Min Heap in Python - GeeksforGeeks

Source: https://www.geeksforgeeks.org/min-heap-in-python/

Courses Tutorials Practice Jobs Python Tutorial Data Types Interview Questions Examples Quizzes DSA Python Data Science NumPy Pandas Practice Django Flask Technical Scripter 2026 Explore Python Fundamentals Python Introduction Input and Output in Python Python Variables Python Operators Python Keywords Python Data Types Conditional Statements in Python Loops in Python - For, While and Nested Loops Python Functions Recursion in Python Python Lambda Functions Python Data Structures Python String Python Lists Python Tuples Python Dictionary Python Sets Python Arrays List Comprehension in Python Advanced Python Python OOP Concepts Python Exception Handling File Handling in Python Python Database Tutorial Python MongoDB Tutorial Python MySQL Python Packages Python Modules Python DSA Libraries List of Python GUI Library and Packages Data Science with Python NumPy Tutorial - Python Library Pandas Tutorial Matplotlib Tutorial Python Seaborn Tutorial StatsModel Library - Tutorial Learning Model Building in Scikit-learn TensorFlow Tutorial PyTorch Tutorial Web Development with Python Flask Tutorial Django Tutorial | Learn Django Framework Django ORM - Inserting, Updating & Deleting Data Templating With Jinja2 in Flask Django Templates Build a REST API using Flask - Python Building a Simple API with Django REST Framework Python Practice Python Quiz Python Coding Practice Python Interview Questions and Answers Data Science Course 90% Refund Min Heap in Python Last Updated : 12 Jul, 2025 A Min-Heap is a Data Structure with the following properties. It is a complete Complete Binary Tree . The value of the root node must be the smallest among all its descendant nodes and the same thing must be done for its left and right sub-tree also. Table of Content Example of Min Heap Min Heap Representation as an Array Operations on Min Heap Implementation of Min Heap in Python Implementation of Min Heap Using Python’s heapq Library Implementation of Min Heap using Priority Queue The elements of a heap can be mapped into an array using the following rules: If a node is stored at index k : Left child is stored at index 2k + 1 (for 0-based indexing) or 2k (for 1-based indexing). Right child is stored at index 2k + 2 (for 0-based indexing) or 2k + 1 (for 1-based indexing). Example of Min Heap Tree Representation : 5 13 / \ / \ 10 15 16 31 / / \ / \ 30 41 51 100 41 Array Representation : For the first tree: [5, 10, 15, 30] For the second tree: [13, 16, 31, 41, 51, 100, 41] Min Heap Representation as an Array Since a Min Heap is a Complete Binary Tree, it is commonly represented using an array. In an array representation: The root element is stored at Arr[0] . For any i-th node (at Arr[i]): Parent Node → Arr[(i - 1) / 2] Left Child → Arr[(2 * i) + 1] Right Child → Arr[(2 * i) + 2] Operations on Min Heap getMin() : It returns the root element of Min Heap. Time Complexity of this operation is O(1) . extractMin() : Removes the minimum element from MinHeap. Time Complexity of this Operation is O(Log n) as this operation needs to maintain the heap property (by calling heapify()) after removing root. 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 larger than its parent, then we don’t need to do anything. Otherwise, we need to traverse up to fix the violated heap property. Implementation of Min Heap in Python Please refer Min-Heap for details. Python class MinHeap : def __init__ (self): self . a = [] """Insert a new element into the Min Heap.""" def insert (self , val): self . a . append (val) i = len (self . a) - 1 while i > 0 and self . a [(i - 1) // 2] > self . a [i]: self . a [i], self . a [(i - 1) // 2] = self . a [(i - 1) // 2], self . a [i] i = (i - 1) // 2 """Delete a specific element from the Min Heap.""" def delete (self , value): i = - 1 for j in range (len (self . a)): if self . a [j] == value : i = j break if i == - 1 : return self . a [i] = self . a [- 1] self . a . pop () while True : left = 2 * i + 1 right = 2 * i + 2 smallest = i if left < len (self . a) and self . a [left] < self . a [smallest]: smallest = left if right < len (self . a) and self . a [right] < self . a [smallest]: smallest = right if smallest != i : self . a [i], self . a [smallest] = self . a [smallest], self . a [i] i = smallest else : break """Heapify function to maintain the heap property.""" def minHeapify (self , i , n): smallest = i left = 2 * i + 1 right = 2 * i + 2 if left < n and self . a [left] < self . a [smallest]: smallest = left if right < n and self . a [right] < self . a [smallest]: smallest = right if smallest != i : self . a [i], self . a [smallest] = self . a [smallest], self . a [i] self . minHeapify (smallest , n) """Search for an element in the Min Heap.""" def search (self , element): for j in self . a : if j == element : return True return False def getMin (self): return self . a [0] if self . a else None def printHeap (self): print ("Min Heap:" , self . a) # Example Usage if __name__ == "__main__" : h = MinHeap () values = [10 , 7 , 11 , 5 , 4 , 13] for value in values : h . insert (value) h . printHeap () h . delete (7) print ("Heap after deleting 7:" , h . a) print ("Searching for 10 in heap:" , "Found" if h .

search (10) else "Not Found") print ("Minimum element in heap:" , h . getMin ()) Output Min Heap: [4, 5, 11, 10, 7, 13] Heap after deleting 7: [4, 5, 11, 10, 13] Searching for 10 in heap: Found Minimum element in heap: 4 Implementation of Min Heap Using Python's heapq Library Python's heapq module implements a Min Heap by default. Python # Python3 program to demonstrate working of heapq from heapq import heapify , heappush , heappop # Creating empty heap heap = [] heapify (heap) # Adding items to the heap using heappush function heappush (heap , 10) heappush (heap , 30) heappush (heap , 20) heappush (heap , 400) # printing the value of minimum element print ("Head value of heap : " + str (heap [0])) # printing the elements of the heap print ("The heap elements : ") for i in heap : print (i , end = ' ') print (" \n ") element = heappop (heap) # printing the elements of the heap print ("The heap elements : ") for i in heap : print (i , end = ' ') Output Head value of heap : 10 The heap elements : 10 30 20 400

The heap elements : 20 30 400 Implementation of Min Heap using queue.PriorityQueue Please refer queue.PriorityQueue for details. Python from queue import PriorityQueue q = PriorityQueue () # insert into queue q . put (10) q . put (20) q . put (5) # remove and return # lowest priority item print (q . get ()) print (q . get ()) # check queue size print ('Items in queue :' , q . qsize ()) # check if queue is empty print ('Is queue empty :' , q . empty ()) # check if queue is full print ('Is queue full :' , q . full ()) Output 5 10 Items in queue : 1 Is queue empty : False Is queue full : False Comment Article Tags: Article Tags: Python Technical Scripter 2019 Python-DSA