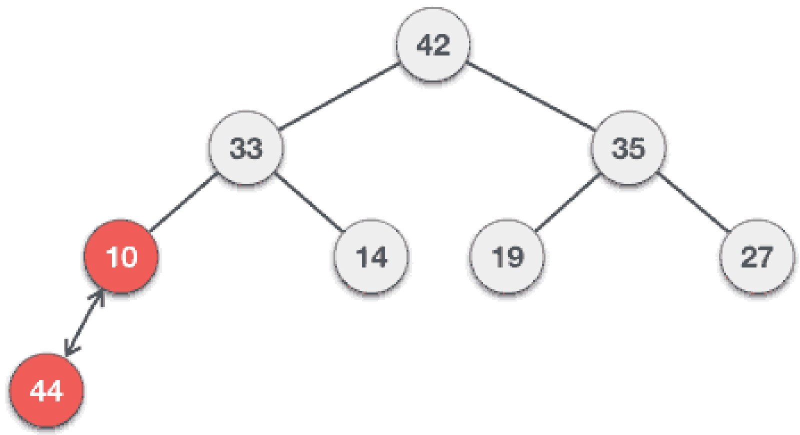
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31

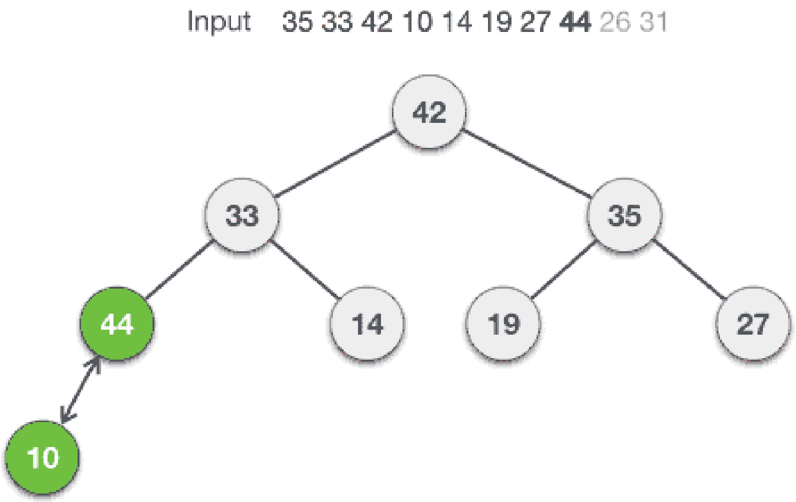


Max Heap – Add Example

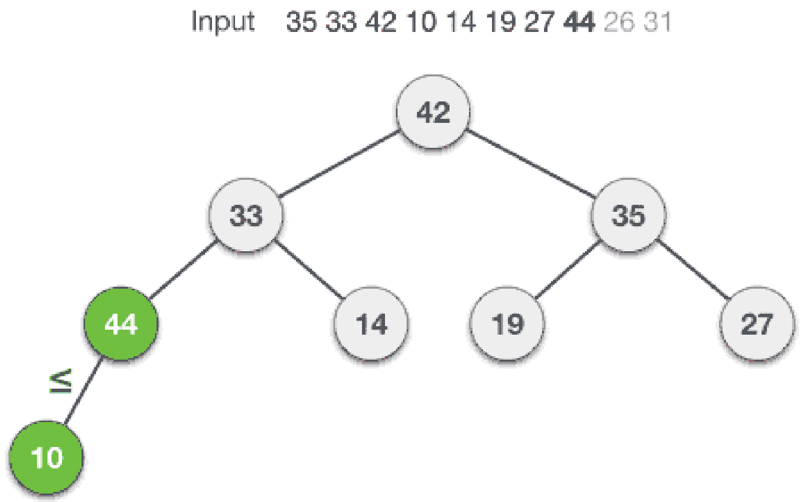
Input 35 33 42 10 14 19 27 **44** 26 31



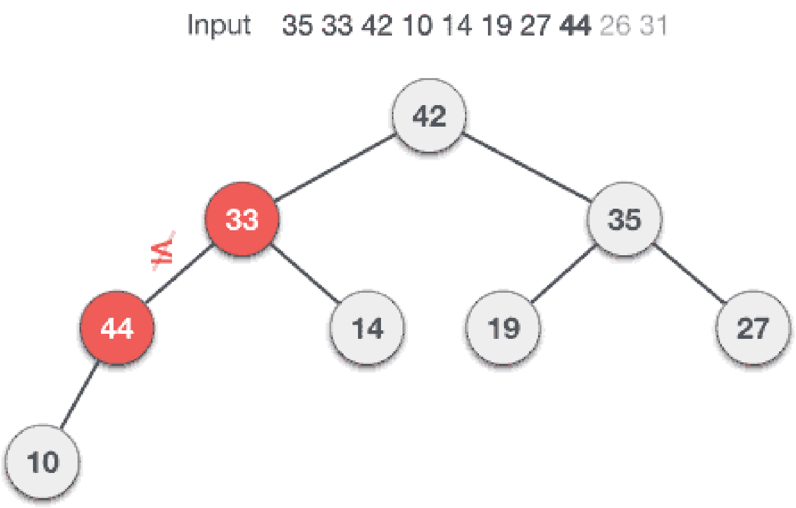
Max Heap – Add Example



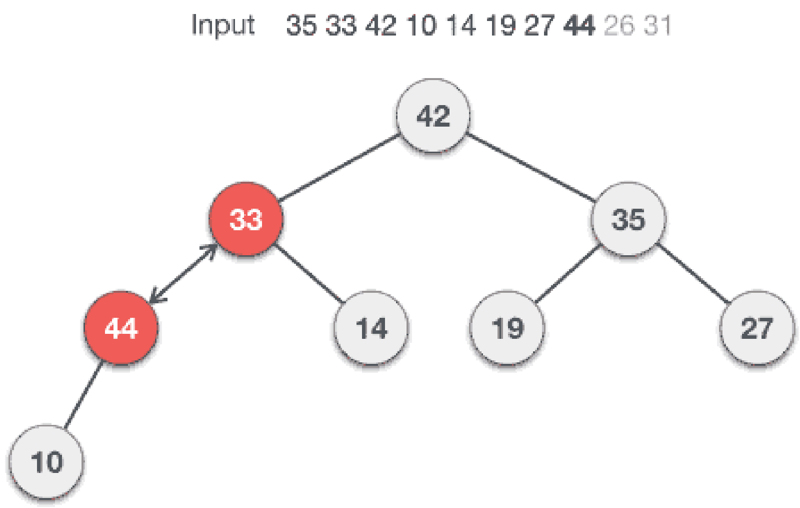
Max Heap – Add Example



Max Heap – Add Example

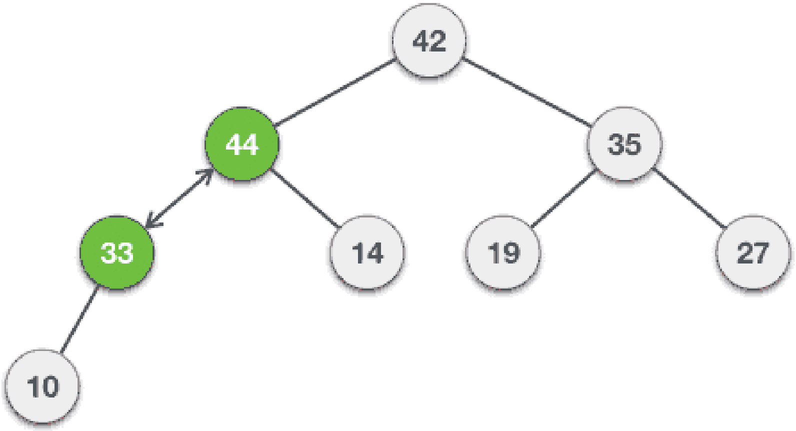


Max Heap – Add Example



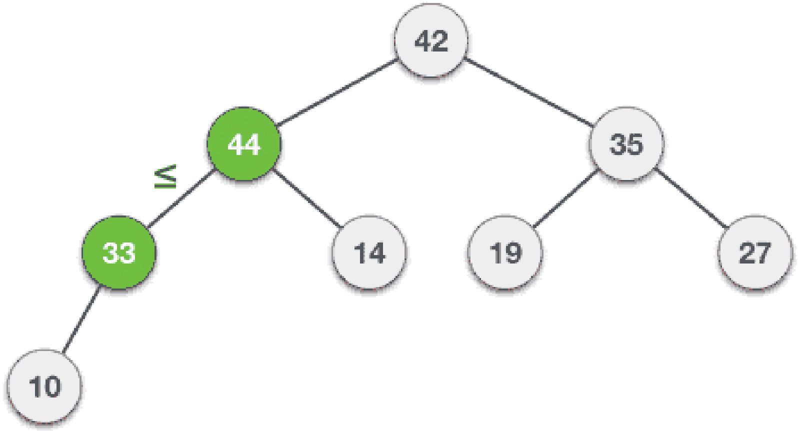
