

「集思广益一波」

CleanBlue

2022 年 11 月 16 日

目录

| | | |
|----------|---------------------|----------|
| 1 | 数据结构 | 1 |
| 1.1 | 单调队列/滑动窗口 | 1 |
| 1.2 | 单调栈 | 1 |
| 2 | 基本算法 | 1 |
| 2.1 | 二分 | 1 |

1 数据结构

1.1 单调队列/滑动窗口

有一个长为 n 的序列 a ，以及一个大小为 k 的窗口。现在这个从左边开始向右滑动，每次滑动一个单位，求出每次滑动后窗口中的最大值和最小值。

```

1 int n, k;
2 std::cin >> n >> k;
3
4 std::vector<int> a(n);
5 for (int &i : a) {
6     std::cin >> i;
7 }
8
9 std::deque<int> q;
10 for (int i = 0; i < n; ++i) { //输出最小值
11     while (!q.empty() && a[q.back()] >= a[i]) {
12         q.pop_back();
13     }
14     q.emplace_back(i);
15     if (i >= k - 1) {
16         while (!q.empty() && q.front() <= i - k)
17             q.pop_front();
18         std::cout << a[q.front()] << ' ';
19     }
20 }
21 std::cout << '\n';
22 while (!q.empty()) {
23     q.pop_back();
24 }
25
26 for (int i = 0; i < n; ++i) { //输出最大值
27     while (!q.empty() && a[q.back()] <= a[i]) {
28         q.pop_back();
29     }
30     q.emplace_back(i);
31     if (i >= k - 1) {
32         while (!q.empty() && q.front() <= i - k)
33             q.pop_front();
34         std::cout << a[q.front()] << ' ';
35     }
36 }
37 std::cout << '\n';
38
39 }
40

```

1.2 单调栈

输出右边第一个比 a_i 大的数的下标。

```

1 int n;
2 std::cin >> n;
3
4 std::vector<int> a(n), ans(n);
5 for (int &i : a) {
6     std::cin >> i;
7 }
8
9 std::stack<int> s;
10 for (int i = n - 1; i >= 0; --i) {
11     while (!s.empty() && a[s.top()] <= a[i]) {

```

```

12         s.pop();
13     }
14     if (s.empty()) {
15         ans[i] = -1;
16     } else {
17         ans[i] = s.top();
18     }
19     s.emplace(i);
20 }
21
22 for (int i = 0; i < n; ++i) {
23     std::cout << ans[i] + 1 << "\n";
24 }

```

2 基本算法

2.1 二分

```

1 int binary_search_1(const std::vector<int> &a,
2 int x) {
3     // a 单调递增
4     // 在 [0, n) 中查找 x 或 x 的后继的位置
5     int n = a.size(), l = 0, r = n;
6     while (l < r) {
7         int mid = (l + r) / 2;
8         if (a[mid] >= x) {
9             r = mid;
10        } else {
11            l = mid + 1;
12        }
13    }
14    return l;
15 }
16
17 int binary_search_2(const std::vector<int> &a,
18 int x) {
19     // a 单调递增
20     // 在 [0, n) 中查找 x 或 x 的前驱的位置
21     int n = a.size(), l = 0, r = n;
22     while (l < r) {
23         int mid = (l + r + 1) / 2;
24         if (a[mid] <= x) {
25             l = mid;
26        } else {
27            r = mid - 1;
28        }
29    }
30    return l;
31 }
32
33 double float_binary_search(/*...*/) {
34     constexpr double eps = 1e-7;
35     double l = 0, r = 1e18;
36     while (r - l > eps) {
37         double mid = (l + r) / 2;
38         if (/*...*/) {
39             l = mid;
40        } else {
41            r = mid;
42        }
43    }
44    return l;
45 }

```