Tracking current Maximum Element in a Stack

Given a Stack, keep track of the maximum value in it. The maximum value may be the top element of the stack, but once a new element is pushed or an element is pop from the stack, the maximum element will be now from the rest of the elements.

Examples:

```
Input : 4 19 7 14 20
Output : Max Values in stack are
      4 19 19 19 20

Input : 40 19 7 14 20 5
Output :  Max Values in stack are
      40 40 40 40 40 40
```

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

Method 1 (Brute-force): We keep pushing the elements in the main stack and whenever we are asked to return the maximum element, we traverse the stack and print the max element.

Time Complexity : O(n)

Auxiliary Space : O(1)

Method 2 (Efficient): An efficient approach would be to maintain an auxiliary stack while pushing element in the main stack. This auxiliary stack will keep track of the maximum element.

Below is the step by step algorithm to do this:

1. Create an auxiliary stack, say 'trackStack' to keep the track of maximum element
2. Push the first element to both mainStack and the trackStack.
3. Now from the second element, push the element to the main stack. Compare the element with the top element of the track stack, if the current element is greater than top of trackStack then push the current element to trackStack otherwise push the top element of trackStack again into it.
4. If we pop an element from the main stack, then pop an element from the trackStack as well.
5. Now to compute the maximum of the main stack at any point, we can simply print the top element of Track stack.

Step by step explanation :

Suppose the elements are pushed on to the stack in the order {4, 2, 14, 1, 18}

Step 1 : Push 4, Current max : 4

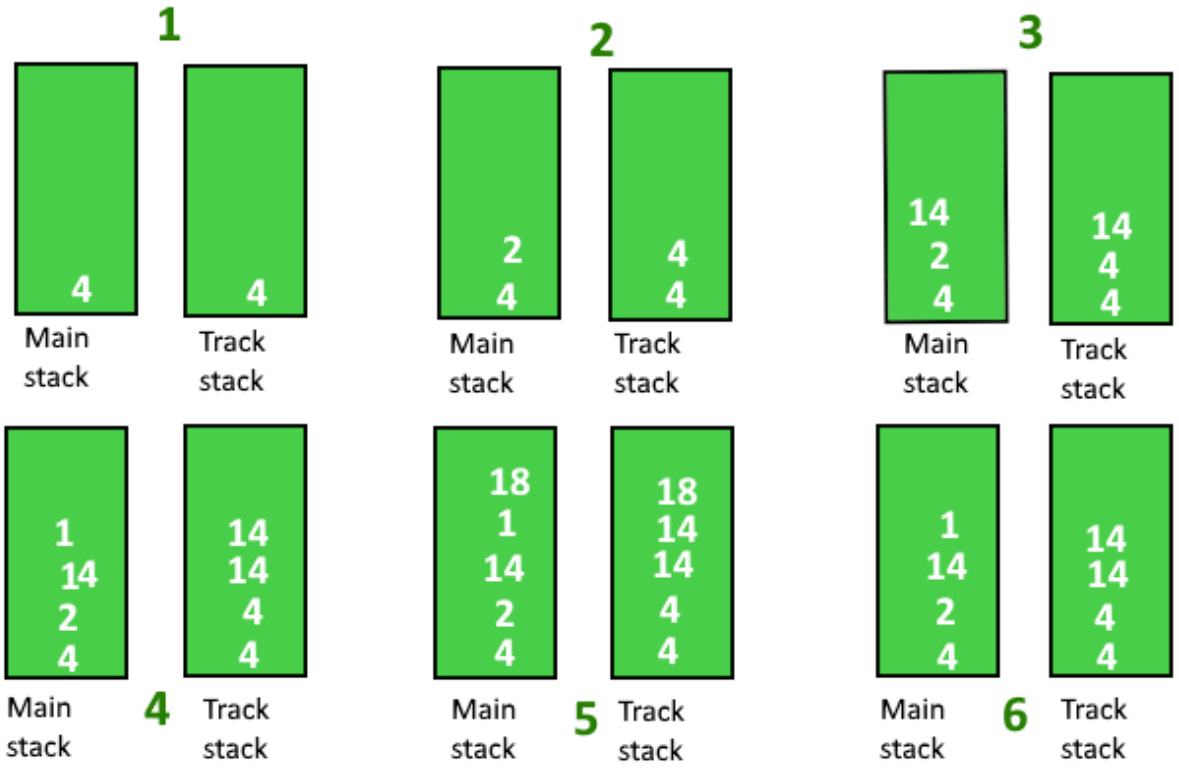
Step 2 : Push 2, Current max : 4

Step 3 : Push 14, Current max : 14

Step 4 : Push 1, Current max : 14

Step 5 : Push 18, Current max : 18

Step 6 : Pop 18, Current max : 14



Below is the C++ implementation of above approach:

```
// C++ program to keep track of maximum
// element in a stack
```

```
#include <bits/stdc++.h>
using namespace std;

class StackWithMax
{
    // main stack
    stack<int> mainStack;

    // tack to keep track of max element
    stack<int> trackStack;

public:
    void push(int x)
    {
        mainStack.push(x);
        if (mainStack.size() == 1)
        {
            trackStack.push(x);
            return;
        }

        // If current element is greater than
        // the top element of track stack, push
        // the current element to track stack
        // otherwise push the element at top of
        // track stack again into it.
        if (x > trackStack.top())
            trackStack.push(x);
        else
            trackStack.push(trackStack.top());
    }

    int getMax()
    {
        return trackStack.top();
    }

    int pop()
    {
        mainStack.pop();
        trackStack.pop();
```

```
    }
};

// Driver program to test above functions
int main()
{
    StackWithMax s;
    s.push(20);
    cout << s.getMax() << endl;
    s.push(10);
    cout << s.getMax() << endl;
    s.push(50);
    cout << s.getMax() << endl;
    return 0;
}
```

Run on IDE | Copy Code

Output:

```
20
20
50
```

Time Complexity : O(1)

Auxiliary Complexity : O(n)

This article is contributed by **Rohit**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

C# | Check if a Stack contains an element

Delete middle element of a stack

Design a stack with operations on middle element

Count the number of pop operations on stack to get each element of the array

Find maximum in a stack in O(1) time and O(1) extra space

Design a stack to retrieve original elements and return the minimum element in O(1) time and O(1) space

Tracking bird migration using Python-3

Project Idea | Ward Tracking System

Cookie Tracking and Stealing using Cross-Site Scripting

Project Idea | Real Time Vehicle Tracking

Maximum element in min heap

Maximum element between two nodes of BST

Maximum sum subarray removing at most one element

Type of array and its maximum element

Find maximum element of each row in a matrix

Article Tags : Stack

Practice Tags : Stack

Be the First to upvote.

2

To-do Done

Based on **20** vote(s)

Feedback Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

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

Load Comments

Share this post!

A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org