链滴

PAT 甲级刷题实录——1007

作者：aopstudio
原文链接：https://ld246.com/article/1579078527522
来源网站：链滴

原题链接

https://pintia.cn/problem-sets/994805342720868352/problems/994805514284679168

寻找解法

这题一开始有点懵，一开始以为是选个固定长度的最大子串，那样会简单很多。但并不是，子串的长是没有限制的，于是复杂度一下子上去了。自己之前没有遇到过这样的问题，于是上网搜索了别人的法。说实话，有一些解法我是想都不敢去想的，也佩服有些人还真敢用，比如这篇文章的：https://blg.csdn.net/weixin_42267752/article/details/82414578

这个算法完全就是硬算，时间复杂度高到爆炸，亏他还好意思说是大神

之后又参考了这篇文章的：https://www.cnblogs.com/Breathmint/p/10275935.html

这篇文章的水准明显就高了，首先他没有用数组，而是在输入数据的同时就进行计算，一下子就把空复杂度降到了O(1)。另外他的算法也非常精妙，当数据输入完毕时结果也已经计算出来。大家可以看篇文章里的思路，我的算法对这篇文章的算法做了一点小小的改进，代码如下，大家可以同时参考一。

代码

```
#include <iostream>
using namespace std;
int main()
{
    int num;    //数字个数
    int currentSum=0;    //记录子串之和
    int maxSum = -1;    //最大子串之和
    int subFirst, subLast;    //记录最大子串的第一个和最后一个数字
    int startFlag;    //目前计算的子串的首元素
    bool updateFlag = false;    //用来记录是否在下一次循环中更新startFlag
    int first, last;    //记录整个数字串的第一个和最后一个数字
    int currentNum;    //记录当前读取的数字
    cin >> num;
    for (int i = 0; i < num; i++)
    {
        cin >> currentNum;
        if (i == 0)
            subFirst = subLast = startFlag = first = currentNum;    //初始化并确定first
        if (i == num-1)
            last = currentNum;
        currentSum += currentNum;    //计算目前计算的子串之和
        if (updateFlag == true)    //根据上一次循环的标记更新startFlag
        {
            startFlag = currentNum;
            updateFlag = false;
        }
        if (currentSum > maxSum)    //目前计算的子串之和大于已知的最大子串和，则更新最大子串，同时更新最大子串的首元素和尾元素
        {
            subFirst = startFlag;    //更新最大子串的首元素
            subLast = currentNum;    //更新最大子串的尾元素
```

```
            maxSum = currentSum;    //更新最大子串和
        }
        if (currentSum < 0)    //当前计算的子串之和小于0，则重新从零开始计算子串之和，同时标记
一次循环中更新startFlag
        {
            currentSum = 0;        //重新从零开始计算currentSum
            updateFlag = true;        //记录下一次循环中更新startFlag
        }
    }
    if (maxSum == -1)    //全是负数
    {
        cout << 0 << ' ' << first << ' ' << last;
    }
    else
    {
        cout << maxSum << ' ' << subFirst << ' ' << subLast;
    }
    return 0;
}
```

思路以及改进

在运算过程中，子串分为最大子串和目前计算的子串。算法的基本思路为：每次都将本轮循环读取到元素加到目前计算的子串之和中。当目前计算的子串之和大于已知的最大子串之和时，就要更新最大串之和以及最大子串的首元素和尾元素，其中首元素更新为之前已经记录的目前计算的子串的首元素下文会说何时记录），尾元素为当前读取的元素。因为最大子串之和不能为负数，所以每次当目前计的子串之和<0时，就要重新将目前计算的子串之和从0开始计算，另外将目前计算的子串的首元素设为下一轮循环读到的元素（这里需要读者细细地理解一下）。

原文章的代码中设置了一个标记为leftFlag，并将它设为0代表下一轮循环更新目前计算的子串的首元。但我认为这样不妥，因为在原文章的代码中leftFlag既承担了判断是否更新目前计算的子串的首元的作用，也承担了存储目前计算的子串的首元素的作用，而如果数字串中读取到0元素赋给leftFlag，会干扰到目前计算的子串的首元素的更新。我的改进是另外设一个bool变量，用true代表更新leftFla。同时我发现原文章的代码中rightFlag并没有起到作用，因此我就删减了这个变量，并把leftFlag重命名为startFlag。