# 【第十七周】单调队列与经典问题

## HZOJ-271. 滑动窗口

```
/*************************************************************************
  > File Name: HZOJ271.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;

int main() {
    int n, k;
    vector<int> arr;
    cin >> n >> k;
    for (int i = 0, a; i < n; i++) {
        cin >> a;
        arr.push_back(a);
    }
    deque<int> q;
    for (int i = 0; i < n; i++) {
        while (q.size() && arr[q.back()] > arr[i]) q.pop_back();
        q.push_back(i);
        if (i - q.front() == k) q.pop_front();
        if (i + 1 < k) continue;
        if (i + 1 > k) cout << " ";
        cout << arr[q.front()];
    }
    cout << endl;
    q.clear();
    for (int i = 0; i < n; i++) {
        while (q.size() && arr[q.back()] < arr[i]) q.pop_back();
        q.push_back(i);
        if (i - q.front() == k) q.pop_front();
        if (i + 1 < k) continue;
        if (i + 1 > k) cout << " ";
        cout << arr[q.front()];
    }
    cout << endl;
    return 0;
}
```

## HZOJ372. 双生序列

```
/*************************************************************************
  > File Name: HZOJ372.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;

int main() {
    int n;
    cin >> n;
    vector<int> a(n), b(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    for (int i = 0; i < n; i++) cin >> b[i];
    deque<int> q1, q2;
    int p;
    for (p = 0; p < n; p++) {
        while (q1.size() && a[p] < q1.back()) q1.pop_back();
        while (q2.size() && b[p] < q2.back()) q2.pop_back();
        q1.push_back(a[p]);
        q2.push_back(b[p]);
        if (q1.size() != q2.size()) break;
    }
    cout << p << endl;
    return 0;
}
```

## 239. 滑动窗口最大值

```
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        deque<int> q;
        vector<int> ret;
        for (int i = 0; i < nums.size(); i++) {
            while (q.size() && nums[q.back()] < nums[i]) q.pop_back();
            q.push_back(i);
            if (i - q.front() == k) q.pop_front();
            if (i + 1 < k) continue;
            ret.push_back(nums[q.front()]);
        }
        return ret;
    }
};
```

## 剑指 Offer 59 - II. 队列的最大值

```
class MaxQueue {
public:
    deque<int> q, mq;
    MaxQueue() {}

    int max_value() {
        if (mq.size() == 0) return -1;
        return mq.front();
    }

    void push_back(int value) {
        q.push_back(value);
        while (mq.size() && value > mq.back()) mq.pop_back();
        mq.push_back(value);
        return ;
    }

    int pop_front() {
        if (q.size() == 0) return -1;
        if (q.front() == mq.front()) mq.pop_front();
        int ret = q.front();
        q.pop_front();
        return ret;
    }
};

/**
 * Your MaxQueue object will be instantiated and called as such:
 * MaxQueue* obj = new MaxQueue();
 * int param_1 = obj->max_value();
 * obj->push_back(value);
 * int param_3 = obj->pop_front();
 */
```

## 862. 和至少为 K 的最短子数组

```
class Solution {
public:
    int shortestSubarray(vector<int>& nums, int k) {
        deque<int> q;
        vector<int> sum(nums.size() + 1);
        sum[0] = 0;
        for (int i = 0; i < nums.size(); i++) sum[i + 1] = sum[i] + nums[i];
        q.push_back(0);
        int pos = -1, ans = -1;
        for (int i = 1; i < sum.size(); i++) {
            while (q.size() && sum[i] - sum[q.front()] >= k) {
                pos = q.front();
                q.pop_front();
            }
            if (pos != -1 && (i - pos < ans || ans == -1)) ans = i - pos;
            while (q.size() && sum[i] < sum[q.back()]) q.pop_back();
            q.push_back(i);
        }
        return ans;
    }
};
```

## 1438. 绝对差不超过限制的最长连续子数组

```
class Solution {
public:
    bool check(vector<int> &nums, int k, int limit) {
        deque<int> qmin, qmax;
        for (int i = 0; i < nums.size(); i++) {
            while (qmin.size() && nums[i] < nums[qmin.back()]) qmin.pop_back();
            while (qmax.size() && nums[i] > nums[qmax.back()]) qmax.pop_back();
            qmin.push_back(i);
            qmax.push_back(i);
            if (i + 1 < k) continue;
            if (i - qmin.front() == k) qmin.pop_front();
            if (i - qmax.front() == k) qmax.pop_front();
            if (nums[qmax.front()] - nums[qmin.front()] <= limit) return true;
        }
        return false;
    }
    int bs(vector<int> &nums, int l, int r, int limit) {
        if (l == r) return l;
        int mid = (l + r + 1) >> 1;
        if (check(nums, mid, limit)) l = mid;
        else r = mid - 1;
        return bs(nums, l, r, limit);
    }
    int longestSubarray(vector<int>& nums, int limit) {
        return bs(nums, 1, nums.size(), limit);
    }
};
```

## 513. 找树左下角的值

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
```

```
class Solution {
public:
    int max_k, val;
    void dfs(TreeNode *root, int k) {
        if (root == nullptr) return ;
        if (k > max_k) {
            max_k = k, val = root->val;
        }
        dfs(root->left, k + 1);
        dfs(root->right, k + 1);
        return ;
    }
    int findBottomLeftValue(TreeNode* root) {
        max_k = -1, val = 0;
        dfs(root, 0);
        return val;
    }
};
```

## 135. 分发糖果

