Dash
All
Articles
Videos
Problems
Quiz
Contest
Prev
Next

Design a Data Structure with Min and Max operations

Problem: Design a Data Structure a SpecialQueue which supports following operations enqueue, deque, getMin() or getMax() where getMin() operation takes O(1) time.
Example:

```
Let the data to be inserted in queue be -
4, 2, 1, 6

Operation     Queue       Output
push(4)         4           -
push(2)        4, 2         -
push(1)       4, 2, 1       -
getMin()      4, 2, 1       1
push(6)     4, 2, 1, 6      -
pop()         2, 1, 6       4
pop()          1, 6         2
pop()            6          1
getMin()         6          6

// Notice the getMin() function call
// It returns the minimum element
// of all the values present in the queue
```

Approach: The idea is to use Doubly ended Queue to store in increasing c minimum element and store in decreasing order if the structure is to return the the Data Structure is defined as follows:

Enque

- Insert the element into the queue structure.
- If the size of the Deque structure is empty that is the size of the Deque is 0. Then, Insert the element from the back.
- Otherwise, If there are some elements in the Deque structure then pop the elements out from the Deque until the back of the Deque is greater than the current element and then finally insert the element from back.

Deque

- If the first element of the Deque is equal to the front element of the queue then pop the elements out from the Queue and the Deque at the same time.
- Otherwise, Pop the element from the front of the queue to maintain the order of the elements.

Get Minimum

- Return the front element of the Deque to get the minimum element of the current element of the queue.

Below is the implementation of the above approach:

Java C++

```
import java.util.*;
import java.io.*;

class SpecialQueue {

    Queue<Integer> q;
    Deque<Integer> dq;

    public SpecialQueue(){
        q = new LinkedList<>();
        dq = new ArrayDeque<>();
    }

    void enque(int data){
          // remove all elements from
          // from deque which are greater
        // than the current element 'data'
        while(!dq.isEmpty() && dq.getLast() > data){
            dq.removeLast();
        }
          // If queue is empty then
          // while loop is skipped.
        dq.addLast(data);
        q.add(data);
    }

    void deque(){
```

```
            // If min element is present
            // at front of queue
        if(dq.getFirst() == q.peek()){
            dq.removeFirst();
        }
        q.remove();
    }

    int getMin() throws Exception{
            // If queue is Empty, return Exception
        if(q.isEmpty())
            throw new Exception("Queue is Empty");
        else
            return dq.getFirst();
    }
      public static void main(String[] args) throws Exception {
        SpecialQueue arr = new SpecialQueue();
        arr.enque(1);
        arr.enque(2);
          arr.enque(4);
        System.out.println(arr.getMin());
        arr.deque();
        System.out.println(arr.getMin());
    }
}
```

Output:

```
1
2
```

Article marked as read.

Dash
All
Articles
Videos
Problems
Quiz
Contest

Time Complexity: O(n)

Auxiliary Space: O(n)

Marked as Read

Report An Issue

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