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



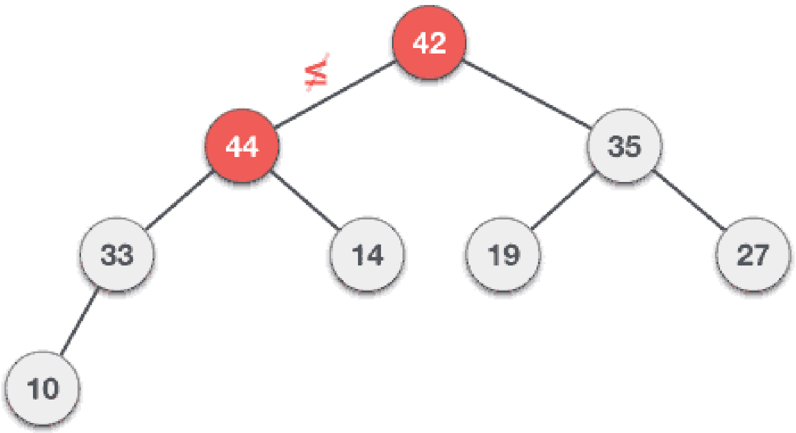
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



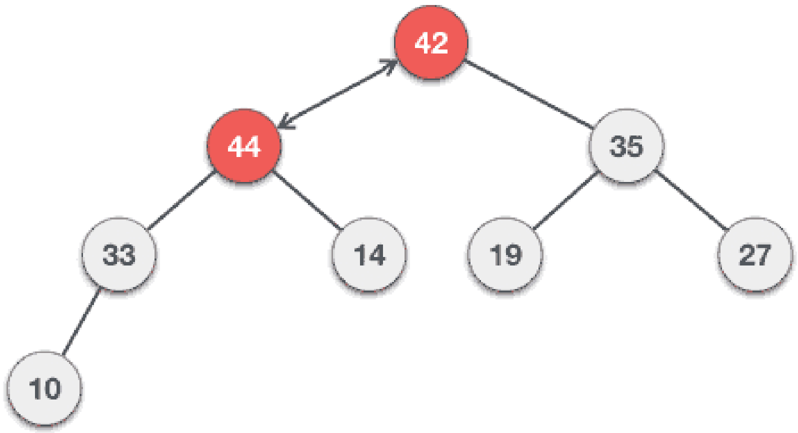
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



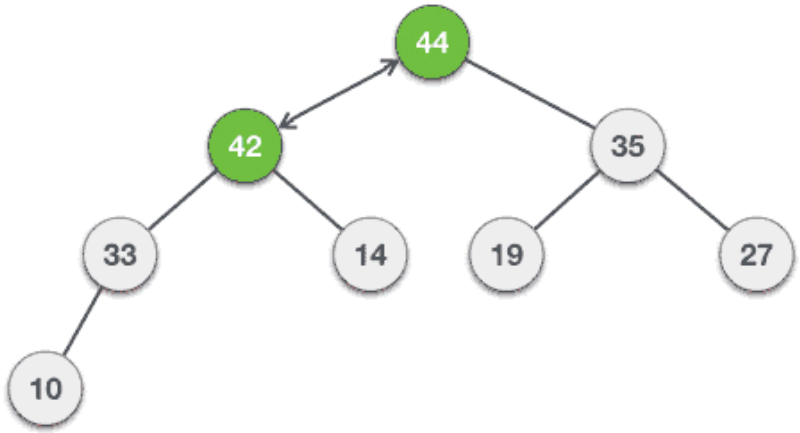
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



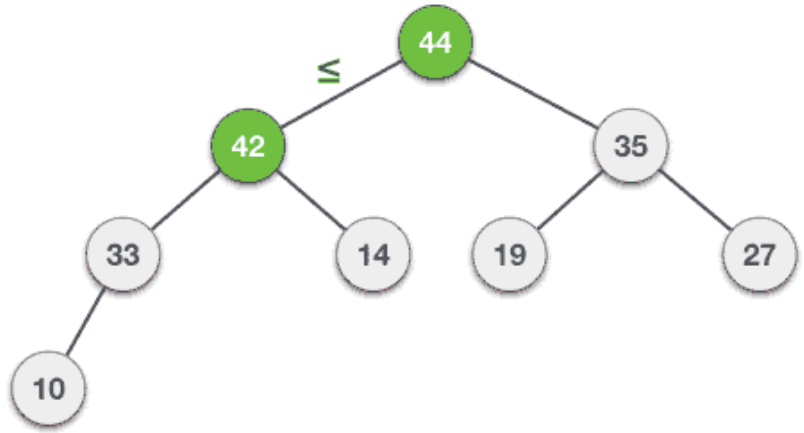
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



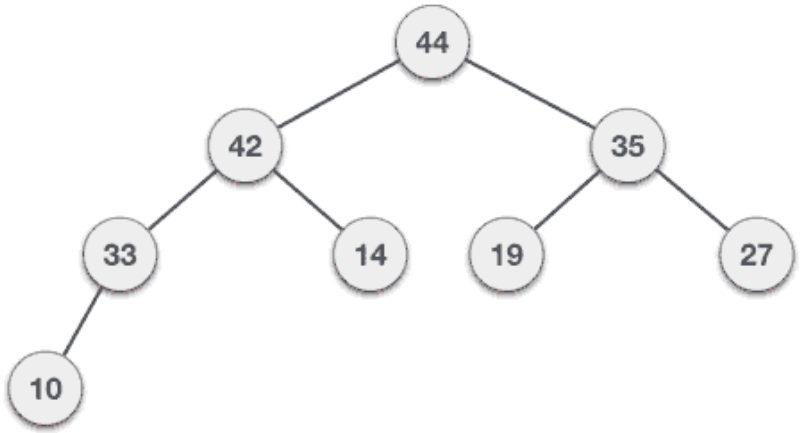
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



