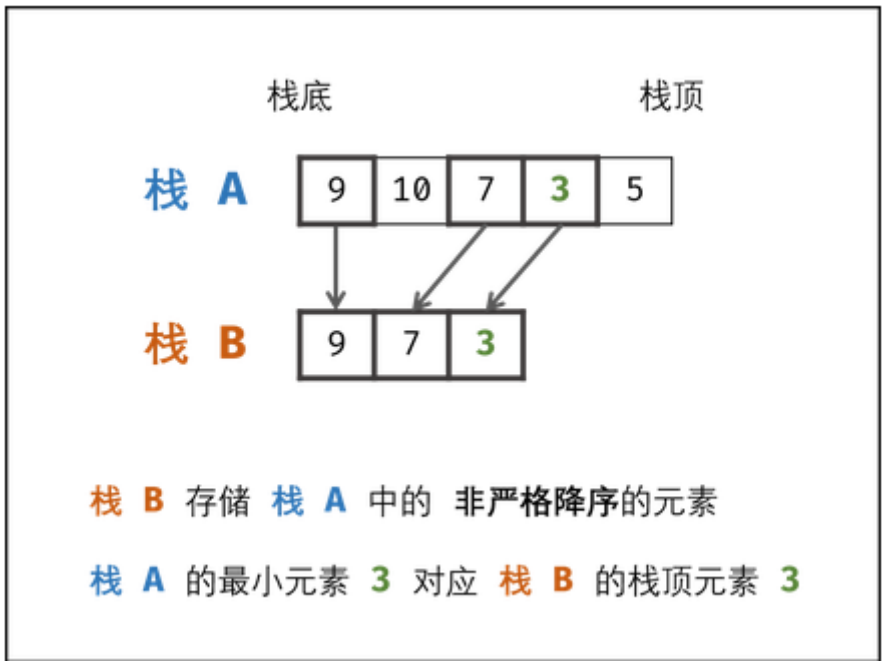
# 剑指Offer 30-包含min函数的栈

## 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

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

```
    {
        mins.push(INT_MAX);
    }

    void push(int x)
    {
        s.push(x);  //正常压栈
        int temp=::min(x,mins.top());   //调用本类中的min函数
        mins.push(temp);
    }

    void pop() {
        s.pop();
        mins.pop();
    }

    int top()
    {
        return s.top();
    }

    int min()
    {
        return mins.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->min();
 */
```

# 单栈解决

```
class MinStack {
private:
    stack<int> s;
    int minV=INT_MAX;    //最小值
public:
    /** initialize your data structure here. */
    MinStack()
    {
    }

    void push(int x)
    {
        s.push(minV);    //加入上一个最小值
        if(x<minV) minV=x;  //更新最小值
        s.push(x);  //加入该数值

    }

    void pop() {
        s.pop();    //pop出
        minV = s.top();  //得到去掉该值后的最小值
        s.pop();
    }

    int top()
    {
        return s.top();
    }

    int min()
    {
        return minV;
    }
};

/**
```

```
 * 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->min();
 */
```