剑指Offer（三十）：连续子数组的最大和

2017年12月25日 11:13:16 发表评论 4,734 °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

HZ偶尔会拿些专业问题来忽悠那些非计算机专业的同学。今天测试组开完会后,他又发话了:在古老的一维模式识别中,常常需要计算连续子向量的最大和当向量全为正数的时候,问题很好解决。但是,如果向量中包含负数,是否应该包含某个负数,并期望旁边的正数会弥补它呢？例如:{6,-3,-2,7,-15,1,2,2},连续子向量的最大和为8(从第0个开始,到第3个为止)。你会不会被他忽悠住？(子向量的长度至少是1)

1、思路

数组分析：下图是我们计算数组（1，-2，3，10，-4，7，2，-5）中子数组的最大和的过程。通过分析我们发现，累加的子数组和，如果大于零，那么我们继续累加就行；否则，则需要剔除原来的累加和重新开始。

过程如下：

步骤	操作	累加的子数组和	最大的子数组和
1	加 1	1	1
2	加-2	-1	1
3	抛弃前面的和-1，加 3	3	3
4	加 10	13	13
5	加-4	9	13
6	加 7	16	16
7	加 2	18	18
8	加-5	13	18

2、代码

C++：

```C++
class Solution {
public:
    int FindGreatestSumOfSubArray(vector<int> array) {
        if(array.empty()){
            return 0;
        }
        // 初始化变量，maxSum为最大和，curSum为当前和
        int maxSum = array[0];
        int curSum = array[0];
```

```
        // 遍历所有元素
        for(int i = 1; i < array.size(); i++){
            // 如果当前和小于等于0，说明之前的是负数，则抛弃前面的和，重新计算
            if(curSum <= 0){
                curSum = array[i];
            }
            // 如果没有问题，直接累加
            else{
                curSum += array[i];
            }
            // 更新最大和
            if(curSum > maxSum){
                maxSum = curSum;
            }
        }
        return maxSum;
    }
};
```

Python：

```
# -*- coding:utf-8 -*-
class Solution:
    def FindGreatestSumOfSubArray(self, array):
        # write code here
        if len(array) == 0:
            return 0
        maxSum = array[0]
        curSum = array[0]

        for each in array[1:]:
            if curSum <= 0:
                curSum = each
            else:
                curSum += each
            if curSum > maxSum:
                maxSum = curSum
        return maxSum
```


不要欺骗别人，能被你骗到的都是相信你的人。--- 乔布斯