【第十八周】单调栈与经典问题

sample

```
/*************************************************************************
  > File Name: sample.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

void output(vector<int> &arr, const char *msg) {
    printf("%s", msg);
    for (auto x : arr) {
        printf("%5d", x);
    }
    printf("\n");
    return ;
}

int main() {
    int n;
    cin >> n;
    vector<int> ind(n);
    vector<int> arr(n);
    vector<int> pre(n), nxt(n);
    stack<int> s;
    for (int i = 0; i < n; i++) ind[i] = i;
    for (int i = 0, a; i < n; ++i) cin >> arr[i];
    for (int i = 0; i < n; i++) {
        while (s.size() && arr[i] < arr[s.top()]) {
            nxt[s.top()] = i;
            s.pop();
        }
        if (s.size() == 0) pre[i] = -1;
        else pre[i] = s.top();
        s.push(i);
    }
    while (s.size()) nxt[s.top()] = n, s.pop();
    output(ind, "ind : ");
```

```
    output(arr, "now : ");
    output(pre, "pre : ");
    output(nxt, "nxt : ");
    return 0;
}
```

155. 最小栈

```
class MinStack {
public:
    /** initialize your data structure here. */
    stack<int> s, min_s;
    MinStack() {}

    void push(int val) {
        s.push(val);
        if (min_s.size() == 0 || val <= min_s.top()) {
            min_s.push(val);
        }
        return ;
    }

    void pop() {
        if (s.top() == min_s.top()) min_s.pop();
        s.pop();
        return ;
    }

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

    int getMin() {
        return min_s.top();
    }
};

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack* obj = new MinStack();
 * obj->push(val);
 * obj->pop();
 * int param_3 = obj->top();
 * int param_4 = obj->getMin();
 */
```

496. 下一个更大元素 I

```
class Solution {
public:
    vector<int> nextGreaterElement(vector<int>& nums1, vector<int>& nums2) {
        unordered_map<int, int> h;
        stack<int> s;
        for (auto x : nums2) {
            while (s.size() && x > s.top()) {
                h[s.top()] = x;
                s.pop();
            }
            s.push(x);
        }
        vector<int> ret(nums1.size());
        for (int i = 0; i < nums1.size(); i++) {
            if (h.find(nums1[i]) == h.end()) ret[i] = -1;
            else ret[i] = h[nums1[i]];
        }
        return ret;
    }
};
```

503. 下一个更大元素 II

```
class Solution {
public:
    vector<int> nextGreaterElements(vector<int>& nums) {
        stack<int> s;
        vector<int> ret(nums.size());
        for (int i = 0; i < nums.size(); i++) ret[i] = -1;
        for (int i = 0; i < nums.size(); i++) {
            while (s.size() && nums[i] > nums[s.top()]) {
                ret[s.top()] = nums[i];
                s.pop();
            }
            s.push(i);
        }
        for (int i = 0; i < nums.size(); i++) {
            while (s.size() && nums[i] > nums[s.top()]) {
                ret[s.top()] = nums[i];
                s.pop();
            }
            s.push(i);
        }
        return ret;
    }
};
```

901. 股票价格跨度

```
class StockSpanner {
public:
    typedef pair<int, int> PII;
    int t;
    stack<PII> s;
    StockSpanner() {
        t = 0;
        s.push(PII(INT_MAX, t++));
    }

    int next(int price) {
        while (s.size() && price >= s.top().first) s.pop();
        int ret = t - s.top().second;
        s.push(PII(price, t++));
        return ret;
    }
};

/**
 * Your StockSpanner object will be instantiated and called as such:
 * StockSpanner* obj = new StockSpanner();
 * int param_1 = obj->next(price);
 */
```

739. 每日温度

```
class Solution {
public:
    vector<int> dailyTemperatures(vector<int>& temperatures) {
        vector<int> ret(temperatures.size());
        stack<int> s;
        for (int i = 0; i < temperatures.size(); ++i) {
            while (s.size() && temperatures[i] > temperatures[s.top()]) {
                ret[s.top()] = i - s.top();
                s.pop();
            }
            s.push(i);
        }
        return ret;
    }
};
```

84. 柱状图中最大的矩形

```
class Solution {
public:
    int largestRectangleArea(vector<int>& heights) {
```

```
        stack<int> s;
        vector<int> l(heights.size()), r(heights.size());
        int n = heights.size();
        for (int i = 0; i < n; i++) l[i] = -1, r[i] = n;
        for (int i = 0; i < n; i++) {
            while (s.size() && heights[i] <= heights[s.top()]) {
                r[s.top()] = i;
                s.pop();
            }
            if (s.size()) l[i] = s.top();
            s.push(i);
        }
        int ans = 0;
        for (int i = 0; i < n; i++) {
            ans = max(ans, heights[i] * (r[i] - l[i] - 1));
        }
        return ans;
    }
};
```

42. 接雨水

```
class Solution {
public:
    int trap(vector<int>& height) {
        int ans = 0;
        stack<int> s;
        for (int i = 0; i < height.size(); i++) {
            while (s.size() && height[i] > height[s.top()]) {
                int now = s.top();
                s.pop();
                if (s.size() == 0) continue;
                int a = height[i] - height[now];
                int b = height[s.top()] - height[now];
                ans += (i - s.top() - 1) * min(a, b);
            }
            s.push(i);
        }
        return ans;
    }
};
```

456. 132 模式

```
class Solution {
public:
    bool find132pattern(vector<int>& nums) {
        vector<int> l(nums.size());
        l[0] = INT_MAX;
```

```
        for (int i = 1; i < nums.size(); i++) l[i] = min(l[i - 1], nums[i - 1]);
        stack<int> s;
        for (int i = nums.size() - 1; i >= 0; --i) {
            int val = nums[i];
            while (s.size() && nums[i] > s.top()) val = s.top(), s.pop();
            s.push(nums[i]);
            if (l[i] < nums[i] && val < nums[i] && val > l[i]) return true;
        }
        return false;
    }
};
```

907. 子数组的最小值之和

```
class Solution {
public:
    int sumSubarrayMins(vector<int>& arr) {
        stack<int> s;
        int mod_num = 1e9 + 7;
        long long ans = 0;
        vector<long long> sum(arr.size() + 1);
        sum[0] = 0;
        for (int i = 0; i < arr.size(); i++) {
            while (s.size() && arr[i] <= arr[s.top()]) s.pop();
            int ind = s.size() ? s.top() : -1;
            s.push(i);
            sum[s.size()] = (sum[s.size() - 1] + arr[i] * (i - ind)) % mod_num;
            ans += sum[s.size()];
            ans %= mod_num;
        }
        return ans;
    }
};
```

1856. 子数组最小乘积的最大值

```
class Solution {
public:
    int maxSumMinProduct(vector<int>& nums) {
        stack<int> s;
        vector<int> l(nums.size()), r(nums.size());
        int n = nums.size();
        for (int i = 0; i < n; i++) l[i] = -1, r[i] = n;
        for (int i = 0; i < n; i++) {
            while (s.size() && nums[i] <= nums[s.top()]) {
                r[s.top()] = i;
                s.pop();
            }
            if (s.size()) l[i] = s.top();
```

```
            s.push(i);
        }
        vector<long long> sum(n + 1);
        sum[0] = 0;
        for (int i = 0; i < n; i++) sum[i + 1] = sum[i] + nums[i];
        long long ans = 0;
        for (int i = 0; i < n; i++) {
            ans = max(ans, nums[i] * (sum[r[i]] - sum[l[i] + 1]));
        }
        return ans % (long long)(1e9 + 7);
    }
};
```