☑ PAT 甲级题目讲解：1007《Maximum Subsequence Sum》

🧩 题目简介

本题是经典的**最大连续子序列和**（Maximum Subsequence Sum）问题。

若存在多个最大和相同的子序列，请输出**最先出现的**那个，也就是起点索引 `i` 最小的子序列；
如果起点相同，则终点索引 `j` 更小的子序列优先（即“最小字典序优先”）。

给定一个长度为 K 的整数序列 $\{N_1, N_2, \ldots, N_K\}$，要求找出**和最大的连续子序列**，并输出：

- 该子序列的**最大和**；
- 子序列的**第一个元素值**；
- 子序列的**最后一个元素值**。

特殊约定：

- 若所有数均为负，则最大子序列和为 `0`，同时输出整个序列的首尾元素。

🧪 样例分析

输入样例：

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

观察整个序列：

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

手动分析所有可能的子序列：

- 存在两个最大和相同的子序列：`1 2 3 4` 和 `3 7`，其和均为 `10`；
- 根据题目要求：选择 **字典序最小** 的那个，即起点较小的；
  - `1 2 3 4` 起点在下标 1，终点下标 4；
  - `3 7` 起点下标 7，虽更短但下标更大，**不选**。

因此，最终输出：

```
10 1 4
```

🔍 解题思路

本题核心是动态规划的思想，即使用 **前缀局部最优** 信息，逐步更新整体最优。

🧠 Kadane 算法核心思想

Kadane 算法解决的问题是： **如何在线性时间内找出连续子序列的最大和?**

它的核心思想是：

- 对于当前遍历到的元素 a_i，我们维护一个**当前连续子序列的和** `s`；
  - 如果 `s` < 0，说明前面的累计对后续结果有害，应从当前位置 `i` 重新开始新的子序列；
  - 否则，继续将当前元素加入子序列；
- 每次迭代后，都尝试更新全局最大子序列和 `maxs`。

☑ 数学表达

假设当前遍历到位置 i，我们有：

- 若前缀和 $s < 0$，则：

$$s := a_i$$

- 否则继续累加：

$$s := s + a_i$$

然后更新最大值：

$$\text{maxs} := \max(\text{maxs}, s)$$

📌 为什么要重启?

因为：

- 如果当前的累计和是负数，继续加上后续数字只会让总和更小；
- 所以遇到负和时，**果断重启子序列更优。**

🎯 基本策略：

- 用一个变量 `s` 记录当前连续子序列的和；
- 如果当前和 `s` 小于 `0`，则重置从当前位置重新开始；
- 每次更新时判断是否比历史最大和更优。

📎 变量说明

变量名	含义
`k`	输入整数序列长度
`a[]`	原始整数序列
`s`	当前子序列的累加和
`tp`	当前子序列起点索引
`p`	最终答案中子序列起点索引
`q`	最终答案中子序列终点索引
`maxs`	当前为止的最大子序列和
`f`	标记是否存在非负数（用于特殊判断）

☑ Step 1：输入与特判处理

```
scanf("%d", &k);
for(int i = 0; i < k; i++){
    scanf("%d", &a[i]);
    if(a[i] >= 0) f = 1;  // 存在非负数
}
if(!f){
    // 所有数为负，直接输出规则要求的形式
    printf("0 %d %d", a[0], a[k - 1]);
    return 0;
}
```

☑ Step 2：Kadane 算法主过程

采用经典的最大子序列求和算法，每次遍历更新最大和并记录位置：

```
maxs = INT_MIN;  // 初始化为最小整数
for(int i = 0; i < k; i++){
    if(s < 0){  // 当前子序列的累加和为负值
        tp = i;       // 当前位置作为新的起点
        s = a[i];     // 重置当前和
    }
    else{
        s += a[i];    // 累加当前值
    }
    if(s > maxs){
        maxs = s;     // 更新最大和
        p = tp;       // 更新起点
        q = i;        // 更新终点
    }
}
```

☑ Step 3：输出结果

最终输出的是最大和 + 起点值 + 终点值：

```
printf("%d %d %d", maxs, a[p], a[q]);
```

☑ 完整代码

```
#include <bits/stdc++.h>
using namespace std;

int k, a[10005], s, tp, p, q, maxs;
bool f;

int main(){
    scanf("%d", &k);
    for(int i = 0; i < k; i++){
        scanf("%d", &a[i]);
        if(a[i] >= 0) f = 1;
    }
    if(!f){
        printf("0 %d %d", a[0], a[k - 1]);
        return 0;
    }
    maxs = INT_MIN;
    for(int i = 0; i < k; i++){
        if(s < 0){
            tp = i; // 更新序列起点
            s = a[i];
        }
        else{
            s += a[i];
        }
        if(s > maxs){
            maxs = s;
            p = tp;
            q = i;
        }
    }
    printf("%d %d %d", maxs, a[p], a[q]);
    return 0;
}
```

常见错误提醒

错误类型	错误表现
所有元素为负时处理错误	忘记特判，导致输出错误或数组越界
`maxs` 初始值不当	考虑到只有负数和 0 的情况，不能初始化为 0， -1 或者 `INT_MIN` 正确
子序列起点更新不当	忘记在 `s < 0` 时更新 `tp`，导致区间定位错误
最大和判断顺序错	应在每次 `s > maxs` 时更新序列起始位置与和
起点与终点输出的是索引	应输出对应值 `a[p]`，`a[q]` 而非下标

☑ 总结归纳

- **本题核心**是经典的 Kadane 算法；
- 在遍历时使用 `s < 0` 作为重启标志；
- 特判所有为负数的情况；
- 同时跟踪子序列的起点和终点，方便输出原始数值。

⏱ 复杂度分析

- 时间复杂度：$\mathcal{O}(n)$
- 空间复杂度：$\mathcal{O}(n)$（由于存储输入序列）

🧠 思维拓展

- 若要求输出所有等价最大子序列，如何存储多个答案？
- 若改为“最大乘积子序列”，应如何调整？
- 本题也可以作为滑动窗口 + 动态规划的变种训练。