DEPARTMENT OF
COMPUTERSCIENCE&ENGINEERING
Discover. Learn. Empower.

DOMAIN WINTER WINNING CAMP

Name:Divya Sharma

Branch: BE-CSE

Semester:5th

UID:22BCS14192

Section/Group:22BCS_FL_IOT-603/B

DAY-4[23-12-2024]

STACK AND QUEUE STANDARD QUESTION

VERY EASY:

Q1.Design astackthatsupportspush,pop,top,andretrievingtheminimumelementin constant time.

ImplementtheMinStackclass:

- MinStack()initializesthestack object.
- voidpush(intval) pushestheelementvalontothestack.
- voidpop() removestheelementonthetopofthestack.
- inttop() gets thetopelementofthe stack.
- intgetMin() retrievestheminimumelementinthe stack.

Youmustimplementa solutionwithO(1)timecomplexityfor each function.

Example 1:

Input

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

Output

[null,null,null,null,-3,null,0,-2]

Explanation

MinStackminStack=newMinStack();

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.getMin();//return-3

minStack.pop();

minStack.top(); // return 0

minStack.getMin();//return-2

CODE:

```
#include<iostream>
#include <stack>
#include <limits.h>

classMinStack{
private:
   std::stack<int> stack;
   std::stack<int>minStack;

public:
   MinStack(){
      //Constructorinitializesanemptystack
   }

   voidpush(intval){
      stack.push(val);
      //PushontominStackifit'sempty orvalislessthanorequaltothecurrentminimum if
      (minStack.empty() || val <= minStack.top()) {
         minStack.push(val);
      }
   }

   voidpop(){
      if(stack.top()==minStack.top()){
         minStack.pop();
      }
      stack.pop();
   }

   inttop(){
      returnstack.top();
   }

   intgetMin(){
      returnminStack.top();
   }
};

intmain(){
   MinStackminStack;
```

```
    minStack.push(-2);
    minStack.push(0);
    minStack.push(-3);
    std::cout<<minStack.getMin()<<std::endl;//Output:-3
    minStack.pop();
    std::cout << minStack.top() << std::endl;    // Output: 0
    std::cout<<minStack.getMin()<<std::endl;//Output:-2
    return 0;
}
```

OUTPUT:

```
-3
0
-2
```

Example 2:

Input:

["MinStack","push","push","push","push","getMin","pop","getMin","top","getMin"]

[[],[5],[3],[7],[3],[],[],[],[],[]]

Output

[null,null,null, null,null, 3, null, 3, 7, 3]

Explanation:

MinStack minStack = new MinStack();

minStack.push(5); #Stack:[5],MinStack:[5]

minStack.push(3); #Stack: [5,3],MinStack: [5,3]

minStack.push(7); #Stack: [5,3, 7],MinStack: [5,3]

minStack.push(3); #Stack:[5,3,7,3],MinStack:[5,3,3]

minStack.getMin();# Returns 3

minStack.pop(); #Removes3;Stack:[5,3,7],MinStack:[5,3]

minStack.getMin();# Returns 3

minStack.top(); # Returns 7

minStack.getMin();#Returns3

- Minimumvaluesaremaintainedas:[5]→[5,3]→[5,3]→ [5,3]
- Afterpops,theminimumvaluesupdate accordingly.

CODE:

```
#include<iostream>

#include <stack>

#include <climits>


classMinStack{

private:

    std::stack<int>mainStack;

    std::stack<int> minStack;


public:

    MinStack(){

        //InitializeminStackwithasentinelvalue

        minStack.push(INT_MAX);

    }


    void push(int val) {

        mainStack.push(val);

        // Update the minimum stack

        minStack.push(std::min(val,minStack.top()));

    }


    void pop() {

        mainStack.pop();

        minStack.pop();
```

```
    }

    inttop(){
        returnmainStack.top();
    }

    intgetMin() {
        returnminStack.top();
    }
};

intmain() {
    MinStack minStack;
    minStack.push(5);    //Stack:[5],MinStack:[5]
    minStack.push(3);    //Stack:[5,3],MinStack:[5, 3]
    minStack.push(7);    //Stack:[5,3, 7],MinStack:[5, 3]
    minStack.push(3);    //Stack:[5, 3,7, 3],MinStack: [5,3, 3]

    std::cout << minStack.getMin() << std::endl; // Output: 3
    minStack.pop();      //Removes3;Stack:[5,3,7],MinStack:[5,3]
    std::cout << minStack.getMin() << std::endl; // Output: 3
    std::cout << minStack.top() << std::endl;    // Output: 7
    std::cout<<minStack.getMin()<<std::endl;//Output:3

    return0;
```

}

OUTPUT:

```
3
3
7
3
```

Example 3:

Input:

["MinStack","push","push", "push","getMin","pop","getMin","pop", "getMin"]

[[],[2],[1], [4], [],[],[], [], []]

Output:

[null,null,null, null,1, null, 1, null, 2]

Explanation:

minStack=MinStack()
minStack.push(2)
minStack.push(1)
minStack.push(4)
minStack.push(1)
print(minStack.getMin())#Output:1
minStack.pop()
print(minStack.getMin())#Output:1
minStack.pop()
print(minStack.getMin())#Output:1
minStack.pop()
print(minStack.getMin())#Output:2

- Minimumvaluesaremaintainedas:[2]→[2,1]→[2,1]→[2,1]
- Afterpops,theminimumvaluesupdateaccordingly.

