GeeksMan
Data Structure
Lesson 6

Trilok Kaushik

Founder of GeeksMan

Chirag Soni

Nupur Pahuja

Jessica Mishra

Team Coordinators

Get min at pop

You are given an array **A** of size **N**. You need to first push the elements of the array into a stack and then print minimum in the stack at each pop.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(N).

ALGORITHM :

1. Create two stacks , one to keep track of all the elements , s1 , and the other as a supporting stack to keep minimum elements.
2. While pushing the elements in the stack s1 , if the supporting stack is empty , then push the element in the supporting stack .
3. Else if the top of the supporting stack is greater than the incoming element then push the element in the supporting stack
4. While popping , if the popped element is equal to the top of the supporting stack , then pop it from the supporting stack also.
5. The top of the supporting stack is the minimum element.

CODE :

```
1.  //Initial Template for C++
2.
3.
4.  #include <bits/stdc++.h>
5.  using namespace std;
6.
7.  stack<int> _push(int arr[],int n);
8.
9.  void _getMinAtPop(stack<int>s);
10.
11.    // } Driver Code Ends
12.
13.
14.    //User function Template for C++
```

```cpp
15.    //User function Template for C++
16.    stack<int> _push(int arr[],int n)
17.    {
18.       stack<int>st;
19.       for(int i = 0;i<n;i++)
20.       {
21.            st.push(arr[i]);
22.       }
23.       return st;
24.     // your code here
25.    }
26.
27.    /* print minimum element of the stack each time
28.       after popping
29.    */
30.
31.    void _getMinAtPop(stack<int>s)
32.    {
33.       vector<int>v;
34.       while(!s.empty())
35.       {
36.            v.push_back(s.top());
37.            s.pop();
38.       }
39.       stack<int>ss;
40.       ss.push(v.back());
41.       for(int i = v.size() - 2;i >= 0 ;i--)
42.       {
43.            if(v[i] > ss.top())
44.                 ss.push(ss.top());
45.             else
46.                 ss.push(v[i]);
47.       }
48.       while(!ss.empty())
49.       {
50.            cout << ss.top() << " ";
51.            ss.pop();
52.       }
53.    }
54.
55.
56.
57.
```

```cpp
58.
59.    /* inserts elements of the array into
60.       stack and return the stack
61.    */
62.
63.
64.    // { Driver Code Starts.
65.    int main() {
66.        int t;
67.        cin>>t;
68.        while(t--)
69.        {
70.            int n;
71.            cin>>n;
72.            int arr[n];
73.            for(int i=0;i<n;i++)
74.            cin>>arr[i];
75.            stack<int>mys=_push(arr,n);
76.            _getMinAtPop(mys);
77.            cout<<endl;
78.
79.
80.
81.        }
82.        return 0;
83.    }
84.
85.     // } Driver Code Ends
86.
```

Get minimum element from stack

You are given N elements and your task is to Implement a Stack in which you can get a minimum element.

By solving the above ques (Get min at pop) , you can solve this problem easily by creating a supporting stack , which can store the minimum elements. But we will try to solve this question in O(1) time complexity .

So , for O(1) time complexity , we cannot use any containers , and hence we will define a variable to store the minimum value.

ALGORITHM:

push(x):

1. Define a variable minEle to store minimum value.
2. If the stack is empty , then push x in the stack and minEle = x;
3. If the stack is not empty , then either x > stack.top() or x < stack.top().
4. If x > minEle , then push x in the stack .
5. If x < minEle , then minEle = x and push (2*x - stack.top()) in the stack , so that we can keep the track of minimum element before x.

pop():

1. As and when an element is popped out from the stack , two cases arise , either the popped element was the minimum element , or it was not .
2. If the popped element is less than minEle , then minEle = (2*minEle - tp) , where tp is the popped out element . Hence , we can retrieve the previous minimum element.

CODE :

```cpp
1. #include <bits/stdc++.h>
2. using namespace std;
3.
4. struct MyStack
5. {
6.     stack<int> s;
7.     int minEle;
```

```cpp
8.      void getMin()
9.      {
10.         if (s.empty())
11.             cout << "Stack is empty\n";
12.         else
13.             cout <<"Minimum Element in the stack is: " <<
    minEle << endl;
14.     }
15.     void pop()
16.     {
17.         if (s.empty())
18.         {
19.             cout << "Stack is empty" << endl;
20.             return;
21.         }
22.
23.         cout << "Element Popped : ";
24.         int tp = s.top();
25.         s.pop();
26.         if (tp < minEle)
27.         {
28.             cout << "Element Popped : "<< minEle << endl;
29.             minEle = 2*minEle - tp;
30.         }
31.         else
32.             cout << tp << endl;
33.     }
34.
35.     void push(int x)
36.     {
37.         if (s.empty())
38.         {
39.             minEle = x;
40.             s.push(x);
41.             cout << "Element Pushed : " << x << endl;
42.             return;
43.         }
44.         if (x < minEle)
45.         {
46.             s.push(2*x - minEle);
47.             minEle = x;
48.         }
49.         else
```

```cpp
50.            s.push(x);
51.            cout << "Element Pushed : " << x <<endl;
52.        }
53.    };
54.
55.    int main()
56.    {
57.        MyStack s;
58.        s.push(2);
59.        s.push(4);
60.        s.getMin();
61.        s.push(3);
62.        s.push(1);
63.        s.getMin();
64.        s.pop();
65.        s.getMin();
66.        s.pop();
67.        s.getMin();
68.        return 0;
69.    }
70.
```

Link of the code : https://sapphireengine.com/@/r9hsan

Max rectangle

Given a binary matrix. Find the maximum area of a rectangle formed only of 1s in the given matrix.

