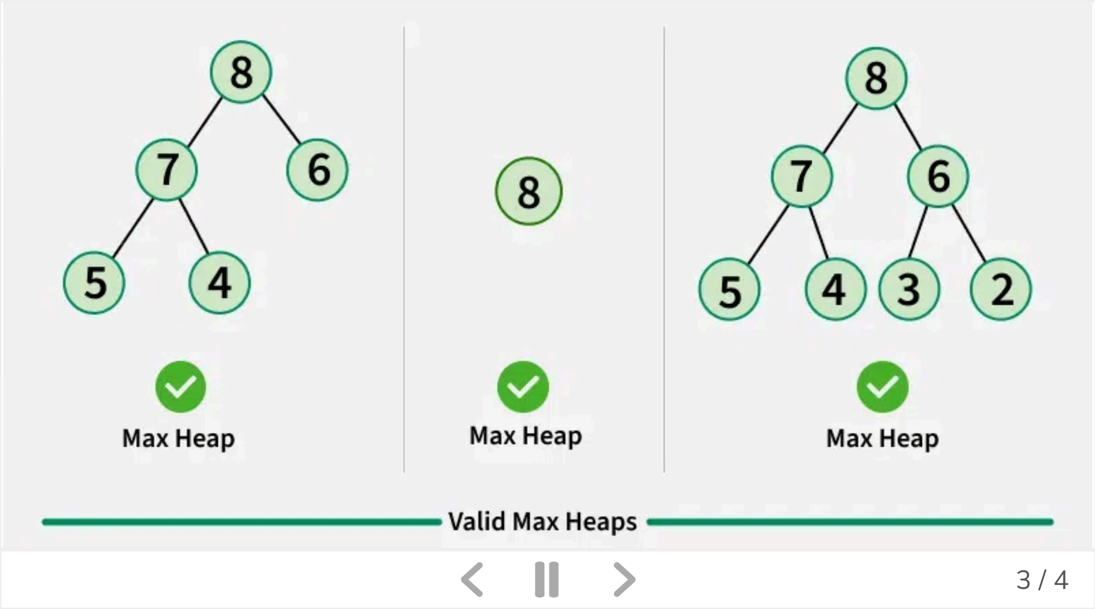
# Binary Heap

Last Updated : 04 Nov, 2024

*efficiently to get the max or min element based on its structure.*

A Binary Heap is either Min Heap or Max Heap. In a Min Binary Heap, the key at the root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in Binary Tree. Max Binary Heap is similar to MinHeap.

### Valid and Invalid examples of heaps



## How is Binary Heap represented?

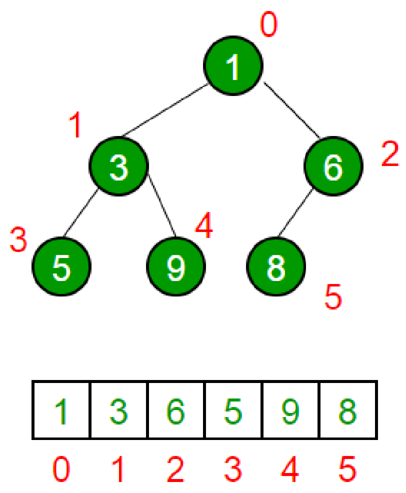
A Binary Heap is a **Complete Binary Tree**. A binary heap is typically represented as an array.

- The root element will be at Arr[0].

- The below table shows indices of other nodes for the $i^{th}$ node, i.e., Arr[i]:

| | |
|---|---|
| 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 |

The traversal method use to achieve Array representation is **Level Order Traversal**. Please refer to Array Representation Of Binary Heap for details.



## Operations on Heap:

Below are some standard operations on min heap:

- **getMin():** It returns the root element of Min Heap. The time Complexity of this operation is **O(1)**. In case of a maxheap it would be **getMax()**.
- **extractMin():** Removes the minimum element from MinHeap. The time Complexity of this Operation is **O(log N)** as this operation needs to maintain the heap property (by calling **heapify()**) after removing the root.
- **decreaseKey():** Decreases the value of the key. The time complexity of this operation is **O(log N)**. If the decreased key value of a node is greater than the parent of the node, then we don't need to do anything. Otherwise, we need to traverse up to fix the violated heap property.
- **insert():** Inserting a new key takes **O(log N)** time. We add a new key at the end of the tree. If the new key is greater than its parent, then we don't need

to do anything. Otherwise, we need to traverse up to fix the violated heap property.

- **delete():** Deleting a key also takes **O(log N)** time. We replace the key to be deleted with the minimum infinite by calling **decreaseKey()**. After decreaseKey(), the minus infinite value must reach root, so we call **extractMin()** to remove the key.

Below is the implementation of basic heap operations.

