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

```
    for(int i = 0;i<n;i++)
    {
        st.push(arr[i]);
    }
    return st;
   // your code here
}

/* print minimum element of the stack each time
   after popping
*/

void _getMinAtPop(stack<int>s)
{
    int l = s.size();
    int a[l];
    int i;
    for(i = l-1;i>=0;i--)
    {
        a[i] = s.top();
        s.pop();
    }
    stack<int>s1;
    stack<int>ss;
    for(i = 0;i<l;i++)
    {
        if(ss.empty())
            {
                s1.push(a[i]);
                ss.push(a[i]);
            }
        else
        {
            if(ss.top() > a[i])
            {
                ss.push(a[i]);
                s1.push(a[i]);
            }
            else
            {
                s1.push(a[i]);
            }
        }
```

```
58.      }
59.      for(i = 0;i<l;i++)
60.      {
61.          int x = s1.top();
62.              if(ss.top() == x)
63.                  {
64.                      cout << x << " ";
65.                      ss.pop();
66.                      s1.pop();
67.                  }
68.              else
69.              {
70.                  cout << ss.top() << " ";
71.                  s1.pop();
72.              }
73.      }
74.  }
75.
76.  /* inserts elements of the array into
77.     stack and return the stack
78.  */
79.
80.
81.  // { Driver Code Starts.
82.  int main() {
83.     int t;
84.     cin>>t;
85.     while(t--)
86.     {
87.         int n;
88.         cin>>n;
89.         int arr[n];
90.         for(int i=0;i<n;i++)
91.         cin>>arr[i];
92.         stack<int>mys=_push(arr,n);
93.         _getMinAtPop(mys);
94.         cout<<endl;
95.
96.
97.
98.     }
99.     return 0;
100. }
```

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 :

```
1. #include <bits/stdc++.h>
2. using namespace std;
3.
4. struct MyStack
```

```
5. {
6.       stack<int> s;
7.       int minEle;
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);
```

```
                minEle = x;
            }
            else
            s.push(x);
            cout << "Element Pushed : " << x <<endl;
    }
};

int main()
{
    MyStack s;
    s.push(2);
    s.push(4);
    s.getMin();
    s.push(3);
    s.push(1);
    s.getMin();
    s.pop();
    s.getMin();
    s.pop();
     s.getMin();
    return 0;
}

```

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

Output: 8

CODE:

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 1000

int maxArea(int M[MAX][MAX], int n, int m);
int main() {
    int T;
    cin >> T;

    int M[MAX][MAX];

    while (T--) {
        int n, m;
        cin >> n >> m;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                cin >> M[i][j];
            }
        }
        cout << maxArea(M, n, m) << endl;
    }
}
// } Driver Code Ends
int mah(int A[],int n)
```

```
{
    if(n==0)
        return 0;
    if(n==1)
        return A[0];
    int mx=0;
     stack<int> st;
     st.push(0);
     int i;
     for(i=1;i<n;i++)
     {
        if(A[i]>=A[st.top()])
            st.push(i);
        else
        {
            while(!st.empty() && A[st.top()]>A[i])
            {
                int temp=A[st.top()];
                st.pop();
                if(st.empty())
                {
                    mx=max(mx,temp*i);
                }
                else
                {
                    mx=max(mx,temp*(i-st.top()-1));
                }
            }
            st.push(i);
        }
     }
while(!st.empty())
            {
                int temp=A[st.top()];
                st.pop();
                if(st.empty())
                {
                    mx=max(mx,temp*i);
                }
                else
                {
                    mx=max(mx,temp*(i-st.top()-1));
                }
```

```
            }
return mx;
}

int maxArea(int M[MAX][MAX], int n, int m) {
   int m1=m;
   int maxarea=0,i,j,x;
   maxarea=max(maxarea,mah(M[0],m1));
   for(i=1;i<n;i++)
   {
       for(j=0;j<m;j++)
       {
           if(M[i][j]==0)
           {
               M[0][j]=0;
           }
           else
           {
               M[0][j]=M[0][j]+M[i][j];
           }
       }
       maxarea=max(maxarea,mah(M[0],m1));
   }
   return maxarea;
}
```

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:

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

vector<int> NGR(int A[],int n)
{
stack<int> st;
vector<int> v;
    int i;
   for(i=n-1;i>=0;i--)
   {
      if(st.size()==0)
      {
          v.push_back(n);
      }
      else if(A[i]>A[st.top()])
          v.push_back(st.top());
      else
      {
          while(st.size()>0 &&  A[st.top()]>=A[i])
              st.pop();
          if(st.size()==0)
              v.push_back(n);
          else
              v.push_back(st.top());

      }
      st.push(i);
   }
  vector<int>v1;
  for(i=n-1;i>=0;i--)
  {
```

```
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.      }
```

```
76.      vector<int> vr,vl,v;
77.      vr=NGR(A,n);
78.       vl=NGL(A,n);
79.      int m=0,c;
80.          for(i=0;i<n;i++)
81.          { c=vr[i]-vl[i]-1;
82.           v.push_back(c);
83.
84.          }
85.      int Arr[n+1];
86.      for(i=0;i<=n;i++)
87.          Arr[i]=-1;
88.      for(i=1;i<n+1;i++)
89.      {
90.          Arr[v[i-1]]=max(Arr[v[i-1]],A[i-1]);
91.      }
92.      for(i=n-1;i>0;i--)
93.      {
94.          Arr[i]=max(Arr[i],Arr[i+1]);
95.      }
96.       for(i=1;i<=n;i++)
97.       {
98.           cout<<Arr[i]<<" ";
99.       }
100. }
101.
102. int main()
103. {
104.     int t;
105.     cin>>t;
106.     while(t--)
107.     {
108.         solve();
109.         cout<<endl;
110.     }
111.     return 0;
112. }
```