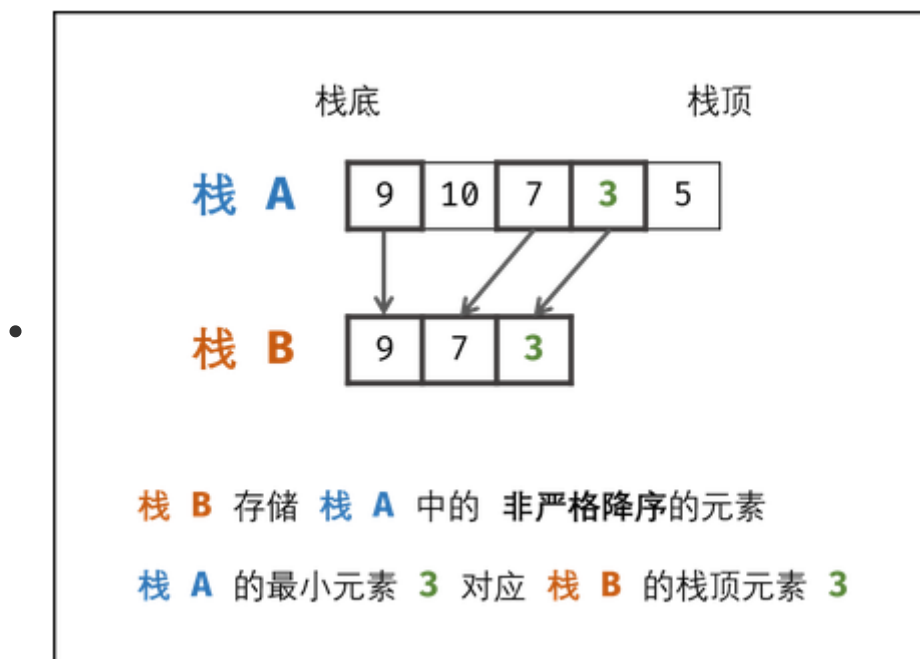


# 剑指Offer 30-包含min函数的栈

## 思路

### 辅助栈

- 两个栈 一个栈用于维护当前的最小值 另一个栈为常规栈
- 常规栈： 用于存储所有元素，保证入栈 `push()` 函数、出栈 `pop()` 函数、获取栈顶 `top()` 函数的正常逻辑。
- 最小栈： 存储常规栈中所有 **非严格降序** 的元素，则常规栈中的最小元素始终对应最小栈 的栈顶元素，即 `min()` 函数只需返回最小栈的栈顶元素即可。



## 代码

### 辅助栈

```
1 class MinStack {
2     private:
3         stack<int> mins,s;    //辅助栈：保存最小值 s为主栈
4     public:
5         /** initialize your data structure here. */
6         MinStack()
```

```

7      {
8          mins.push(INT_MAX);
9      }
10
11     void push(int x)
12     {
13         s.push(x);    //正常压栈
14         int temp=::min(x,mins.top());    //调用本类中的min
函数
15         mins.push(temp);
16     }
17
18     void pop() {
19         s.pop();
20         mins.pop();
21     }
22
23     int top()
24     {
25         return s.top();
26     }
27
28     int min()
29     {
30         return mins.top();
31     }
32 };
33
34 /**
35  * Your MinStack object will be instantiated and called
as such:
36  * MinStack* obj = new MinStack();
37  * obj->push(x);
38  * obj->pop();
39  * int param_3 = obj->top();
40  * int param_4 = obj->min();
41  */

```

# 单栈解决

```
1  class MinStack {
2  private:
3      stack<int> s;
4      int minV=INT_MAX;    //最小值
5  public:
6      /** initialize your data structure here. */
7      MinStack()
8      {
9      }
10
11     void push(int x)
12     {
13         s.push(minV);    //加入上一个最小值
14         if(x<minV) minV=x; //更新最小值
15         s.push(x);    //加入该数值
16
17     }
18
19     void pop() {
20         s.pop();    //pop出
21         minV = s.top(); //得到去掉该值后的最小值
22         s.pop();
23     }
24
25     int top()
26     {
27         return s.top();
28     }
29
30     int min()
31     {
32         return minV;
33     }
34 };
35
36 /**
```

```
37 | * Your MinStack object will be instantiated and called  
    | as such:  
38 | * MinStack* obj = new MinStack();  
39 | * obj->push(x);  
40 | * obj->pop();  
41 | * int param_3 = obj->top();  
42 | * int param_4 = obj->min();  
43 | */
```