Input: n = 4, m = 4   **Output:** 8

M[][] = {{0 1 1 0},
         {1 1 1 1},
         {1 1 1 1},
         {1 1 0 0}}

CODE:

```
1. #include <bits/stdc++.h>
2. using namespace std;
3. #define MAX 1000
4.
5. int maxArea(int M[MAX][MAX], int n, int m);
6. int main() {
7.    int T;
8.    cin >> T;
9.
10.     int M[MAX][MAX];
11.
12.     while (T--) {
13.         int n, m;
14.         cin >> n >> m;
15.
16.         for (int i = 0; i < n; i++) {
17.             for (int j = 0; j < m; j++) {
18.                 cin >> M[i][j];
19.             }
20.         }
21.         cout << maxArea(M, n, m) << endl;
22.     }
23. }
24. // } Driver Code Ends
25. int mah(int A[],int n)
```

```
26.   {
27.       if(n==0)
28.           return 0;
29.       if(n==1)
30.           return A[0];
31.       int mx=0;
32.       stack<int> st;
33.       st.push(0);
34.       int i;
35.       for(i=1;i<n;i++)
36.       {
37.           if(A[i]>=A[st.top()])
38.               st.push(i);
39.           else
40.           {
41.               while(!st.empty() && A[st.top()]>A[i])
42.               {
43.                   int temp=A[st.top()];
44.                   st.pop();
45.                   if(st.empty())
46.                   {
47.                       mx=max(mx,temp*i);
48.                   }
49.                   else
50.                   {
51.                       mx=max(mx,temp*(i-st.top()-1));
52.                   }
53.               }
54.               st.push(i);
55.           }
56.       }
57.   while(!st.empty())
58.           {
59.               int temp=A[st.top()];
60.               st.pop();
61.               if(st.empty())
62.               {
63.                   mx=max(mx,temp*i);
64.               }
65.               else
66.               {
67.                   mx=max(mx,temp*(i-st.top()-1));
68.               }
```

```
69.                    }
70.    return mx;
71.    }
72.
73.    int maxArea(int M[MAX][MAX], int n, int m) {
74.        int m1=m;
75.        int maxarea=0,i,j,x;
76.        maxarea=max(maxarea,mah(M[0],m1));
77.        for(i=1;i<n;i++)
78.        {
79.            for(j=0;j<m;j++)
80.            {
81.                if(M[i][j]==0)
82.                {
83.                    M[0][j]=0;
84.                }
85.                else
86.                {
87.                    M[0][j]=M[0][j]+M[i][j];
88.                }
89.            }
90.            maxarea=max(maxarea,mah(M[0],m1));
91.        }
92.        return maxarea;
93.    }
```

Maximum of minimum for every window size

Given an integer array **A[]** of size **N**. The task is to find the maximum of the minimum of every window size in the array.

Input: 7   **Output:**

      10 20 30 50 10 70 30   70 30 20 10 10 10 10

CODE:

```cpp
1. #include <bits/stdc++.h>
2. #include <iostream>
3. using namespace std;
4.
5. vector<int> NGR(int A[],int n)
6. {
7. stack<int> st;
8. vector<int> v;
9.    int i;
10.   for(i=n-1;i>=0;i--)
11.   {
12.       if(st.size()==0)
13.       {
14.           v.push_back(n);
15.       }
16.       else if(A[i]>A[st.top()])
17.           v.push_back(st.top());
18.       else
19.       {
20.           while(st.size()>0 &&  A[st.top()]>=A[i])
21.               st.pop();
22.           if(st.size()==0)
23.               v.push_back(n);
24.           else
25.               v.push_back(st.top());
26.
27.       }
28.       st.push(i);
29.   }
30.   vector<int>v1;
31.   for(i=n-1;i>=0;i--)
32.   {
```

```cpp
33.        v1.push_back(v[i]);
34.    }
35.    return v1;
36.
37.    }
38.    vector<int> NGL(int A[],int n)
39.    {
40.        stack<int> st;
41.    vector<int> v;
42.        int i;
43.    for(i=0;i<n;i++)
44.    {
45.        if(st.size()==0)
46.        {
47.            v.push_back(-1);
48.        }
49.        else if(A[i]>A[st.top()])
50.            v.push_back(st.top());
51.        else
52.        {
53.            while(st.size()>0 &&  A[st.top()]>=A[i])
54.                st.pop();
55.            if(st.size()==0)
56.                v.push_back(-1);
57.            else
58.                v.push_back(st.top());
59.
60.        }
61.        st.push(i);
62.    }
63.    return v;
64.    }
65.
66.    void solve()
67.    {
68.        int n;
69.        cin>>n;
70.        int A[n];
71.        int i;
72.        for(i=0;i<n;i++)
73.        {
74.            cin>>A[i];
75.        }
```

```cpp
    vector<int> vr,vl,v;
    vr=NGR(A,n);
     vl=NGL(A,n);
    int m=0,c;
        for(i=0;i<n;i++)
        { c=vr[i]-vl[i]-1;
         v.push_back(c);

        }
    int Arr[n+1];
    for(i=0;i<=n;i++)
        Arr[i]=-1;
    for(i=1;i<n+1;i++)
    {
        Arr[v[i-1]]=max(Arr[v[i-1]],A[i-1]);
    }
    for(i=n-1;i>0;i--)
    {
        Arr[i]=max(Arr[i],Arr[i+1]);
    }
     for(i=1;i<=n;i++)
     {
         cout<<Arr[i]<<" ";
     }
}

int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        solve();
        cout<<endl;
    }
    return 0;
}
```