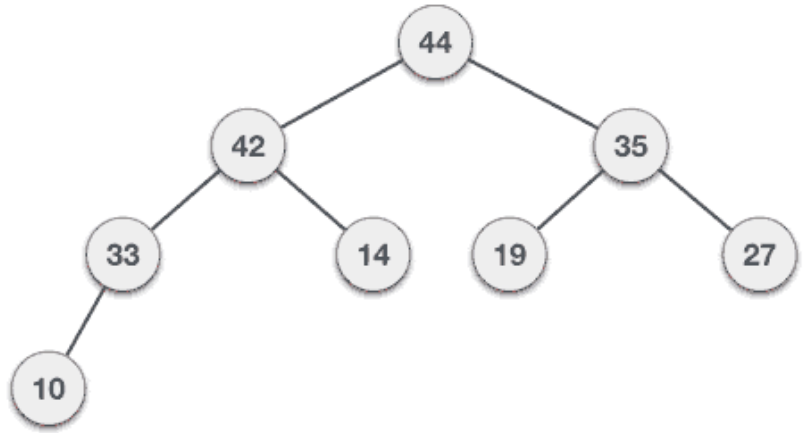
Max Heap – Add Example

Input 35 33 42 10 14 19 27 **44** 26 31



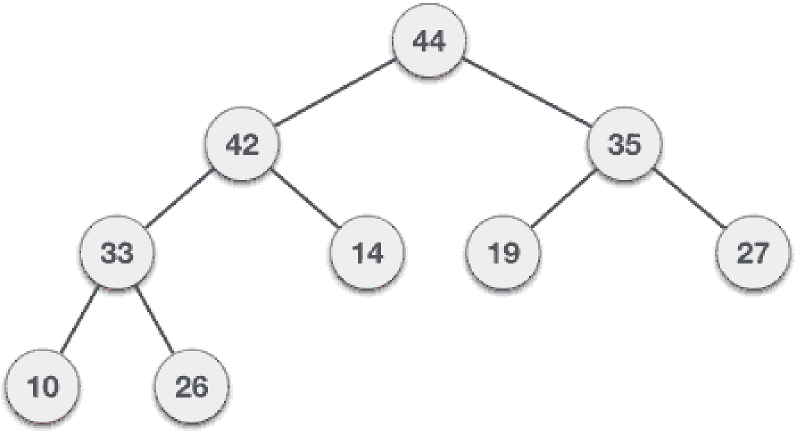
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 **26** 31



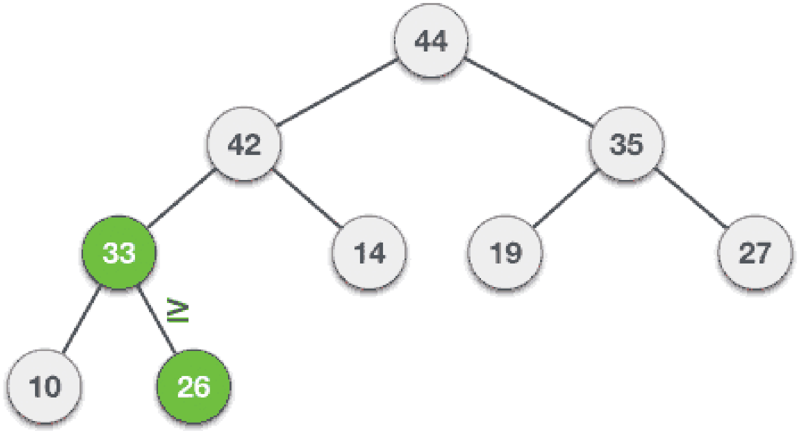
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 **26** 31



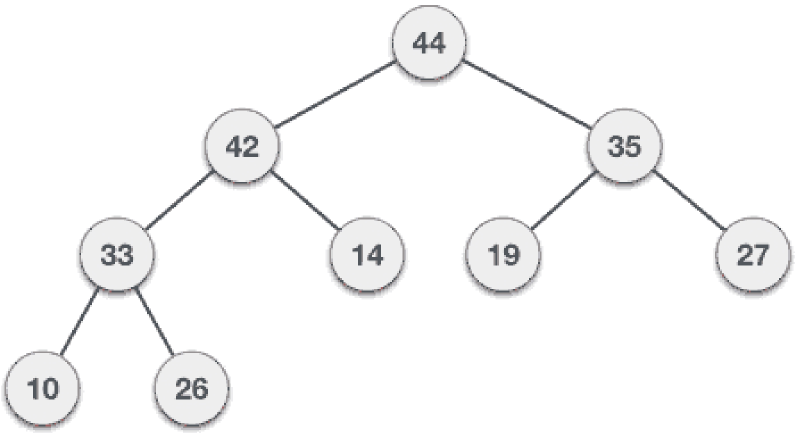
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 **26** 31



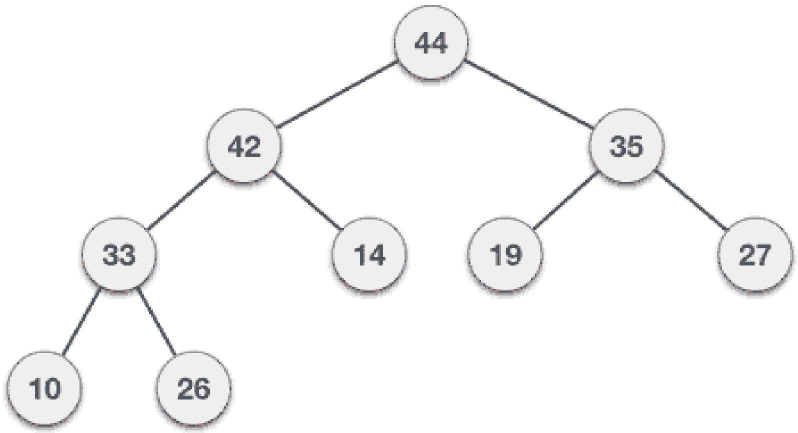
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 **26** 31



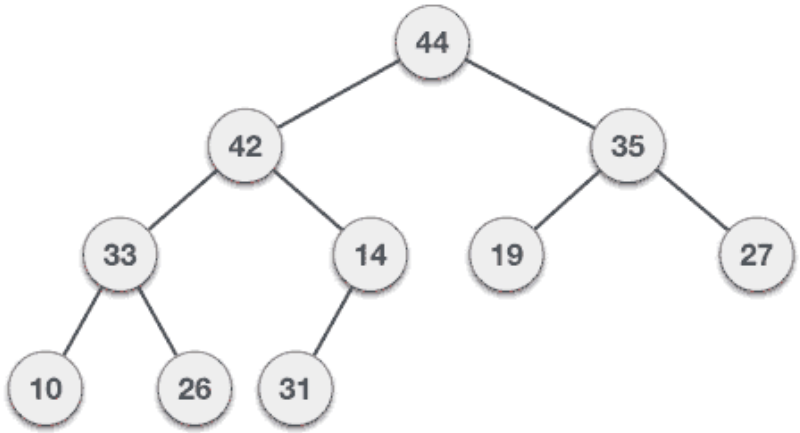
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 **31**



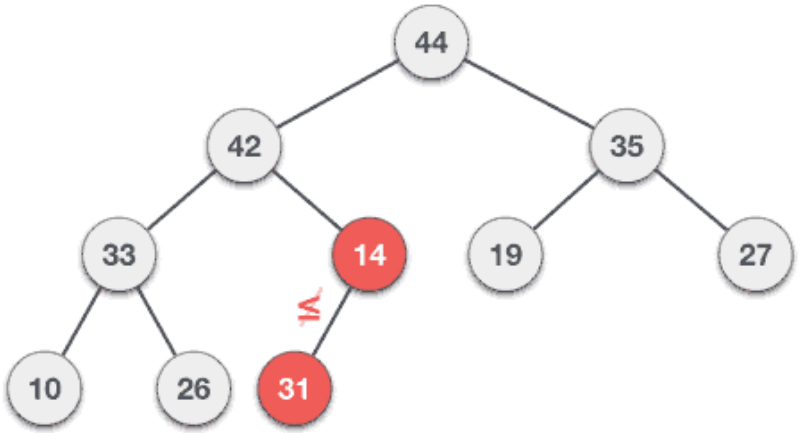
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 **31**



