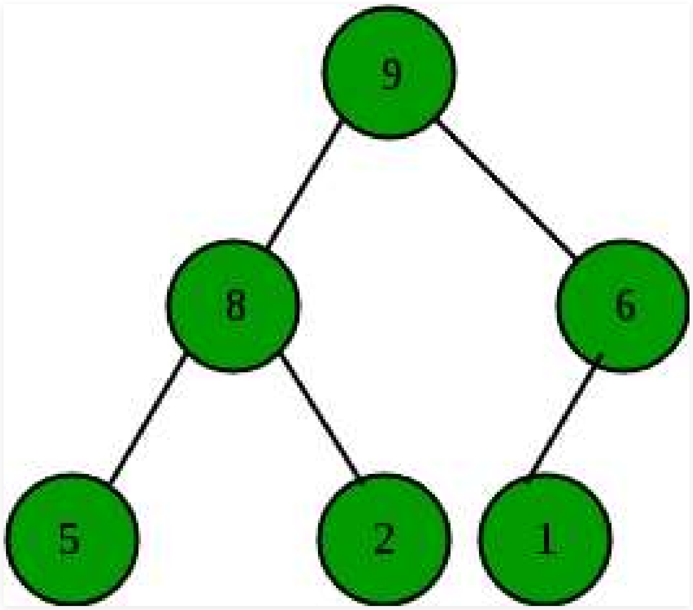
Max Heap in Python

Difficulty Level : Medium ● Last Updated : 16 May, 2022

A Max-Heap is a complete binary tree in which the value in each internal node is greater than or equal to the values in the children of that node.
Mapping the elements of a heap into an array is trivial: if a node is stored a index k, then its left child is stored at index **2k + 1** and its right child at index **2k + 2**.

Examples of Max Heap :



`Arr[(i-1)/2]` Returns the parent node.

`Arr[(2*i)+1]` Returns the left child node.

`Arr[(2*i)+2]` Returns the right child node.

Operations on Max Heap:

1. **getMax()**: It returns the root element of Max Heap. Time Complexity of this operation is **O(1)**.
2. **extractMax()**: Removes the maximum element from MaxHeap. Time Complexity of this Operation is **O(Log n)** as this operation needs to maintain the heap property (by calling heapify()) after removing root.
3. **insert()**: Inserting a new key takes **O(Log n)** time. We add a new key at the end of the tree. If new key is smaller than its parent, then we don't need to do anything. Otherwise, we need to traverse up to fix the violated heap property.

Note : In below implementation, we do indexing from index 1 to simplify the implementation.

```python
# Python3 implementation of Max Heap
import sys

class MaxHeap:

    def __init__(self, maxsize):

        self.maxsize = maxsize
        self.size = 0
        self.Heap = [0] * (self.maxsize + 1)
        self.Heap[0] = sys.maxsize
        self.FRONT = 1

    # Function to return the position of
    # parent for the node currently
    # at pos
    def parent(self, pos):

        return pos // 2
        return 2 * pos

    # Function to return the position of
    # the right child for the node currently
    # at pos
    def rightChild(self, pos):

        return (2 * pos) + 1

    # Function that returns true if the passed
    # node is a leaf node
    def isLeaf(self, pos):

        if pos >= (self.size//2) and pos <= self.size:
            return True
        return False

    # Function to swap two nodes of the heap
    def swap(self, fpos, spos):

        self.Heap[fpos], self.Heap[spos] = (self.Heap[spos],
                                            self.Heap[fpos])

    # Function to heapify the node at pos
    def maxHeapify(self, pos):

        # If the node is a non-leaf node and smaller
        # than any of its child
        if not self.isLeaf(pos):
            if (self.Heap[pos] < self.Heap[self.leftChild(pos)] or
                self.Heap[pos] < self.Heap[self.rightChild(pos)]):

                # Swap with the left child and heapify
                # the left child
                if (self.Heap[self.leftChild(pos)] >
                    self.Heap[self.rightChild(pos)]):
                    self.swap(pos, self.leftChild(pos))
                    self.maxHeapify(self.leftChild(pos))

                # Swap with the right child and heapify
                # the right child
                else:
                    self.swap(pos, self    htChild(pos))
                    self.maxHeapify(sel    ghtChild(pos))
        self.size += 1
        self.Heap[self.size] = element

        current = self.size

        while (self.Heap[current] >
               self.Heap[self.parent(current)]):
            self.swap(current, self.parent(current))
            current = self.parent(current)

    # Function to print the contents of the heap
    def Print(self):

        for i in range(1, (self.size // 2) + 1):
            print(" PARENT : " + str(self.Heap[i]) +
                  " LEFT CHILD : " + str(self.Heap[2 * i]) +
                  " RIGHT CHILD : " + str(self.Heap[2 * i + 1]))

    # Function to remove and return the maximum
    # element from the heap
    def extractMax(self):

        popped = self.Heap[self.FRONT]
        self.Heap[self.FRONT] = self.Heap[self.size]
        self.size -= 1
        self.maxHeapify(self.FRONT)

        return popped

# Driver Code
if __name__ == "__main__":

    print('The maxHeap is ')

    maxHeap = MaxHeap(15)
    maxHeap.insert(5)
    maxHeap.insert(3)
    maxHeap.insert(17)
    maxHeap.insert(10)
    maxHeap.insert(84)
    maxHeap.insert(19)
    maxHeap.insert(6)
    maxHeap.insert(22)
    maxHeap.insert(9)
```

