数据结构

单调队列

▶ 题目

给定一个大小为 $n \leq 10^6$ 的数组。

有一个大小为 k 的滑动窗口，它从数组的最左边移动到最右边。

你只能在窗口中看到 k 个数字。

每次滑动窗口向右移动一个位置。

以下是一个例子：

该数组为 `[1 3 -1 -3 5 3 6 7]`，k 为 3。

窗口位置	最小值	最大值
[1 3 -1] -3 5 3 6 7	-1	3
1 [3 -1 -3] 5 3 6 7	-3	3
1 3 [-1 -3 5] 3 6 7	-3	5
1 3 -1 [-3 5 3] 6 7	-3	5
1 3 -1 -3 [5 3 6] 7	3	6
1 3 -1 -3 5 [3 6 7]	3	7

你的任务是确定滑动窗口位于每个位置时，窗口中的最大值和最小值。

输入格式

输入包含两行。

第一行包含两个整数 n 和 k，分别代表数组长度和滑动窗口的长度。

第二行有 n 个整数，代表数组的具体数值。

同行数据之间用空格隔开。

输出格式

输出包含两个。

第一行输出，从左至右，每个位置滑动窗口中的最小值。

第二行输出，从左至右，每个位置滑动窗口中的最大值。

输入样例：

```
8 3 1 3 -1 -3 5 3 6 7
```

输出样例：

```
-1 -3 -3 -3 3 3 3 5 5 6 7
```

AcWing 154. 滑动窗口

滑动窗口求最值的问题都是用单调队列来解决的

单调队列的元素从队尾**插入**，从队头**取出**，为保证整个队列的单调性，元素在队尾进行**调整**

也就是每次从对头取出的元素一定是整个单调队列当中的最值，而每次都会从队尾插入元素，与此同时也会在队尾调整元素

用代码描述这一整个过程（先后顺序不能乱）就是：

- 在队尾删除元素以保证整个队列的单调性
- 在队尾插入一个新的元素
- 在对头取出元素，此元素便是整个单调队列的最值

在滑动窗口求最值的问题中，需要在最前面加上队头元素是否会离开滑动窗口，但不管怎么样，上面的三个顺序不能变

关于单调队列的代码实现：

我们定义 hh 表示队头指针，tt 表示队尾指针，初始时 $hh=0,\ tt=-1$ ，队列当中存储的是各个元素的下标

判断队列是否为空：$hh \le tt$

队尾插入：$h[++tt]=k$

队头删去：$h++$

在保证队列单调性的部分，我们需要考虑的是删掉的元素与**当前遍历到的元素** $a[i]$ 之间的大小关系

如果我们期望队列严格单调递增，那么我们需要删去所有小于等于 $a[i]$ 的的元素

如果我们期望队列单调递增，那么需要删去所有小于 $a[i]$ 的元素，即队列当中允许存在等于 $a[i]$ 的元素

就本题而已，由于需要求的只是最大和最小值，因此有没有等号都可以

关于队头元素何时出队的问题，我们需要明确**单调队列中元素个数是多少**，即**单调队列最左边的数的下标是什么**

就本题来说，单调队列中只有 k 个数，因此最远的合法下标为 $i-k+1$ ，当 $q[hh]<i-k+1$ 时表示队头元素已经出队

完整代码：

```
#include <iostream>

using namespace std;

const int N = 1e6 + 10;
int a[N], q[N];
int n, k;

int main()
{
    cin >> n >> k;
    for(int i = 1; i <= n; i++) cin >> a[i];
    int hh = 0, tt = -1;
    for(int i = 1; i <= n; i++)
    {
        if(hh <= tt && i - q[hh] > k - 1) hh++;//判断队头元素是否离开窗口内
        while(hh <= tt && a[q[tt]] >= a[i]) tt--;//保证队列内元素严格递减，因此需要将所以大于等于a[i]全部删去
        q[++tt] = i;//先将元素插入到队列中，之后才能从队头取数
        if(i >= k) cout << a[q[hh]] << " ";//只有遍历到的数大于窗口长度，就可以输出
    }
    cout << endl;
    hh = 0, tt = -1;
    for(int i = 1; i <= n; i++)
    {
        if(hh <= tt && i - q[hh] > k - 1) hh++;
        while(hh <= tt && a[q[tt]] <= a[i]) tt--;
        q[++tt] = i;
        if(i >= k) cout << a[q[hh]] << " ";
    }
    return 0;
}
```

KMP