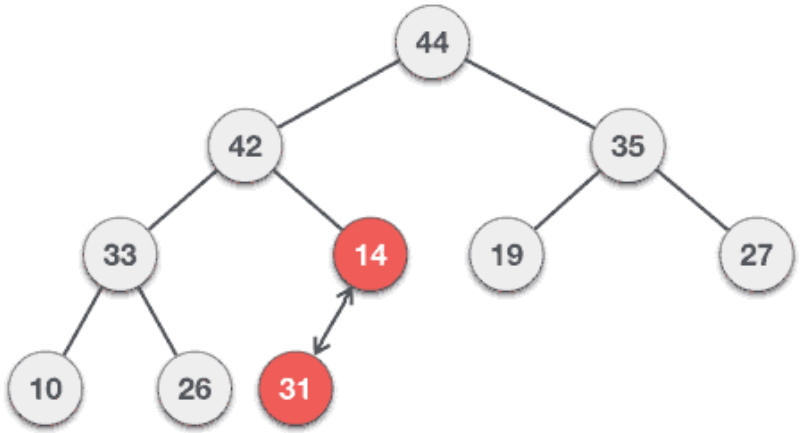
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 **31**



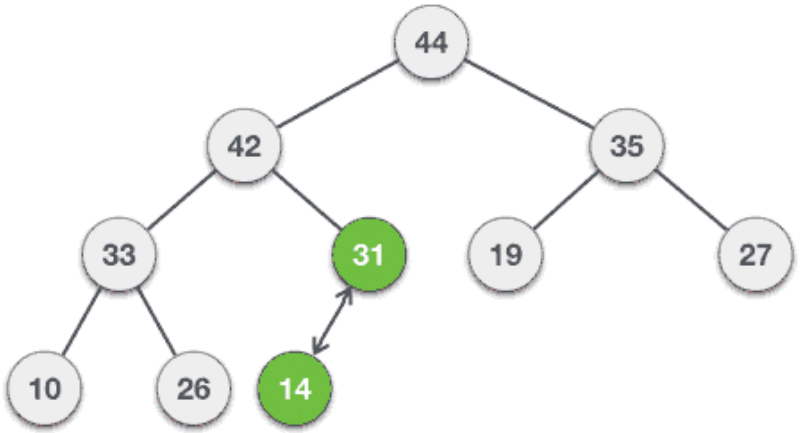
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 **31**



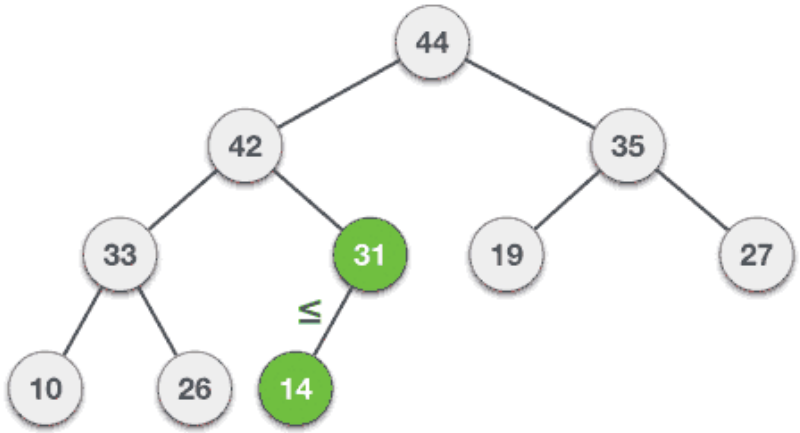
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 **31**



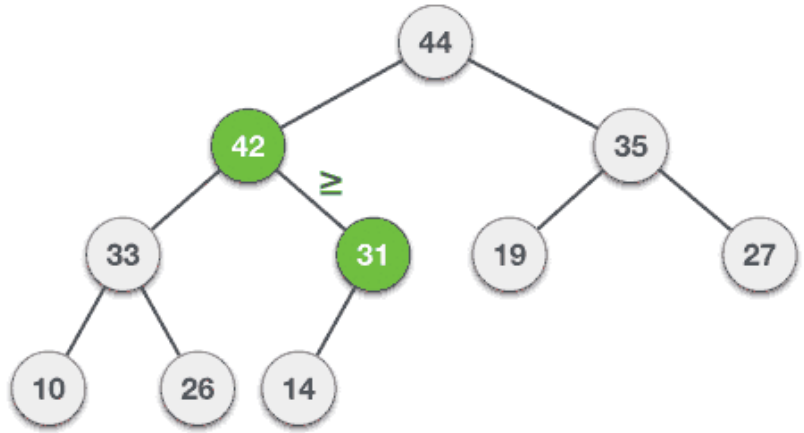
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 **31**



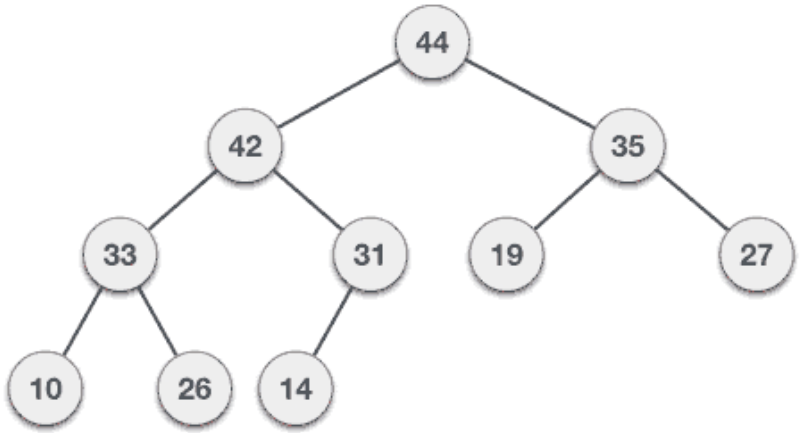
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 31



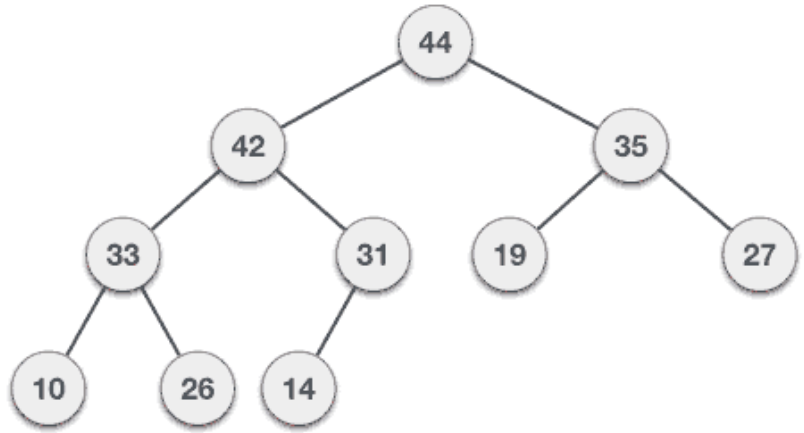
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 31