```
class Solution {
public:
    int candy(vector<int>& ratings) {
        vector<int> l(ratings.size()), r(ratings.size());
        for (int i = 0, j = 1; i < l.size(); i++) {
            if (i && ratings[i] > ratings[i - 1]) j += 1;
            else j = 1;
            l[i] = j;
        }
        for (int i = r.size() - 1, j = 1; i >= 0; i--) {
            if (i < r.size() - 1 && ratings[i] > ratings[i + 1]) j += 1;
            else j = 1;
            r[i] = j;
        }
        int ans = 0;
        for (int i = 0; i < l.size(); i++) ans += max(l[i], r[i]);
        return ans;
    }
};
```

## 365. 水壶问题

```
class Solution {
public:
    typedef pair<int, int> PII;
    PII getNext(int k, int x, int X, int y, int Y) {
        switch (k) {
            case 0: return PII(0, y);
            case 1: return PII(x, 0);
            case 2: {
                int delta = min(x, Y - y);
                return PII(x - delta, y + delta);
            }
            case 3: {
                int delta = min(X - x, y);
                return PII(x + delta, y - delta);
            }
            case 4: return PII(X, y);
            case 5: return PII(x, Y);
        }
        return PII(0, 0);
    }
    struct HASH {
        long long operator()(const PII &a) const {
            return ((long long)(a.first) << 31) + a.second;
        }
    };
    bool canMeasureWater(int jug1Capacity, int jug2Capacity, int targetCapacity) {
        unordered_set<PII, HASH> vis;
        queue<PII> q;
        vis.insert(PII(0, 0));
        q.push(PII(0, 0));
        while (!q.empty()) {
            PII cur = q.front();
            if (cur.first + cur.second == targetCapacity) return true;
            q.pop();
            for (int i = 0; i < 6; i++) {
                PII temp = getNext(i, cur.first, jug1Capacity, cur.second, jug2Capacity);
                if (vis.find(temp) != vis.end()) continue;
                vis.insert(temp);
                q.push(temp);
            }
        }
        return false;
    }
};
```

## 1760. 袋子里最少数目的球

```
class Solution {
public:
    int f(vector<int> &nums, int x) {
        int cnt = 0;
        for (int i = 0; i < nums.size(); i++) {
            cnt += nums[i] / x + !!(nums[i] % x) - 1;
        }
        return cnt;
    }
    int bs(vector<int> &nums, int l, int r, int n) {
        if (l == r) return l;
        int mid = (l + r) >> 1;
        if (f(nums, mid) <= n) r = mid;
        else l = mid + 1;
        return bs(nums, l, r, n);
    }
    int minimumSize(vector<int>& nums, int maxOperations) {
        int l = 1, r;
        for (auto x : nums) r = max(r, x);
        return bs(nums, l, r, maxOperations);
    }
};
```

## 45. 跳跃游戏 II

```
class Solution {
public:
    int jump(vector<int>& nums) {
        if (nums.size() <= 1) return 0;
        int pre = 1, pos = nums[0], cnt = 1;
        while (pos + 1 < nums.size()) {
            int j = pre;
            for (int i = pre + 1; i <= pos; i++) {
                if (i + nums[i] > j + nums[j]) j = i;
            }
            pre = pos + 1, pos = j + nums[j];
            cnt += 1;
        }
        return cnt;
    }
};
```

## 93. 复原 IP 地址

```
class Solution {
public:
    void dfs(string &s, int k, int ind, vector<string> &ret) {
        if (ind >= s.size()) return ;
        if (k == 4) {
            int num = 0;
            if (s.size() - ind > 1 && s[ind] == '0') return ;
            for (int i = ind; i < s.size(); i++) {
                num = num * 10 + s[i] - '0';
                if (num > 255) return ;
            }
            ret.push_back(s);
            return ;
        }
        for (int i = ind, num = 0; i < s.size(); i++) {
            num = num * 10 + s[i] - '0';
            if (num > 255) return ;
            if (i - ind >= 1 && s[ind] == '0') return ;
            s.insert(i + 1, ".");
            dfs(s, k + 1, i + 2, ret);
            s.erase(i + 1, 1);
        }
        return ;
    }
    vector<string> restoreIpAddresses(string s) {
        vector<string> ret;
        dfs(s, 1, 0, ret);
        return ret;
    }
};
```

## 46. 全排列

```
class Solution {
public:
    vector<vector<int>> permute(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        vector<vector<int>> ret;
        do {
            ret.push_back(nums);
        } while (next_permutation(nums.begin(), nums.end()));
        return ret;
    }
};
```

## 43. 字符串相乘

```
class Solution {
public:
    string multiply(string num1, string num2) {
        vector<int> a(num1.size()), b(num2.size()), c(a.size() + b.size() - 1);
        for (int i = 0; i < num1.size(); i++) a[a.size() - i - 1] = num1[i] - '0';
        for (int i = 0; i < num2.size(); i++) b[b.size() - i - 1] = num2[i] - '0';
        for (int i = 0; i < a.size(); i++) {
            for (int j = 0; j < b.size(); j++) {
                c[i + j] += a[i] * b[j];
            }
        }
        for (int i = 0; i < c.size(); i++) {
            if (c[i] < 10) continue;
            if (i + 1 == c.size()) c.push_back(0);
            c[i + 1] += c[i] / 10;
            c[i] %= 10;
        }
        while (c.size() > 1 && c[c.size() - 1] == 0) c.pop_back();
        string ret = "";
        for (int i = c.size() - 1; i >= 0; i--) ret += c[i] + '0';
        return ret;
    }
};
```