滑动窗口的最大值

# 滑动窗口的最大值--剑指Offer

## 题目描述

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]}。

## 思路分析

### 总体思路：基于一个双端队列Deque实现滑动窗口SlidingWindow；首先对滑动窗口初始化，然后在每个滑动窗口中，队列头部始终保存最大值的索引，这样每次取出队列头部元素即可获取当前滑动窗口的最大值；保持从头部到尾部是索引对应值从大到小的顺序：实现方法while循环从尾部不断取值，若小于等于当前值，则从尾部删除该元素，直到尾部元素大于当前值或队列为空；需要判断头部元素是否溢出窗口，若溢出则删除该元素；最后无论如何，都需要把当前值插入到队列尾部。

**问题1**：为什么无论如何把当前值插入到队列尾部？

答：<因为当前值属于**最年轻的元素**，在前面元素都溢出窗口的时候，有可能是后续窗口的最大值，所以无论如何都要插入到队列中，只是因为年轻。>

**问题2**：为什么队列中保存索引值而不直接保存元素值？

**需要对头部元素是否溢出窗口进行判断**：利用当前索引与队列头部索引之差判断队列头部元素是否溢出窗口，若溢出窗口，则删除头部元素即可

### 具体实现分成两部分：(1).欢滑动窗口初始化；(2).逐步移动滑动窗口。

\* **首先对滑动窗口初始化**：while循环，队列非空且从队列尾部不断取值，若比当前值小，就从尾部删除，直到队列为空或队列值大于当前值；循环之后，添加当前值至队列尾部；

\* **然后逐步移动滑动窗口**：首先将队列头部元素保存至结果list中，然后同初始化操作一样，从尾部逐渐取值判断，若比当前值小，则从尾部删除该值，直到队列为空或队列值大于当前值；**需要对头部元素是否溢出窗口进行判断**； 最后无论如何都要从尾部插入当前值，只是因为年轻。

\* 最终返回结果list即可。

## Java代码

public ArrayList<Integer> **maxInWindows**(int[] nums, int size){

ArrayList<Integer> result = new ArrayList<Integer>();

if(nums == null||nums.length == 0 ||size <= 0 || nums.length < size) return result;

ArrayDeque<Integer> deque = new ArrayDeque<Integer>();//双端队列

//初始化滑动窗口

for(int index = 0;index < size;index++){

while(!deque.isEmpty()&&nums[index] >= nums[deque.peekLast()])

deque.removeLast();//从队列尾部逐渐清理比当前值小的元素

deque.addLast(index);//保存最新的索引

}

**//滑动窗口逐步移动**

for(int index = size;index < nums.length;index++){

result.add(nums[deque.peekFirst()]);//保存当前窗口的最大值

while(!deque.isEmpty()&&nums[index] >= nums[deque.peekLast()])

deque.removeLast();//从队列尾部逐渐清理比当前值小的元素

if(!deque.isEmpty()&&index - deque.peekFirst()>=size)

deque.removeFirst();//删除溢出的队列头部元素

deque.addLast(index);//保存最新的索引

}

**result.add(nums[deque.peekFirst()]);**//保存最后一个滑动窗口的最大值

return result;

}

测试：

@Test

public void test(){

int[] nums = {2,3,4,2,6,2,5,1};

int size = 3;

ArrayList<Integer> result = maxInWindows(nums,size);

System.out.println(result.toString());

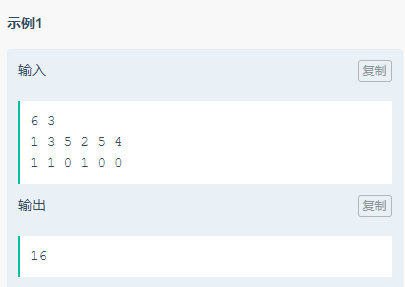
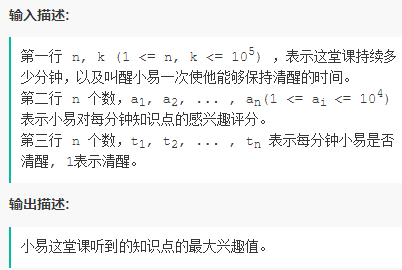
}

结果：[4, 4, 6, 6, 6, 5]

# 瞌睡问题

## 问题描述

小易觉得高数课太无聊了，决定睡觉。不过他对课上的一些内容挺感兴趣，所以希望你在老师讲到有趣的部分的时候叫醒他一下。你知道了小易对一堂课每分钟知识点的感兴趣程度，并以分数量化，以及他在课堂上每分钟是否会睡着，你可以叫醒他一次，这会使得他在接下来的k分钟内保持清醒，你需要选择一种方案最大化小易这堂课听到的知识点的分值。



## 思路分析

首先计算所有清醒时间的总分数，然后初始话第一个滑动窗口，计算当前窗口的总分数；

然后逐步滑动窗口，通过总分数的变化值是否大于0，决定当前是否需要更新总分数。

## Java代码

/\*\*

\* 思路分析：利用滑动窗口解决

\* @param n 一堂课的分钟数

\* @param k 叫醒一次清醒分钟数

\* @param perValue 每分钟的分数

\* @param isSleep 每分钟是否睡着

\* @return 返回最大的总分数

\*/

public int maxSumValue(int n,int k,int[] perValues,int[] isSleep){

int maxSumValue = 0;

**//计算所有清醒的总分数**

**for(int i = 0;i < n ;i++)**

**maxSumValue += perValues[i]\*isSleep[i];**

**//初始化第一个窗口**

**for(int i = 0;i < k; i++)**

**maxSumValue += perValues[i]\*(1-isSleep[i]);**

for(int i = k;i < n;i++){

int add = perValues[i]\*(1-isSleep[i]) - perValues[i-k]\*(1-isSleep[i-k]);

**if(add > 0) maxSumValue += add;**

}

return maxSumValue;

}

# 队列的最大值<利用滑动窗口保存最大值>

## 问题描述

定义一个队列并实现函数max得到队列里的最大值，要求函数max、push\_back和pop\_front的时间复杂度都是O(1)。

## 思路分析

利用两个队列实现。一个对列存放元素，另一个队列中实现滑动窗口的功能。

## Java代码