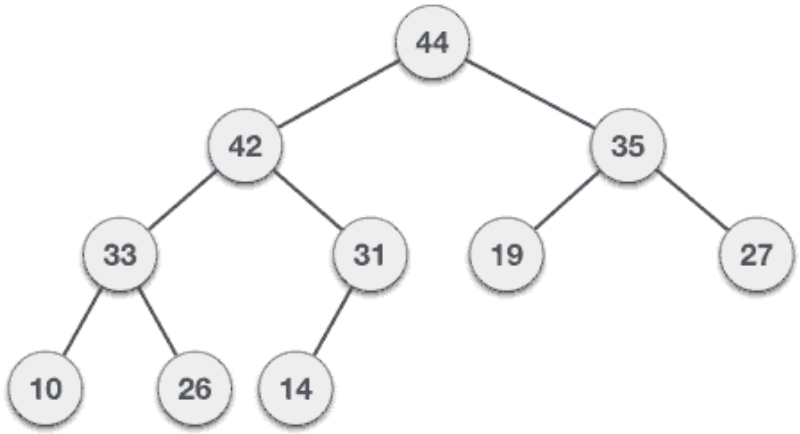
Max Heap – Add Example

Input 35 33 42 10 14 19 27 44 26 31



Max Heap – Add Example

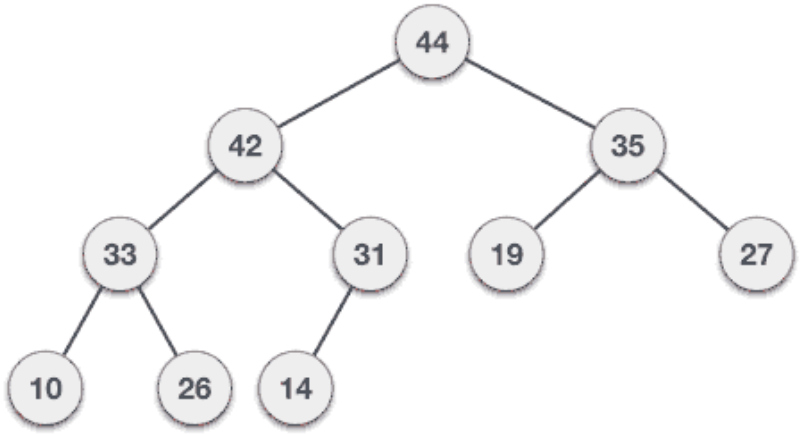
Input 35 33 42 10 14 19 27 44 26 31



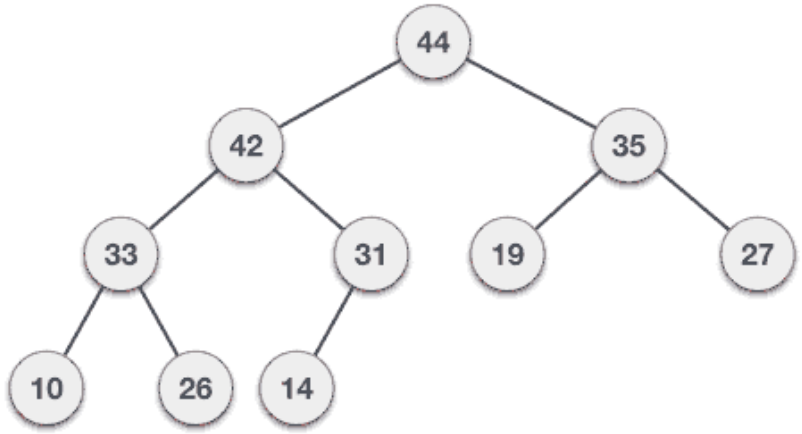
Max Heap – Deletion

1. Remove root node.
2. Move the last element of last level to root.
3. Compare the value of this child node with its parent.
4. If value of parent is less than child, then swap them.
5. Repeat step 3 & 4 until Heap property holds.

