Design a stack which can give maximum frequency element

Difficulty Level : Expert • Last Updated : 06 Mar, 2022

Given **N** elements and the task is to implement a stack which removes and returns the maximum frequency element on every pop operation. If there's a tie in the frequency then the topmost highest frequency element will be returned.

Examples:

Input:
push(4) 8
push(6) 6
push(7) 7
push(6) 6
push(8); 4
Output:
pop() -> returns 6, as 6 is the most frequent (frequency of 6 = 2).
pop() -> returns 8 (6 also has the highest frequency but it is not the topmost)

Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.

Approach: Maintaining two HashMap, one is frequency HashMap which maps elements to their frequencies and other is setMap which maps all the element with same frequency in one group (Stack).
FrequencyStack has 2 functions:

1. **push(int x):** map the element (x) with frequency HashMap and update the maxfreq variable (i.e. holds the maximum frequency till now). **setMap** maintains a stack which contains all the elements with same frequency.
2. **pop():** First get the maxfreq element from setMap and then decrement the frequency of the popped element. After popping, if the stack becomes empty then decrement the maxfreq.

Below is the implementation of the above approach:

C++

```
// C++ implementation of the approach
#include<bits/stdc++.h>
using namespace std;

// freqMap is to map element to its frequency
map<int, int> freqMap;
```

```
    // of the stack element
    int maxfreq = 0;

    // Function to insert x in the stack
    void push(int x)
    {

        // Frequency of x
        int freq = freqMap[x] + 1;

        // Mapping of x with its frequency
        freqMap[x]= freq;

        // Update maxfreq variable
        if (freq > maxfreq)
            maxfreq = freq;

        // Map element to its frequency list
        // If given frequency list doesn't exist
        // make a new list of the required frequency
        setMap[freq].push(x);
    }

    // Function to remove maximum frequency element
    int pop()
    {

        // Remove element from setMap
        // from maximum frequency list
        int top = setMap[maxfreq].top();
        setMap[maxfreq].pop();

        // Decrement the frequency of the popped element
        freqMap[top]--;

        // If whole list is popped
        // then decrement the maxfreq
        if (setMap[maxfreq].size() == 0)
            maxfreq--;
        return top;
    }
        // Push elements to the stack
        push(4);
        push(6);
        push(7);
        push(6);
        push(8);

        // Pop elements
        cout << (pop()) << "\n" ;
        cout << (pop());
        return 0;
    }

    // This code is contributed by Arnab Kundu
```

Java

```
// Java implementation of the approach
import java.util.*;

public class freqStack {

    // freqMap is to map element to its frequency
    static Map<Integer, Integer> freqMap = new HashMap<>();

    // setMap is to map frequency to the
    // element list with this frequency
    static Map<Integer, Stack<Integer> > setMap = new HashMap<>();

    // Variable which stores maximum frequency
    // of the stack element
    static int maxfreq = 0;

    // Function to insert x in the stack
    public static void push(int x)
    {

        // Frequency of x
        int freq = freqMap.getOrDefault(x, 0) + 1;
        // Update maxfreq variable
        if (freq > maxfreq)
            maxfreq = freq;

        // Map element to its frequency list
        // If given frequency list doesn't exist
        // make a new list of the required frequency
        setMap.computeIfAbsent(freq, z -> new Stack()).push(x);
    }

    // Function to remove maximum frequency element
    public static int pop()
    {

        // Remove element from setMap
        // from maximum frequency list
        int top = setMap.get(maxfreq).pop();

        // Decrement the frequency of the popped element
        freqMap.put(top, freqMap.get(top) - 1);

        // If whole list is popped
        // then decrement the maxfreq
        if (setMap.get(maxfreq).size() == 0)
            maxfreq--;
        return top;
    }

    // Driver code
    public static void main(String[] args)
    {

        // Push elements to the stack
        push(4);
        push(6);
        push(7);
        push(6);
        push(8);

        // Pop elements
        System.out.println(pop());
```

Python3

```
# Python3 implementation of the approach

# freqMap is to map element to its frequency
freqMap = {};

# setMap is to map frequency to the
# element list with this frequency
setMap = {};

# Variable which stores maximum frequency
# of the stack element
maxfreq = 0;

# Function to insert x in the stack
def push(x) :
    global maxfreq;
    if x not in freqMap :
        freqMap[x] = 0

    # Frequency of x
    freq = freqMap[x] + 1;

    # Mapping of x with its Frequency
    freqMap[x]= freq

    # Update maxfreq Variable
    if (freq > maxfreq) :
        maxfreq = freq

    # Map element to its frequency list
    # If given frequency list doesn't exist
    # make a new list of the required frequency
    if freq not in setMap :
        setMap[freq] = []

    setMap[freq].append(x);
    # Remove element from setMap
    # from maximum frequency list
    top = setMap[maxfreq][-1];
    setMap[maxfreq].pop();

    # Decrement the frequency
    # of the popped element
    freqMap[top] -= 1;

    # If whole list is popped
    # then decrement the maxfreq
    if (len(setMap[maxfreq]) == 0) :
        maxfreq -= 1;

    return top;

# Driver code
if __name__ == "__main__" :

    # Push elements to the stack
    push(4);
    push(6);
    push(7);
    push(6);
    push(8);

    # Pop elements
    print(pop()) ;
    print(pop());

# This code is contributed by AnkitRai01
```

