PAT_A1007.Maximum Subsequence Sum (25)

1. Abstraction

1.1 Method and idea

```
1.动态规划
2.two pointer
```

1.2 Notice

```
1.题目中明确指出，全部为零的序列是一种特殊情况，所以需要单独处理。
```

2. Problem

Given a sequence of K integers { N1, N2, ..., NK }. A continuous subsequence is defined to be { Ni, Ni+1, ..., Nj } where 1 <= i <= j <= K. The *Maximum Subsequence* is the continuous subsequence which has the largest sum of its elements. For example, given sequence { -2, 11, -4, 13, -5, -2 }, its maximum subsequence is { 11, -4, 13 } with the largest sum being 20.

Now you are supposed to find the largest sum, together with the first and the last numbers of the maximum subsequence.

Input Specification:

Each input file contains one test case. Each case occupies two lines. The first line contains a positive integer K (<= 10000). The second line contains K numbers, separated by a space.

Output Specification:

For each test case, output in one line the largest sum, together with the first and the last numbers of the maximum subsequence. The numbers must be separated by one space, but there must be no extra space at the end of a line. In case that the maximum subsequence is not unique, output the one with the smallest indices i and j (as shown by the sample case). If all the K numbers are negative, then its maximum sum is defined to be 0, and you are supposed to output the first and the last numbers of the whole sequence.

Sample Input:

```
10
-10 1 2 3 4 -5 -23 3 7 -21
```

Sample Output:

```
10 1 4
```

3. Code

3.1 Edit 0

3.1.1 Algorithm abstraction

```
1.按位置求出和序列。
2.从首尾两个位置开始向中间遍历。
```

3.1.2 Notice

```
* None
```

```
#include <cstdio>
#include <algorithm>
#include <vector>
using std::vector;
using std::fill;

int main(){
    const int maxn=10010;
    int num=0;
    int sum[maxn];
    fill(sum,sum+maxn,0);
    scanf("%d",&num);
    int temp;
    for(int i=1;i<=num;++i){
        scanf("%d",&temp);
        sum[i]=temp+sum[i-1];
    }


    int max_sum=0;
```

```
    int index_left=-1;
    int index_right=-1;
    int sum_i_j=1+num;
    int temp_sum=0;
    for(int i=num;i>0;--i){
        for(int j=0;j<i;++j){
            temp_sum=sum[i]-sum[j];
            if(temp_sum>max_sum){
                max_sum=temp_sum;
                index_left=j;
                index_right=i;
                sum_i_j=i+j;
            }
            else if(temp_sum==max_sum){
                if(i+j<sum_i_j){
                    index_left=j;
                    index_right=i;
                }
            }
        }
    }
    if(index_left==-1){
        printf("%d %d %d",max_sum,sum[1],sum[num]-sum[num-1]);
    }
    else{
        printf("%d %d %d",max_sum,sum[index_left+1]-sum[index_left],sum[index_right]-
sum[index_right-1]);
    }
    return 0;
}
```

3.2 Edit 1:

3.2.1 Algorithm abstraction

```
1.动态规划，数组dp[i]用来记录以第i个数字结尾的序列的最大子序列和
2.单独建立一个数组用来统计序列的起始位置
```

3.2.2 Notice

```
1.需要注意序找序列起始位置的方法，有的方法可能会造成某些测试点无法通过，目前位找到原因
```

错误代码:

```
#include <iostream>
#include <vector>
#include <algorithm>
using std::fill;
using std::cin;
using std::cout;
```

```
using std::endl;
using std::vector;

const int maxn=10010;
int dp[maxn];
int state[maxn];
void init(int *dp,int len,int init){
    fill(dp,dp+len,init);
    return;
}

int main(){
    init(dp,maxn,0);
    init(state,maxn,0);
    vector<int> array;
    int count;
    int temp;
    cin>>count;
    int flag=false;
    for(int i=0;i<count;++i){
        cin>>temp;
        if(temp>=0){
            flag=true;
        }
        array.push_back(temp);
    }
    if(flag==false){
        cout<<0<<" "<<*array.begin()<<" "<<*(array.end()-1)<<endl;
        return 0;
    }
    dp[0]=array[0];
    for(int i=1;i<count;++i){
        if(dp[i-1]<=0){
            dp[i]=array[i];
        }
        else{
            dp[i]=dp[i-1]+array[i];
        }
    }
    int max=-99999999;
    int end_lo=-1;
    for(int i=0;i<count;++i){
        if(max<dp[i]){
            max=dp[i];
            end_lo=i;
        }
    }
    temp=end_lo;
    while(max!=0&&temp>=0){
        max-=array[temp--];
    }
    cout<<dp[end_lo]<<" "<<array[temp+1]<<" "<<array[end_lo]<<endl;

    return 0;
```

```
}
```

正确代码:

```
#include <iostream>
#include <vector>
#include <algorithm>
using std::fill;
using std::cin;
using std::cout;
using std::endl;
using std::vector;

const int maxn=10010;
int dp[maxn];
int state[maxn];
void init(int *dp,int len,int init){
    fill(dp,dp+len,init);
    return;
}

int main(){
    init(dp,maxn,0);
    init(state,maxn,0);
    vector<int> array;
    int count;
    int temp;
    cin>>count;
    int flag=false;
    for(int i=0;i<count;++i){
        cin>>temp;
        if(temp>=0){
            flag=true;
        }
        array.push_back(temp);
    }
    if(flag==false){
        cout<<0<<" "<<*array.begin()<<" "<<*(array.end()-1)<<endl;
        return 0;
    }
    dp[0]=array[0];
    for(int i=1;i<count;++i){
        if(dp[i-1]<=0){
            dp[i]=array[i];
            state[i]=i;
        }
        else{
            dp[i]=dp[i-1]+array[i];
            state[i]=state[i-1];
        }


    }
```

```
    int k=0;
    for(int i=1;i<count;++i){
        if(dp[i]>dp[k]){
            k=i;
        }
    }

    cout<<dp[k]<<" "<<array[state[k]]<<" "<<array[k]<<endl;
    return 0;
}
```