剑指Offer（六十四）：滑动窗口的最大值

2018年2月2日 10:39:09 2 9,906 °C 编辑

一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：牛客网

书籍下载：共享资源

二、题目

给定一个数组和滑动窗口的大小，找出所有滑动窗口里数值的最大值。例如，如果输入数组{2,3,4,2,6,2,5,1}及滑动窗口的大小3，那么一共存在6个滑动窗口，他们的最大值分别为{4,4,6,6,6,5}。

针对数组{2,3,4,2,6,2,5,1}的滑动窗口有以下6个：{[2,3,4],2,6,2,5,1}，{2,[3,4,2],6,2,5,1}，{2,3,[4,2,6],2,5,1}，{2,3,4,[2,6,2],5,1}，{2,3,4,2,[6,2,5],1}，{2,3,4,2,6,[2,5,1]}，如下图所示：

数组中的滑动窗口	滑动窗口中的最大值
[2, 3, 4], 2, 6, 2, 5, 1	4
2, [3, 4, 2], 6, 2, 5, 1	4
2, 3, [4, 2, 6], 2, 5, 1	6
2, 3, 4, [2, 6, 2], 5, 1	6
2, 3, 4, 2, [6, 2, 5], 1	6
2, 3, 4, 2, 6, [2, 5, 1]	5

1、思路

我们可以使用一个双端队列deque。

我们可以用STL中的deque来实现，接下来我们以数组{2,3,4,2,6,2,5,1}为例，来细说整体思路。

数组的第一个数字是2，把它存入队列中。第二个数字是3，比2大，所以2不可能是滑动窗口中的最大值，因此把2从队列里删除，再把3存入队列中。第三个数字是4，比3大，同样的删3存4。此时滑动窗口中已经有3个数字，而它的最大值4位于队列的头部。

第四个数字2比4小，但是当4滑出之后它还是有可能成为最大值的，所以我们把2存入队列的尾部。下一个数字是6，比4和2都大，删4和2，存6。就这依次进行，最大值永远位于队列的头部。

但是我们怎样判断滑动窗口是否包括一个数字？应该在队列里存入数字在数组里的下标，而不是数值。当一个数字的下标与当前处理的数字的下标之差大于或者相等于滑动窗口大小时，这个数字已经从窗口中滑出，可以从队列中删除。

整体过程示意图：

步骤	插入数字	滑动窗口	队列中的下标	最大值
1	2	2	0(2)	N/A
2	3	2, 3	1(3)	N/A
3	4	2, 3, 4	2(4)	4
4	2	3, 4, 2	2(4), 3(2)	4
5	6	4, 2, 6	4(6)	6
6	2	2, 6, 2	4(6), 5(2)	6
7	5	6, 2, 5	4(6), 6(5)	6
8	1	2, 5, 1	6(5), 7(1)	5

2、代码

C++：

```C+
class Solution {
public:
    vector<int> maxInWindows(const vector<int>& num, unsigned int size)
    {
        vector<int> maxInWindows;
        // 数组大小要大于等于窗口大小，并且窗口大小大于等于1
        if(num.size() >= size && size >= 1){
            deque<int> index;
            for(unsigned int i = 0; i < size; i++){
                // 如果index非空，并且新添加的数字大于等于队列中最小的数字，则删除队列中最小的数字
                while(!index.empty() && num[i] >= num[index.back()]){
                    index.pop_back();
                }
                index.push_back(i);
            }
            for(unsigned int i = size; i < num.size(); i++){
                maxInWindows.push_back(num[index.front()]);
                while(!index.empty() && num[i] >= num[index.back()]){
                    index.pop_back();
                }
                // 控制窗口大小为size
                if(!index.empty() && index.front() <= int(i - size)){
                    index.pop_front();
                }
                index.push_back(i);
            }
            maxInWindows.push_back(num[index.front()]);
        }
        return maxInWindows;
    }
};
```