C#

```
// C# implementation of the approach
using System;
using System.Collections.Generic;
class GFG {
    // element list with this frequency
    static Dictionary<int, Stack<int>> setMap = new Dictionary<int, Stack

    // Variable which stores maximum frequency
    // of the stack element
    static int maxfreq = 0;

    // Function to insert x in the stack
    static void push(int x)
    {
        int freq = 1;

        // Frequency of x
        if(freqMap.ContainsKey(x))
        {
            freq = freq + freqMap[x];
        }

        // Mapping of x with its frequency
        freqMap[x] = freq;

        // Update maxfreq variable
        if (freq > maxfreq)
            maxfreq = freq;

        // Map element to its frequency list
        // If given frequency list doesn't exist
        // make a new list of the required frequency
        if(!setMap.ContainsKey(freq))
        {
            setMap[freq] = new Stack<int>();
        }

        setMap[freq].Push(x);
    }

    // Function to remove maximum frequency element
    static int pop()
    {

        // Remove element from setMap
```

```
            // Decrement the frequency of the popped element
            freqMap[top] = freqMap[top] - 1;

            // If whole list is popped
            // then decrement the maxfreq
            if (setMap[maxfreq].Count == 0)
                maxfreq--;
            return top;
        }

    static void Main() {
        // Push elements to the stack
        push(4);
        push(6);
        push(7);
        push(6);
        push(8);

        // Pop elements
        Console.WriteLine(pop()) ;
        Console.WriteLine(pop());
    }
}

// This code is contributed by rameshtravel07.
```

Javascript

```
<script>

// Javascript implementation of the approach

// freqMap is to map element to its frequency
var freqMap = new Map();

// setMap is to map frequency to the
// element list with this frequency
var setMap = new Map();

// Variable which stores maximum frequency
```

```
function push(x)
{

    // Frequency of x
    if(!freqMap.has(x))
        freqMap.set(x, 1)
    else
        freqMap.set(x, freqMap.get(x)+1)

    var freq = freqMap.get(x)

    // Mapping of x with its frequency
    freqMap.set(x, freq);

    // Update maxfreq variable
    if (freq > maxfreq)
        maxfreq = freq;

    // Map element to its frequency list
    // If given frequency list doesn't exist
    // make a new list of the required frequency
    if(!setMap.has(freq))
    {
        setMap.set(freq, [x])
    }
    else
    {
        var tmp = setMap.get(freq);
        tmp.push(x);
        setMap.set(freq, tmp);
    }
}

// Function to remove maximum frequency element
function pop()
{

    // Remove element from setMap
    // from maximum frequency list
    var tmp = setMap.get(maxfreq);
    var top = tmp[tmp.length-1];
```

```
        if(freqMap.has(top))
            freqMap.set(top, freqMap.get(top)-1)

        // If whole list is popped
        // then decrement the maxfreq
        if (setMap.get(maxfreq).length == 0)
            maxfreq--;
        return top;
    }

    // Driver code
    // Push elements to the stack
    push(4);
    push(6);
    push(7);
    push(6);
    push(8);

    // Pop elements
    document.write( (pop()) + "<br>");
    document.write( (pop()));

    // This code is contributed by itsok.
    </script>
```

Output:

```
6
8
```

Previous

Number of triangles formed by joining vertices of n-sided polygon with one side common

Next

Overriding toString() method in Scala

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Maximum difference between frequency of two elements such that element having greater frequency is also greater**
13, Feb 17

02 **Maximum element in an array which is equal to its frequency**
17, Apr 20

03 **Maximum length prefix such that frequency of each character is atmost number of characters with minimum frequency**
30, Mar 20

04 **Find maximum in stack in O(1) without using additional stack**

05 **Design a stack to retrieve original elements and return the minimum element in O(1) time and O(1) space**
13, Jun 18

06 **Design a stack with operations on middle element**
08, Jun 13

07 **Check if frequency of character in one string is a factor or multiple of frequency of same character in other string**
16, Nov 18

08 **Count of Binary Strings of length N such that frequency of 1's exceeds frequency of 0's**

Article Contributed By :

dekay
@dekay

Vote for difficulty

Current difficulty : Expert

Easy Normal Medium Hard Expert

Improved By : andrew1234, ankthon, itsok, rameshtravel07, nikhatkhan11

Article Tags : frequency-counting, Hash, Stack

Practice Tags : Hash, Stack

Improve Article Report Issue

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

Load Comments

GeeksforGeeks

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

Courses

News

Top News

Technology

Work & Career

Business

Finance

Lifestyle

Knowledge

Languages

Python

Java

CPP

Golang

C#

SQL

Kotlin

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship

Do Not Sell My Personal Information