Output :

```
The maxHeap is 
 PARENT : 84 LEFT CHILD : 22 RIGHT CHILD : 19
 PARENT : 22 LEFT CHILD : 17 RIGHT CHILD : 10
 PARENT : 19 LEFT CHILD : 5 RIGHT CHILD : 6
 PARENT : 17 LEFT CHILD : 3 RIGHT CHILD : 9
The Max val is 84
```

Using Library functions :

We use heapq class to implement Heaps in Python. By default Min Heap is implemented by this class. But we multiply each value by -1 so that we can use it as MaxHeap.

Array Matrix Strings Hashing Linked List Stack Queue Binary Tree Binary Search Tre

```python
heap = []
heapify(heap)

# Adding items to the heap using heappush
# function by multiplying them with -1
heappush(heap, -1 * 10)
heappush(heap, -1 * 30)
heappush(heap, -1 * 20)
heappush(heap, -1 * 400)

# printing the value of maximum element
print("Head value of heap : "+str(-1 * heap[0]))

# printing the elements of the heap
print("The heap elements : ")
for i in heap:
    print(-1 * i, end = ' ')
rint("\n")

element = heappop(heap)
```

Output :

```
Head value of heap : 400
The heap elements :
400 30 20 10

The heap elements :
30 10 20
```

Like 20

Previous

Next

RECOMMENDED ARTICLES

Page : 1 2 3

02 08, Jun 16

03 Difference between Min Heap and Max Heap
06, Nov 20

04 Find min and max values among all maximum leaf nodes from all possible Binary Max Heap
23, Jun 21

06 Convert BST to Max Heap
08, Jun 18

07 Minimum element in a max heap
13, Oct 18

08 Max Heap in Java
24, Nov 18

Article Contributed By :

chaudhary_19
@chaudhary_19

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : ignacio hugo gomez

Article Tags : Python-Data-Structures, Technical Scripter 2019, Heap, Python, Technical Scripter

Practice Tags : Heap

Start Your Coding Journey Now!

Login

Register

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

- About Us
- Careers
- In Media
- Contact Us
- Privacy Policy
- Copyright Policy

Learn

- Algorithms
- Data Structures
- SDE Cheat Sheet
- Machine learning
- CS Subjects
- Video Tutorials

News

- Top News
- Technology
- Work & Career
- Business
- Finance
- Lifestyle

Languages

- Python
- Java
- CPP
- Golang
- C#
- SQL

Django Tutorial

Improve an Article

HTML

Pick Topics to Write

CSS

Write Interview Experience

JavaScript

Bootstrap

Internships

Video Internship