Constraints:

- -2^31<= val <= 2^31-1
- Methodspop,topand getMinoperationswillalwaysbecalledonnon-emptystacks.
- Atmost3* 10^4callswillbe madetopush,pop,top, andgetMin.

Sources:https://leetcode.com/problems/min-stack/

CODE:

```
#include<iostream>
#include <stack>
#include <limits.h>

classMinStack{
private:
    std::stack<int> stack;
    std::stack<int>minStack;

public:
    MinStack(){
        //Constructorinitializesanemptystack
    }

    voidpush(intval){
        stack.push(val);
        //PushontominStackifit'sempty orvalislessthanorequaltothecurrentminimum if
        (minStack.empty() || val <= minStack.top()) {
            minStack.push(val);
        }
    }

    voidpop(){
        if(stack.top()==minStack.top()){
            minStack.pop();
        }
        stack.pop();
    }

    inttop(){
        returnstack.top();
    }
```

```
    intgetMin(){
        returnminStack.top();
    }
};

intmain(){
    MinStackminStack;
    minStack.push(5);    //Stack:[5],MinStack: [5]
    minStack.push(3);    //Stack:[5,3],MinStack:[5, 3]
    minStack.push(7);    //Stack:[5,3,7],MinStack:[5,3]
    minStack.push(3);    //Stack:[5,3,7,3],MinStack: [5,3,3]

    std::cout << minStack.getMin() << std::endl; // Output: 3
    minStack.pop();      //Removes3;Stack:[5,3,7],MinStack:[5,3]
    std::cout << minStack.getMin() << std::endl; // Output: 3
    std::cout << minStack.top() << std::endl;    // Output: 7
    std::cout<<minStack.getMin()<<std::endl;//Output:3

    return0;
}
```

OUTPUT:

```
3
3
7
3
```

EASY LEVEL:-

Q4.Theschoolcafeteriaofferscircularandsquaresandwichesatlunchbreak,referredtoby numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.

Thenumberofsandwichesinthecafeteriaisequaltothenumberofstudents.Thesandwiches are placed in a stack. At each step:

Ifthestudentat thefrontofthequeueprefers thesandwichonthetopofthestack, theywilltakeit and leave the queue.
Otherwise,theywillleaveitand go tothequeue'send.
Thiscontinues untilnoneofthequeuestudentswant totakethetopsandwichandarethusunableto eat.

Youare giventwointegerarraysstudentsandsandwicheswheresandwiches[i]isthetype ofthe i thsandwichinthestack(i =0isthetopofthestack)andstudents[j]isthepreference of the j th student in the initial queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

Example 1:

Input:students=[1,1,0,0],sandwiches= [0,1,0,1]
Output:0
Explanation:
- Frontstudentleavesthe topsandwichandreturnsto theendof thelinemakingstudents=[1,0,0,1].
- Frontstudentleavesthe topsandwichandreturnsto theendof thelinemakingstudents=[0,0,1,1].
- Frontstudenttakesthetopsandwichandleavesthelinemakingstudents=[0,1,1]and sandwiches =[1,0,1].
- Frontstudentleavesthe topsandwichandreturnsto theendof thelinemakingstudents=[1,1,0].
- Frontstudenttakesthetopsandwichandleavesthelinemakingstudents=[1,0]andsandwiches= [0,1].
- Frontstudentleavesthe topsandwichandreturnsto theendof thelinemakingstudents=[0,1].
- Frontstudenttakesthetopsandwichandleavesthelinemakingstudents=[1]andsandwiches= [1].
- Frontstudenttakesthetopsandwichandleavesthelinemakingstudents=[]andsandwiches=[]. Hence all students are able to eat.

Example 2:
Input:students=[1,1,1,0,0,1],sandwiches= [1,0,0,0,1,1]
Output:3
Constraints:

- 1<=students.length,sandwiches.length<= 100
- students.length== sandwiches.length
- sandwiches[i]is 0 or 1.
- students[i]is 0 or 1.

Approach

- Createtwoqueuesofstudentsandsandwiches
- Andacountvariabletocheckifis loopinleft student
- Ifstudentsinthequeuecannothavetheirorderedsandwiches,itmakesaloop.Ifitisaloop, just break and return the result
- Thenimplementtheprogram likethegivenrules.
  Timecomplexity:O(n)

Spacecomplexity:O(n)

Reference:https://leetcode.com/problems/number-of-students-unable-to-eat-lunch/description/

CODE:

```
#include<iostream>
#include <queue>
#include <vector>

intcountStudentsUnableToEat(std::vector<int>&students,std::vector<int>&sandwiches){
   std::queue<int> studentQueue;
   std::queue<int>sandwichStack;

   //Populatethequeues
   for (int student : students) {
      studentQueue.push(student);
   }
   for (int sandwich : sandwiches) {
      sandwichStack.push(sandwich);
   }

   intcount = 0; //Counter forunsuccessful rotations

   while(!studentQueue.empty()&&count<studentQueue.size()){ if
      (studentQueue.front() == sandwichStack.front()) {
         //Studenteatsthesandwich
         studentQueue.pop();
         sandwichStack.pop();
         count=0; // Reset counter
      }else{
         // Student moves to the back of the queue
         studentQueue.push(studentQueue.front());
         studentQueue.pop();
         count++;
      }
   }

   //Remainingstudentsarethoseunabletoeat
   return studentQueue.size();
}

intmain() {
   std::vector<int>students1= {1,1,0, 0};
   std::vector<int>sandwiches1={0,1,0, 1};
   std::cout<<countStudentsUnableToEat(students1,sandwiches1)<<std::endl;//Output:0

   std::vector<int>students2={1,1,1, 0,0, 1};
```

```
    std::vector<int>sandwiches2={1,0,0, 0,1, 1};
    std::cout<<countStudentsUnableToEat(students2,sandwiches2)<<std::endl;//Output:3

    return0;
}
```

OUTPUT:

```
0
3
```

MEDIUM:-

Q5.Given acircularintegerarraynums(i.e.,thenextelementofnums[nums.length-1]is nums[0]), return the next greater number for every element in nums.

Thenextgreaternumberofanumberxisthefirstgreaternumbertoitstraversing-ordernext in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return -1 for this number.

Example 1:

Input:nums=[1,2,1]

Output: [2,-1,2]

Explanation:

- Thefirst1'snextgreaternumberis 2;
- Thenumber 2can'tfindnextgreaternumber.
- Thesecond1'snextgreaternumberneeds tosearch circularly,whichisalso2.

Example 2:

Input:nums= [1,2,3,4,3]

Output:[2,3,4,-1,4]

Constraints:

- 1<=nums.length <= 104
- -109<= nums[i]<= 109

Reference: https://leetcode.com/problems/next-greater-element-ii/description/

CODE:

```
#include<iostream>
#include <vector>
#include <stack>
```

```
std::vector<int>nextGreaterElements(conststd::vector<int>&nums){ int
    n = nums.size();
    std::vector<int>result(n,-1);//Initializeresultarraywith-1
    std::stack<int> stk; // Stack to store indices

    //Iteratethroughthearraytwice(tosimulatecircularbehavior) for
    (int i = 0; i < 2 * n; ++i) {
        intnum=nums[i%n];

        //Checkfornextgreaterelement
        while(!stk.empty()&&nums[stk.top()]<num){
            result[stk.top()] = num;
            stk.pop();
        }

        //Pushindexontostackforfirstpassonly if (i
        < n) {
            stk.push(i);
        }
    }

    returnresult;
}

intmain(){
    std::vector<int>nums1={1,2,1};
    std::vector<int>result1=nextGreaterElements(nums1); for
    (int num : result1) {
        std::cout<<num<<"";
    }
    std::cout<<std::endl;//Output:2-12

    std::vector<int> nums2 = {1, 2, 3, 4, 3};
    std::vector<int>result2=nextGreaterElements(nums2);
    for (int num : result2) {
        std::cout<<num<<"";
    }
    std::cout<<std::endl;//Output:234-14

    return0;
}
```

OUTPUT:

```
2 -1 2
2 3 4 -1 4
```

HARD

Q6. You are given an array of integers nums, there is a sliding window of size k which is movingfromthevery leftofthearray tothevery right.Youcanonlyseetheknumbers inthe window. Each time the sliding window moves right by one position.
Returnthemaxslidingwindow.
Example 1:
Input:nums =[1,3,-1,-3,5,3,6,7],k=3
Output:[3,3,5,5,6,7]
Explanation:

Window position			Max
[13-1]-353	6	7	3
1[3-1-3]53	6	7	3
13 [-1-35]3	6	7	5
13-1 [-353]	6	7	5
13-1-3 [536]7			6
13-1-35 [367]			7

Example 2:
Input:nums=[1],k=1
Output: [1]
Constraints:

- 1<=nums.length <= 105
- -104<= nums[i]<= 104
- 1<=k<=nums.length

Referenece:https://leetcode.com/problems/sliding-window-maximum/description/

CODE:

```
#include<iostream>
#include <vector>
#include <deque>

std::vector<int>maxSlidingWindow(conststd::vector<int>&nums,intk){ std::vector<int>
   result;
   std::deque<int>dq;//Willstoreindicesof nums

   for(inti=0;i<nums.size();++i){
      //Removeelementsoutofthecurrentwindow if
      (!dq.empty() && dq.front() == i - k) {
         dq.pop_front();
      }

      //Removeelementssmallerthanthecurrentelementfromtheback while
      (!dq.empty() && nums[dq.back()] < nums[i]) {
         dq.pop_back();
      }

      //Addthecurrentelement'sindextothedeque
      dq.push_back(i);
```

```
        //Addthemaximumelementofthecurrentwindowtotheresult if (i >=
        k - 1) {
            result.push_back(nums[dq.front()]);
        }
    }

    returnresult;
}

intmain(){
    std::vector<int>nums1={1,3,-1,-3,5,3,6,7}; int k1
    = 3;
    std::vector<int>result1=maxSlidingWindow(nums1,k1); for
    (int num : result1) {
        std::cout<<num<<"";
    }
    std::cout<<std::endl;// Output:335567

    std::vector<int>nums2={1};
    int k2 = 1;
    std::vector<int>result2=maxSlidingWindow(nums2,k2); for
    (int num : result2) {
        std::cout<<num<<"";
    }
    std::cout<<std::endl;//Output:1

    return0;
}
```

OUTPUT:

```
3 3 5 5 6 7
1
```

VERYHARD

7.There are a number of plants in a garden. Each of the plants has been treated with some amountofpesticide.Aftereachday,ifanyplanthasmorepesticidethantheplantonitsleft, being weaker than the left one, it dies.

Youaregiventheinitialvaluesofthepesticideineach oftheplants.Determinethenumberof days after which no plant dies, i.e. the time after which there is no plant with more pesticide content than the plant to its left.

Example1

p =[3,6,2,7,5]

//pesticide levels

Usea 1-indexedarray.Onday 1, plants2and4dieleavingp'=[3,2,5]. Onday2, plant 3inp'diesleavingp"=[3.2].Thereisnoplantwithahigherconcentrationofpesticidethan the one to its left, so plants stop dying after day 2 .

FunctionDescription
CompletethefunctionpoisonousPlantsintheeditorbelow.

poisonousPlants has the following parameter(s):

intp[n]:thepesticidelevelsineachplant

Returns
- int:the numberofdaysuntilplantsnolongerdiefrompesticide

InputFormat

- Thefirstlinecontainsaninteger n,thesizeofthe arrayp.
- Thenextlinecontains nspace-separatedintegersp[i].

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq p[i] \leq 10^9$

Example2:

SampleInput

7

6 5 8 4 7 10 9

SampleOutput

2

Explanation

Initiallyallplantsare alive.

Plants={(6,1),(5,2), (8,3),(4,4),(7,5), (10,6),(9,7)}

Plants[k] = (i,j) => j^{th}plant has pesticide amount = i.

Afterthe1^{st}day,4plantsremainasplants3,5,and6die. Plants

= {(6,1), (5,2), (4,4), (9,7)}

Afterthe2^{nd}day, 3plantssurviveasplant7dies. Plants

= {(6,1), (5,2), (4,4)}

Plantsstopdyingafterthe2^{nd}day.

Refrences:https://www.hackerrank.com/challenges/poisonous-plants/problem?isFullScreen=true

CODE:

```
#include <iostream>
#include <vector>
#include <stack>
#include<algorithm>

intpoisonousPlants(conststd::vector<int>&p){ int
   n = p.size();
   std::vector<int>days(n,0);//Days eachplanttakestodie
   std::stack<int> s; // Stack to keep track of plant indices

   intmaxDays=0;//Maximumdaysrequiredforplantstostopdying for (int i

   = 0; i < n; ++i) {
      //Checkifcurrentplantwilldie
      intday=0;
      while(!s.empty()&&p[s.top()]>=p[i]){
         day=std::max(day,days[s.top()]);
         s.pop();
      }

      //Ifstackisnotempty,currentplantwilldie if
      (!s.empty()) {
         days[i]=day+1;
      }

      maxDays=std::max(maxDays,days[i]);
      s.push(i);
   }

   returnmaxDays;
}

intmain(){
   std::vector<int>p1={3,6,2,7,5};
   std::cout<<poisonousPlants(p1)<<std::endl;//Output:2

   std::vector<int>p2= {6,5,8,4,7,10,9};
   std::cout<<poisonousPlants(p2)<<std::endl;//Output:2

   return0;
}
```

OUTPUT:

```
2
2
```

DEPARTMENT OF COMPUTERSCIENCE&ENGINEERING