Implementing Heaps using inbuilt classes in C++ and Java

Heaps in C++

In C++ the priority_queue container can be used to implement Heaps. The priority_queue in C++ is a built-in container class which is used to implement priority queues easily. This container uses max heap internally to implement a priority queue efficiently. Therefore, it can be also used in-place of max heaps.

This container can also be modified by passing some additional parameters to be used as a Min Heap.

Syntax:

- For implementing Max Heap:

```
priority_queue< type_of_data > name_of_heap;
```

- For implementing Min Heap:

```
priority_queue< type, vector<type>, greater<type> > name_of_heap;
```

Methods of priority queue:

- **priority_queue::empty()**: The empty() function returns whether the queue is empty.

- **priority_queue::size()**: The size() function returns the size of the queue.
- **priority_queue::top()**: This function returns a reference to the top most element of the queue.
- **priority_queue::push()**: The push(g) function adds the element 'g' at the end of the queue.
- **priority_queue::pop()**: The pop() function deletes the first element of the queue.
- **priority_queue::swap()**: This function is used to swap the contents of one priority queue with another priority queue of same type and size.

Note: The above functions are applicable in case of both Max-Heap and Min-heap implementation of priority_queue container.

Implementation:

CPP

```
// C++ program to implement Max Heap and Min Heap
// using priority_queue in C++ STL

#include <iostream>
#include <queue>

using namespace std;

int main ()
{
    // DECLARING MAX HEAP
    priority_queue <int> max_heap;

    // Add elements to the Max Heap
    // in any order
    max_heap.push(10);
    max_heap.push(30);
```

```
max_heap.push(20);
max_heap.push(5);
max_heap.push(1);

// Print element at top of the heap
// every time and remove it until the
// heap is not empty
cout<<"Element at top of Max Heap at every step:\n";
while(!max_heap.empty())
{
    // Print Top Element
    cout<<max_heap.top()<<" ";

    // Remove Top Element
    max_heap.pop();
}

// DECLARING MIN HEAP
priority_queue <int, vector<int>, greater<int> > min_heap;

// Add elements to the Min Heap
// in any order
min_heap.push(10);
min_heap.push(30);
min_heap.push(20);
min_heap.push(5);
min_heap.push(1);

// Print element at top of the heap
// every time and remove it until the
// heap is not empty
cout<<"\n\nElement at top of Min Heap at every step:\n";
while(!min_heap.empty())
{
    // Print Top Element
    cout<<min_heap.top()<<" ";

    // Remove Top Element
    min_heap.pop();
}
```

```
        return 0;
    }
```


Output:

```
Element at top of Max Heap at every step:
30 20 10 5 1

Element at top of Min Heap at every step:
1 5 10 20 30
```

Heaps in Java

Java also provides us with a built-in class named PriorityQueue which can be used to implement both Max heap and Min heap easily and efficiently.

By default, the PriorityQueue class implements a Min-Heap. However, it can also be modified by using a comparator function to implement Max-Heap as well as shown in the below syntax.

Syntax:

- Implementing Max Heap:

```
Queue<Integer> max_heap = new PriorityQueue<>(
                                    Collections.reverseOrder());
```

- Implementing Min Heap:

```
Queue<Integer> min_heap = new PriorityQueue<>();
```

Dash
All
Articles
Videos
Problems
Quiz

Some useful method of PriorityQueue class in Java:

- **boolean add(E element)**: This method inserts the specified element into this priority queue.
- **public remove()**: This method removes a single instance of the specified element from this queue, if it is present.
- **public poll()**: This method retrieves and removes the head of this queue, or returns null if this queue is empty.
- **public peek()**: This method retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
- **void clear()**: This method is used to remove all of the contents of the priority queue.
- **int size()**: The method is used to return the number of elements present in the set.

Note: The above methods are applicable in case of both Max-Heap and Min-heap implementation of PriorityQueue class.

Implementation:

Java

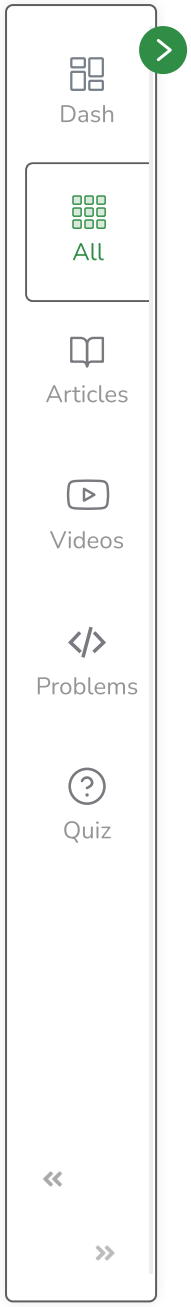
```java
// Java Program to implement Max heap and Min heap
// using the PriorityQueue class

import java.util.*;
import java.io.*;
import java.lang.*;

class GFG
{
    public static void main (String[] args) {

        // DECLARING MAX HEAP
        Queue<Integer> max_heap = new PriorityQueue<>(
                                  Collections.reverseOrder());

        // Add elements to the Max Heap
        // in any order
        max_heap.add(10);
        max_heap.add(30);
        max_heap.add(20);
        max_heap.add(5);
         max_heap.add(1);

         // Print element at top of the heap
        // every time and remove it until the
        // heap is not empty
        System.out.print("Element at top of Max Heap at"
                                        + " every step:\n");
        while(!max_heap.isEmpty())
        {
            // Print Top Element
            System.out.print(max_heap.peek()+" ");

        }

        // DECLARING MIN HEAP
        Queue<Integer> min_heap = new PriorityQueue<>();
```

```
            // Add elements to the Min Heap
            // in any order
            min_heap.add(10);
            min_heap.add(30);
            min_heap.add(20);
            min_heap.add(5);
            min_heap.add(1);

            // Print element at top of the heap
            // every time and remove it until the
            // heap is not empty
            System.out.print("\n\nElement at top of Min Heap "
                                        + "at every step:\n");
            while(!min_heap.isEmpty())
            {
               // Print Top Element
               System.out.print(min_heap.peek()+" ");

               // Remove Top Element
               min_heap.poll();
            }
        }
    }
```

Output

```
Element at top of Max Heap at every step:
30 20 10 5 1

Element at top of Min Heap at every step:
1 5 10 20 30
```

Mark as Read



Report An Issue

If you are facing any issue on this page. Please let us know.