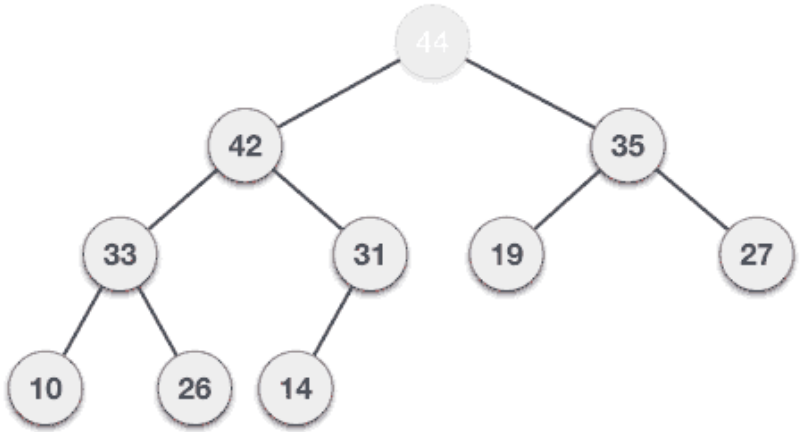
Max Heap – Deletion Example



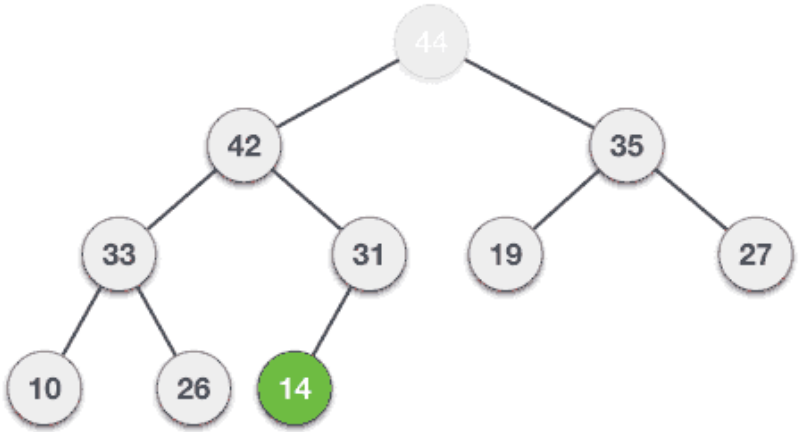
Max Heap – Deletion Example



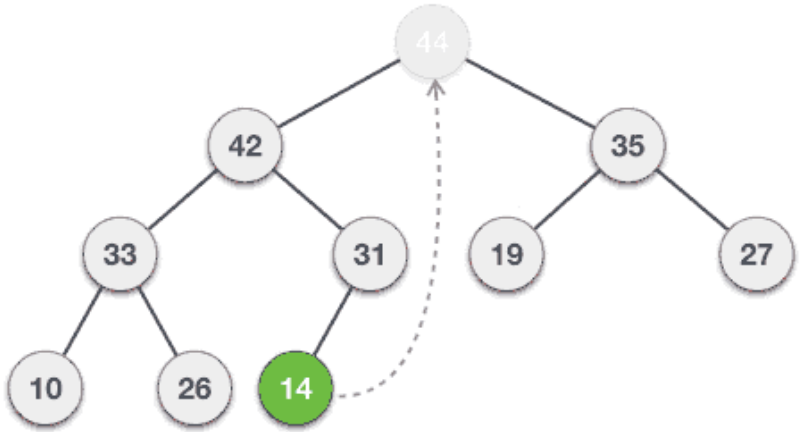
Max Heap – Deletion Example



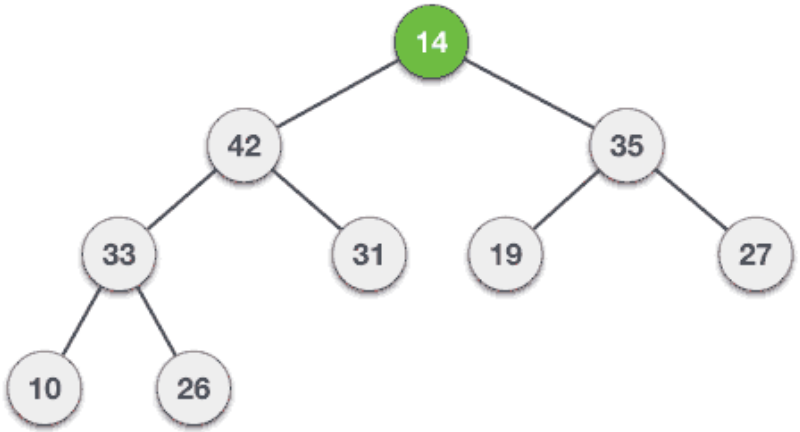
Max Heap – Deletion Example



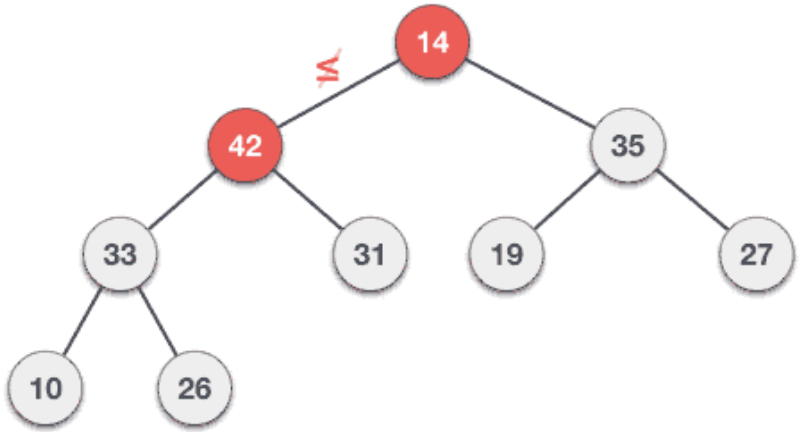
Max Heap – Deletion Example



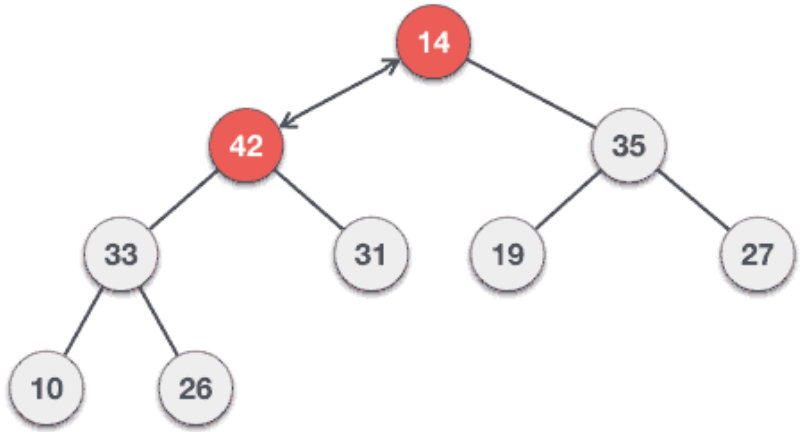
Max Heap – Deletion Example



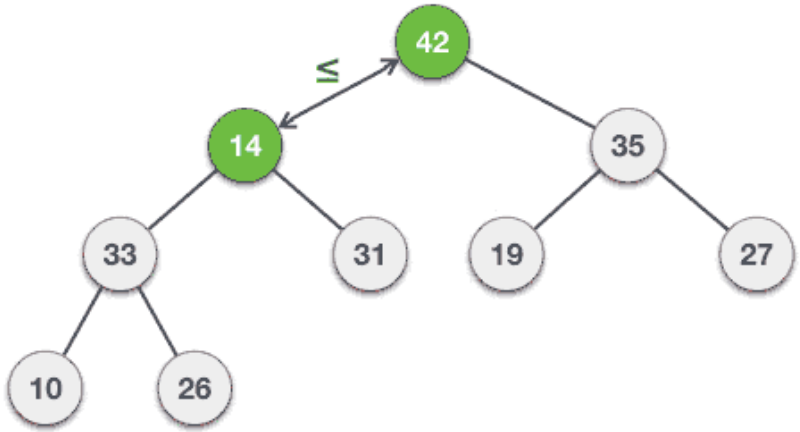
Max Heap – Deletion Example



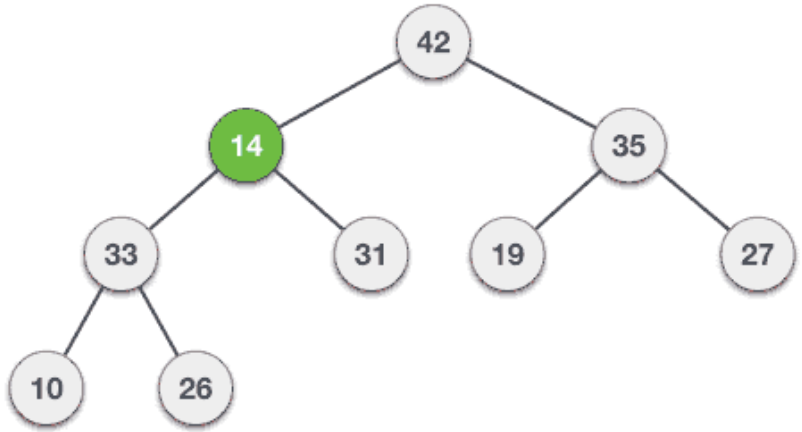
Max Heap – Deletion Example



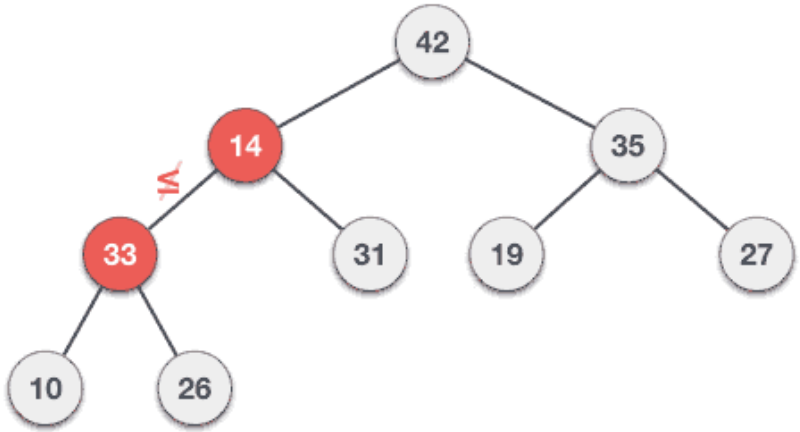
Max Heap – Deletion Example



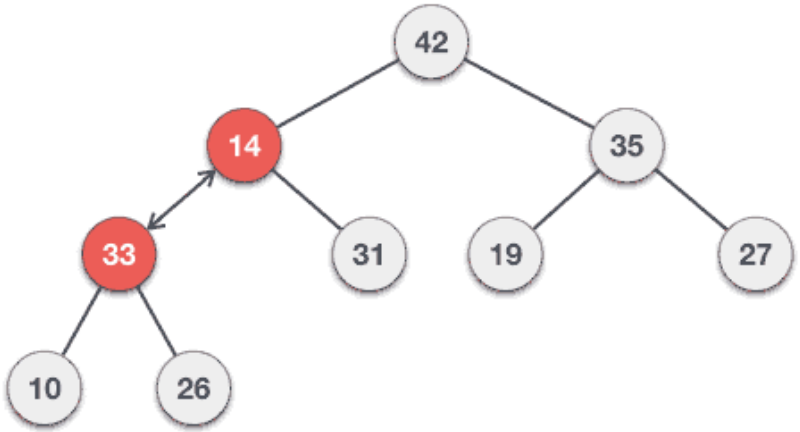
Max Heap – Deletion Example



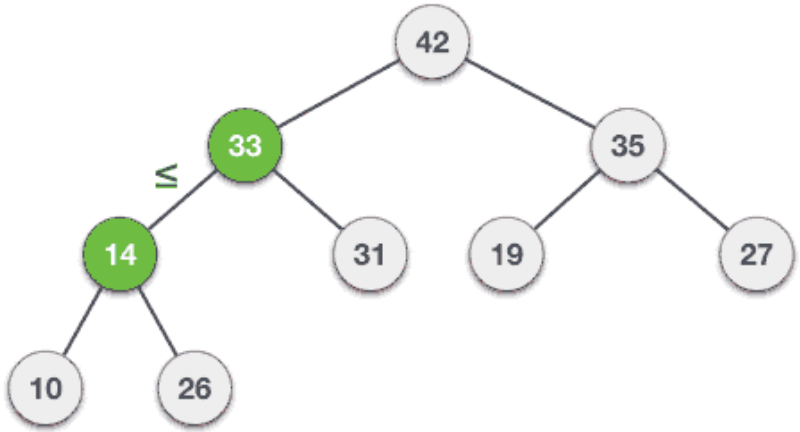
Max Heap – Deletion Example



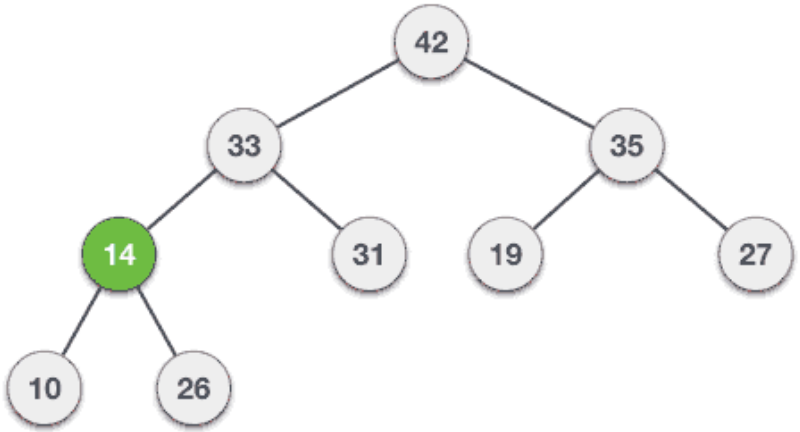
Max Heap – Deletion Example



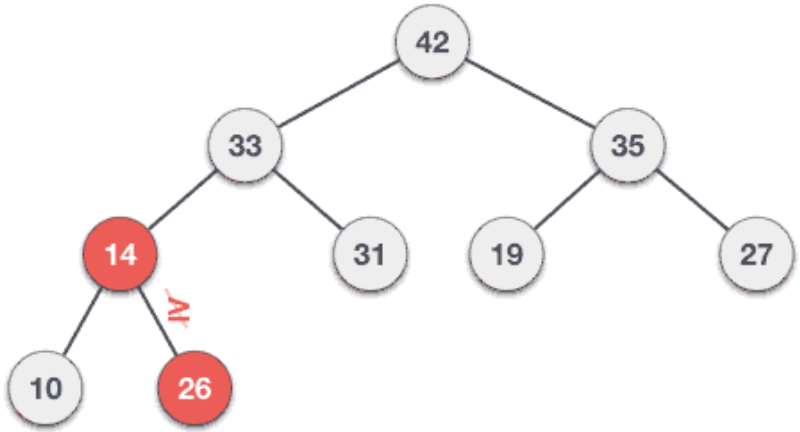
Max Heap – Deletion Example



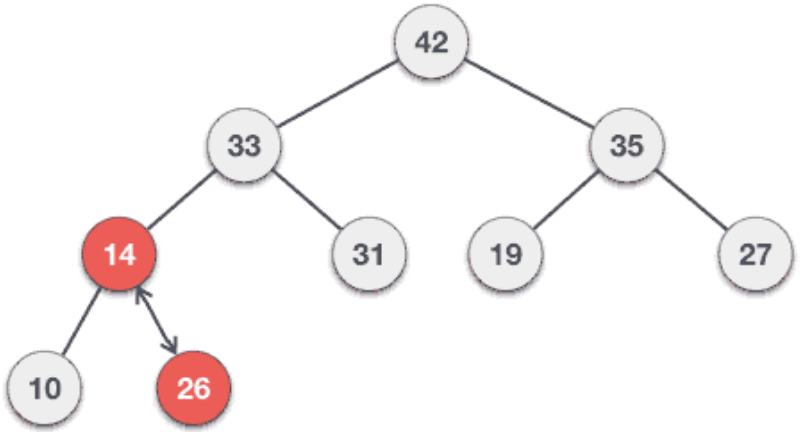
Max Heap – Deletion Example



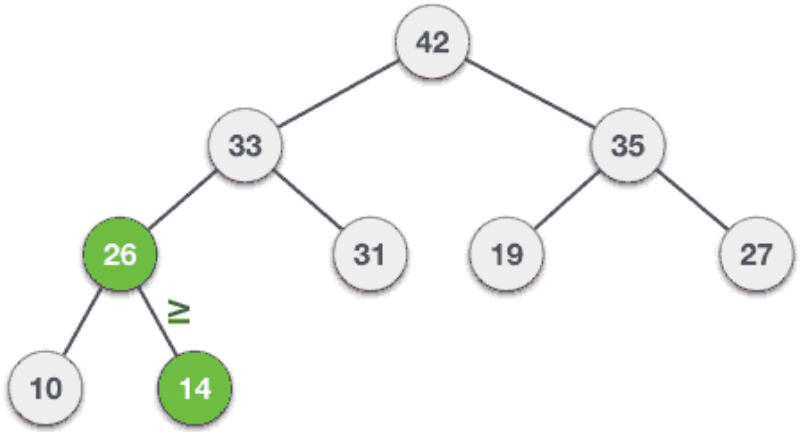
Max Heap – Deletion Example



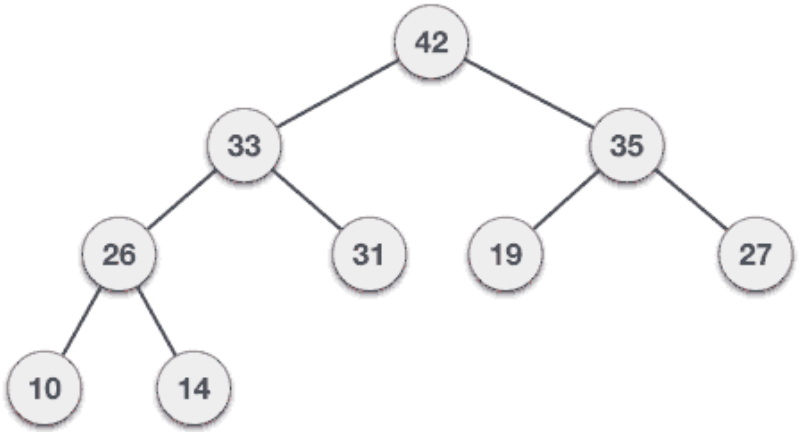
Max Heap – Deletion Example



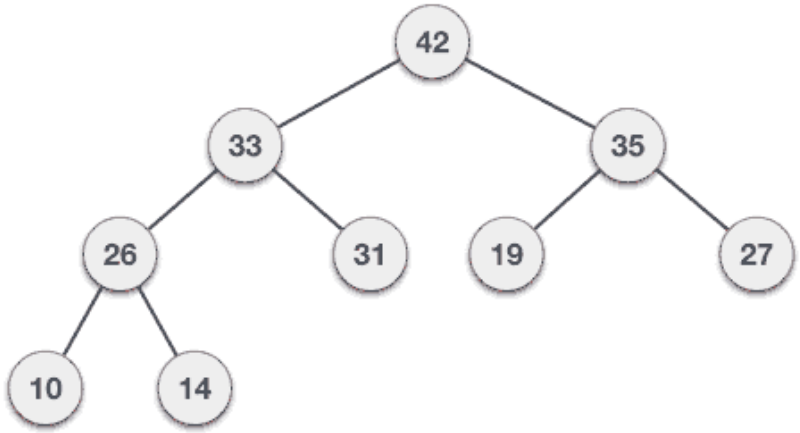
Max Heap – Deletion Example



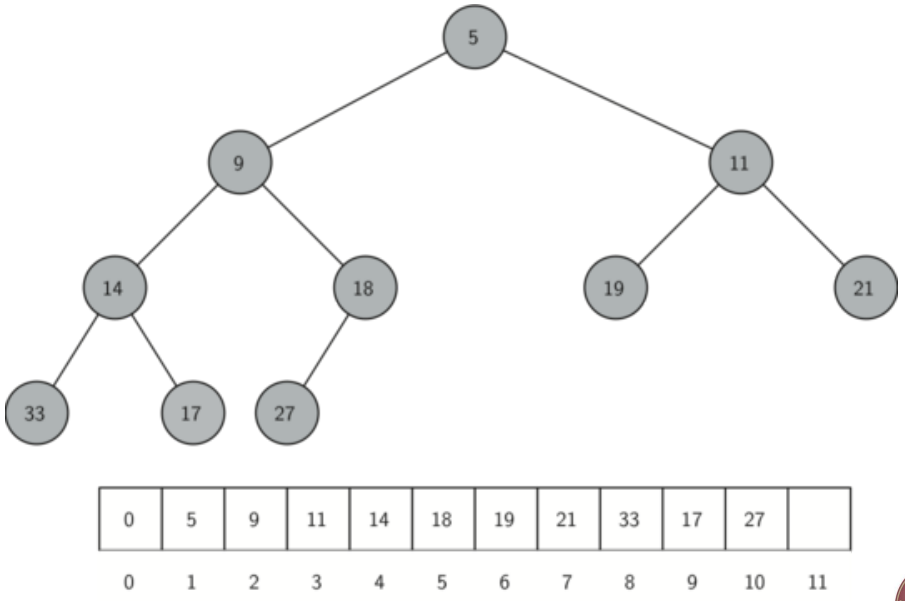
Max Heap – Deletion Example



Max Heap – Deletion Example



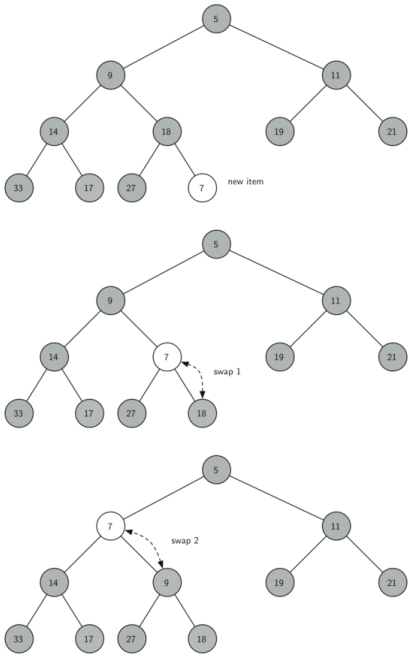
Heap Implementation using Arrays



Binary Heap

```
1 class BinHeap:
2     def __init__(self):
3         self.heapList = [0]
4         self.currentSize = 0
```

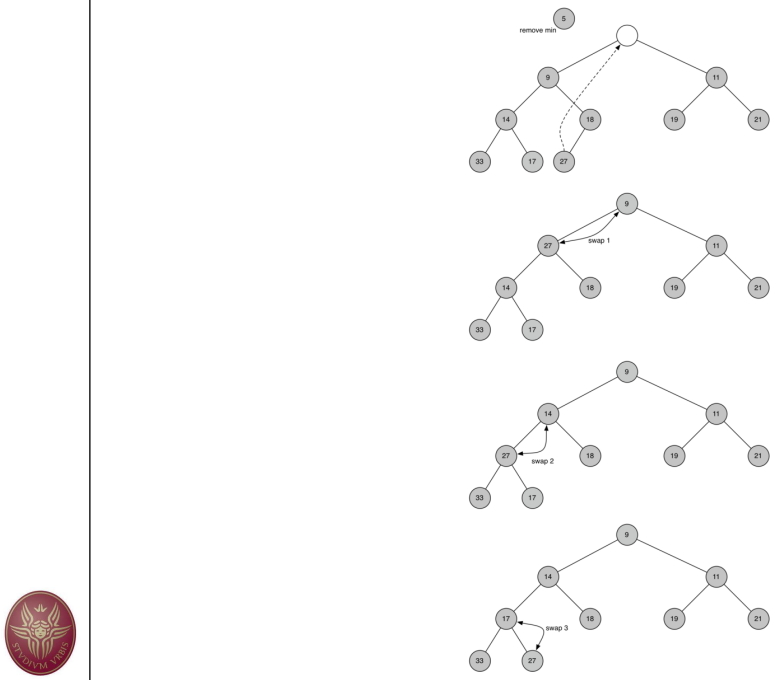
Binary Heap – Insertion



Binary Heap – Insertion

```
def percUp(self,i):
    while i // 2 > 0:
      if self.heapList[i] < self.heapList[i // 2]:
         tmp = self.heapList[i // 2]
         self.heapList[i // 2] = self.heapList[i]
         self.heapList[i] = tmp
      i = i // 2

def insert(self,k):
    self.heapList.append(k)
    self.currentSize = self.currentSize + 1
    self.percUp(self.currentSize)
```



Binary Heap – Remove

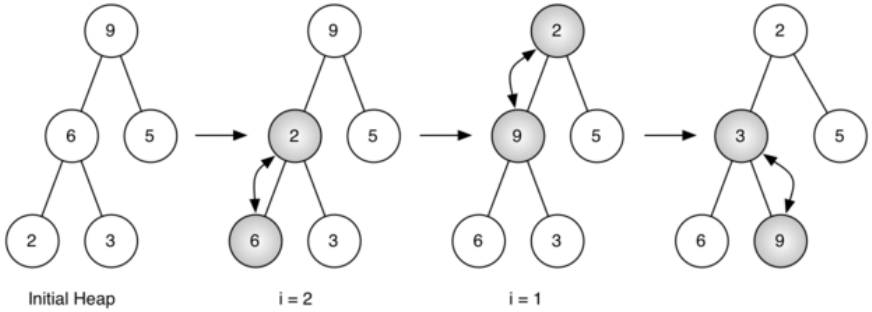
```
def percDown(self,i):
    while (i * 2) <= self.currentSize:
        mc = self.minChild(i)
        if self.heapList[i] > self.heapList[mc]:
            tmp = self.heapList[i]
            self.heapList[i] = self.heapList[mc]
            self.heapList[mc] = tmp
        i = mc

def minChild(self,i):
    if i * 2 + 1 > self.currentSize:
        return i * 2
    else:
        if self.heapList[i*2] < self.heapList[i*2+1]:
            return i * 2
        else:
            return i * 2 + 1
```

Binary Heap – Remove

```
def delMin(self):
    retval = self.heapList[1]
    self.heapList[1] = self.heapList[self.currentSize]
    self.currentSize = self.currentSize - 1
    self.heapList.pop()
    self.percDown(1)
    return retval
```

Binary Heap – Build from List



Binary Heap – Build from List

```
def buildHeap(self,alist):
    i = len(alist) // 2
    self.currentSize = len(alist)
    self.heapList = [0] + alist[:]
    while (i > 0):
        self.percDown(i)
        i = i - 1
```