C++ Java Python C# JavaScript

```python
# A Python program to demonstrate common binary heap operations

# Import the heap functions from python library
from heapq import heappush, heappop, heapify

# heappop - pop and return the smallest element from heap
# heappush - push the value item onto the heap, maintaining
#              heap invarient
# heapify - transform list into heap, in place, in linear time

# A class for Min Heap
class MinHeap:

    # Constructor to initialize a heap
    def __init__(self):
        self.heap = []

    def parent(self, i):
        return (i-1)/2

    # Inserts a new key 'k'
    def insertKey(self, k):
        heappush(self.heap, k)

    # Decrease value of key at index 'i' to new_val
    # It is assumed that new_val is smaller than heap[i]
    def decreaseKey(self, i, new_val):
        self.heap[i]  = new_val
        while(i != 0 and self.heap[self.parent(i)] > self.heap[i]):
            # Swap heap[i] with heap[parent(i)]
            self.heap[i] , self.heap[self.parent(i)] = (
            self.heap[self.parent(i)], self.heap[i])

    # Method to remove minimum element from min heap
    def extractMin(self):
        return heappop(self.heap)

    # This function deletes key at index i. It first reduces
    # value to minus infinite and then calls extractMin()
    def deleteKey(self, i):
        self.decreaseKey(i, float("-inf"))
        self.extractMin()

    # Get the minimum element from the heap
    def getMin(self):
        return self.heap[0]

# Driver pgoratm to test above function
heapObj = MinHeap()
heapObj.insertKey(3)
heapObj.insertKey(2)
heapObj.deleteKey(1)
heapObj.insertKey(15)
heapObj.insertKey(5)
heapObj.insertKey(4)
heapObj.insertKey(45)

print heapObj.extractMin(),
print heapObj.getMin(),
heapObj.decreaseKey(2, 1)
print heapObj.getMin()

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

## Output

```
2 4 1
```

## Applications of Heaps:

- Heap Sort: Heap Sort uses Binary Heap to sort an array in O(nLogn) time.
- Priority Queue: Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(log N) time. Binomial Heap and Fibonacci Heap are variations of Binary Heap. These variations perform union also efficiently.
- Graph Algorithms: The priority queues are especially used in Graph Algorithms like Dijkstra's Shortest Path and Prim's Minimum Spanning Tree.
- Many problems can be efficiently solved using Heaps. See following for example. a) K'th Largest Element in an array. b) Sort an almost sorted array/ c) Merge K Sorted Arrays.

**Related Links:**

- Coding Practice on Heap
- All Articles on Heap
- Quiz on Heap
- PriorityQueue : Binary Heap Implementation in Java Library

GeeksforGeeks 386

**Previous Article**

Introduction to Heap - Data Structure and Algorithm Tutorials

**Next Article**

Advantages and Disadvantages of Heap

## Similar Reads

### Difference between Binary Heap, Binomial Heap and Fibonacci Heap

Binary Heap:A Binary Heap is a Binary Tree with following properties. It’s a complete binary tree i.e., all levels are completely filled except possibly the last...

2 min read

### Convert Min Heap to Max Heap

Given an array representation of min Heap, convert it to max Heap. Examples: Input: arr[] = {3, 5, 9, 6, 8, 20, 10, 12, 18, 9} 3 / \ 5 9 / \ / \ 6 8 20 10 / \ /12 18 9...

10 min read

### Heap Sort for decreasing order using min heap

Given an array of elements, sort the array in decreasing order using min heap. Examples: Input : arr[] = {5, 3, 10, 1} Output : arr[] = {10, 5, 3, 1} Input : arr[] = {1...

13 min read

### When building a Heap, is the structure of Heap unique?

What is Heap? A heap is a tree based data structure where the tree is a complete binary tree that maintains the property that either the children of a node are les...

4 min read

### Difference between Min Heap and Max Heap

A Heap is a special Tree-based data structure in which the tree is a complete binary tree. Since a heap is a complete binary tree, a heap with N nodes has log ...

3 min read

### What's the relationship between "a" heap and "the" heap?

A Heap: "A Heap" refers to the heap data structure where we can store data in a specific order. Heap is a Tree-based data structure where the tree is a complete...

15+ min read

### Complexity analysis of various operations of Binary Min Heap

A Min Heap is a Complete Binary Tree in which the children nodes have a higher value (lesser priority) than the parent nodes, i.e., any path from the root to the le...

3 min read

### Print all the leaf nodes of Binary Heap

Given an array of N elements which denotes the array representation of binary heap, the task is to find the leaf nodes of this binary heap. Examples: Input: arr[] ...

7 min read

### Find min and max values among all maximum leaf nodes from all possible…

Given a positive integer N, the task is to find the largest and smallest elements, from the maximum leaf nodes of every possible binary max-heap formed by...

7 min read

### Check if Binary Heap is completely filled

Given an integer N which denotes the number of elements in a binary heap. Check if the binary Heap is completely filled or not. Examples: Input: 7Output:...

4 min read

Article Tags : DSA Heap

Practice Tags : Heap

Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305

## Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

## Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

## DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

## Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

## Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

## Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

## Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

## DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

## System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns

## Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation

OOAD

System Design Bootcamp

Interview Questions

### School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

World GK

Puzzles

### GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects