# 基本数据结构--栈

目录

[1. 火车进栈 1](#_Toc31156_WPSOffice_Level1)

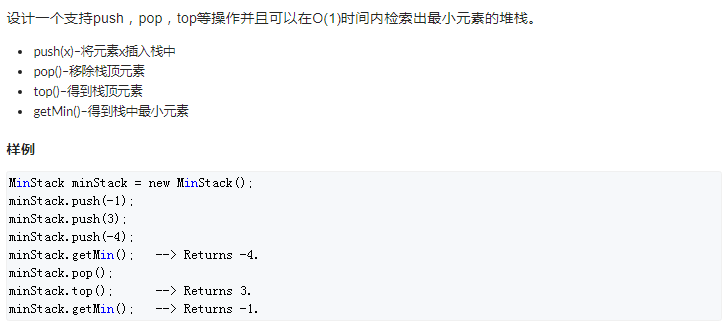
[2. Editor 2](#_Toc7442_WPSOffice_Level1)

[3. 进出栈序列问题 (相当于求卡特兰数第n项) 3](#_Toc8498_WPSOffice_Level1)

[4. Largest Rectangle in a Histogram](#_Toc1052_WPSOffice_Level1) [7](#_Toc1052_WPSOffice_Level1)

### 一、包含min函数的栈

题目描述：



代码:

**class** MinStack {

**public**:

    /\*\* initialize your data structure here. \*/

    stack<**int**> stk , min\_stk;

    MinStack() {}

**void** push(**int** x) {

        stk.push(x);

**if**(min\_stk.size()) x = min(x,min\_stk.top());

        min\_stk.push(x);

    }

**void** pop() {

        stk.pop();

        min\_stk.pop();

    }

**int** top() {

**return** stk.top();

    }

**int** getMin() {

**return** min\_stk.top();

    }

};

/\*\*

 \* Your MinStack object will be instantiated and called as such:

 \* MinStack obj = new MinStack();

 \* obj.push(x);

 \* obj.pop();

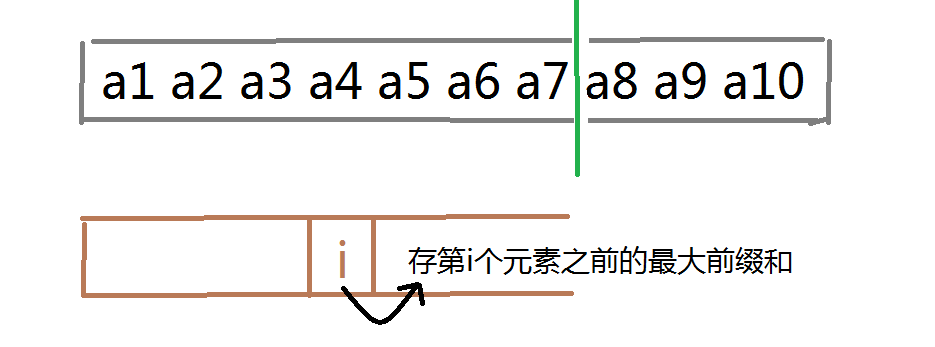
 \* int param\_3 = obj.top();

 \* int param\_4 = obj.getMin();

 \*/

### Editor （类似于“对顶堆”的“对顶栈”）

用两个栈来维护整个序列，两个栈类似于“对顶栈”，由光标分开。



另外对于询问前k个元素中最大的前缀和的操作，需要另外开一个新的栈，记录前i个元素前缀和的最大值。

代码：

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 1000010;

**int** stkl[maxn] , stkr[maxn] ,tl , tr;

**int** s[maxn] , f[maxn]; // s:前缀和  f:前i个前缀和中的最大值

**void** push\_left(**int** x){

    stkl[++tl] = x;

    s[tl] = s[tl - 1] + x;

    f[tl] = max(f[tl - 1] , s[tl]);

}

**int** main(){

**int** n;

**scanf**("%d",&n);

    f[0] = INT\_MIN;

**while**(n--){

**char** str[2];

**int** x;

**scanf**("%s",str);

**if**(str[0] == 'I'){

**scanf**("%d",&x);

            push\_left(x);

        }**else** **if**(str[0] == 'D'){

**if**(tl > 0) tl--;

        }**else** **if**(str[0] == 'L'){

**if**(tl > 0) stkr[++ tr] = stkl[tl--];

        }**else** **if**(str[0] == 'R'){

**if**(tr > 0) push\_left(stkr[tr--]);

        }**else**{

**scanf**("%d",&x);

**printf**("%d\n",f[x]);

        }

    }

**return** 0;

}

### 火车进出栈问题

1. 火车进栈

入栈按照1 2 3 ... n 入栈 ，要求输出字典序前20小的出栈序列

时间复杂度：O(2 ^ n)



#include <bits/stdc++.h>

**using** **namespace** std;

vector<**int**> res; //出栈序列

**int** now = 1; //当前该哪个入栈

stack<**int**> s;

**int** n , cnt = 20;

**void** dfs(){

**if**(!cnt) **return**;

**if**(res.size() == n){

**for**(**int** i = 0 ; i < n ; ++ i) cout << res[i];

        cout << endl;

        cnt--;

**return**;

    }

**if**(s.size()){

        res.push\_back(s.top());

        s.pop();

        dfs();

        s.push(res.back());

        res.pop\_back();

    }

**if**(now <= n){

        s.push(now);

        now++;

        dfs();

        now--;

        s.pop();

    }

}

**int** main(){

    cin >> n;

    dfs();

**return** 0;

}

1. 进出栈序列问题 (相当于求卡特兰数第n项)

（给定n，要求输出n个整数，有多少种出栈序列 n<=60000）

Catalan number ：1,2,5,14,42,132,429,1430,4862,16796 ......

**卡特兰数第n项 ： **

**代码：**

**暴力求会TLE**

**public** **class** Main{

**public** **static** **void** main(String[] args){

        Scanner cin = **new** Scanner(System.in);

**int** n = cin.nextInt();

        BigInteger ans = BigInteger.ONE;

**for**(**int** i = 2 \* n ; i > n ; -- i) ans = ans.multiply(BigInteger.valueOf(i));

**for**(**int** i = n + 1 ; i >= 2 ; -- i) ans = ans.divide(BigInteger.valueOf(i));

        System.out.println(ans);

    }

}

**正解：**所以要打个60000的素数表。先求出结果的每个质因数有多少次幂，然后用快速幂求，再把所有求幂得到的结果乘起来。

**import** java.util.Scanner;

**import** java.math.\*;

**public** **class** Main{

**static** Scanner cin = **new** Scanner(System.in);

**static** **final** **int** maxn = 200100;

**static** **int**[] prime = **new** **int**[maxn];

**static** **boolean**[] vis = **new** **boolean**[maxn];

**static** **int** cnt;//cnt记录1~60000共有多少个素数

**static** **void** init()

    {

        cnt = 0;

**for**(**int** i=2;i<maxn;i++)

        {

**if**(vis[i]==**false**)

                prime[cnt++] = i;

**for**(**int** j=0;j<cnt&&prime[j]\*i<maxn;j++)

            {

                vis[prime[j]\*i] = **true**;

**if**(i%prime[j]==0) **break**;

            }

        }

    }

**static** BigInteger qpow(**int** a,**int** b) //快速幂

    {

        BigInteger base = BigInteger.valueOf(a);

        BigInteger ret = BigInteger.ONE;

**while**(b!=0)

        {

**if**(b%2==1)

            {

                ret = ret.multiply(base);

            }

            base = base.multiply(base);

            b>>=1;

        }

**return** ret;

    }

**public** **static** **void** main(String args[])

    {

**int** n= cin.nextInt();

        init();

        BigInteger ans = BigInteger.ONE;

**int** times,v;

**for**(**int** i=0;i<cnt&&prime[i]<=n\*2;i++)

        {

            times = 0; //记录prime[i]要乘多少个

            v = n\*2;

**while**(v>0) //记录(2n)!要乘多少个prime[i]

            {

                times+=v/prime[i];

                v/=prime[i];

            }

            v = n;

//要减去上面多乘的n!的prime[i]还要除以n!的prime[i]，因此要乘2

**while**(v>0)

            {

                times -= v/prime[i]\*2;

                v/=prime[i];

            }

            ans = ans.multiply(qpow(prime[i],times));

        }

        BigInteger tmp  = BigInteger.valueOf(n+1); //最后除以n+1

        ans = ans.divide(tmp);

        System.out.println(ans);

    }

}

**技巧快速计算n!阶乘中素因子的个数：**

v = n;

**while**(v>0) //记录n!要乘多少个prime[i]

{

times+=v/prime[i];

v/=prime[i];

}

一个样例说明：

1 2 3 4 5 6 7 8 我们求得在8！中2的个数

  1   1   1   1 首先我们先计算出2的倍数的个数：8/2=4

      1       1 其次我们计算出4的倍数的个数： 8/4=2（上面一个式子求出了第一层，现在求第二层）

1 最后我们解出第三层的2的个数： 8/8=1

cnt(x)=[n/(x1)]+[n/(x2)]+[n/(x3)]+...(直到x的次方大于n)

### Largest Rectangle in a Histogram 直方图中最大的矩形 (单调栈 O(n) )

算法竞赛进阶指南书上的写法：

#include <bits/stdc++.h>

using namespace std;

**const** **int** maxn = 100005;

**int** a[maxn] , w[maxn];

**int** s[maxn]; //stack

**int** main(){

    ios::sync\_with\_stdio(**false**);

    cin.tie(0); cout.tie(0);

**int** n , top;

**while**(cin >> n){

**if**(n == 0) **break**;

**for**(**int** i = 1 ; i <= n ; ++ i) cin >> a[i];

        a[n + 1] = top = 0;

**long** **long** **int** ans = 0;

**for**(**int** i = 1 ; i <= n + 1 ; ++ i){

**if**(a[i] > s[top])

                s[++top] = a[i] , w[top] = 1;

**else**{

**int** width = 0;

**while**(s[top] > a[i]){

                    width += w[top];

                    ans = max(ans , 1ll \* width \* s[top]);

                    top--;

                }

                s[++top] = a[i] , w[top] = width + 1;

            }

        }

        cout << ans << endl;

    }

**return** 0;

}

另一种做法：

1. 找左边比当前a[i]小的第一个数的位置左边界left[i]
2. 右边比a[i]小的第一个位置右边界right[i]
3. 按照a[i]为矩形的高计算，更新答案。

#include <bits/stdc++.h>

using namespace std;

**const** **int** maxn = 100005;

**int** n;

**int** h[maxn] , Left[maxn] , Right[maxn] , s[maxn];

**void** get(**int** bound[maxn]){

**int** tt = 0;

    h[0] = -1;

**for**(**int** i = 1 ; i <= n ; ++ i){

**while**(h[s[tt]] >= h[i]) tt--;

        bound[i] = s[tt];

        s[++tt] = i;

    }

}

**int** main(){

**while**(cin >> n){

**if**(!n) **break**;

**for**(**int** i = 1 ; i <= n ; ++ i) cin >> h[i];

        get(Left);

        reverse(h + 1 , h + 1 + n);

        get(Right);

**long** **long** res = 0;

**for**(**int** i = 1 , j = n ; i <= n ; ++ i , -- j)

            res = max(res , 1ll \* h[i] \* (n - Left[j] - Right[i]));

        cout << res << endl;

    }

**return** 0;

}