P3817 小A的糖果

题目描述

小A有 n 个糖果盒，第 i 个盒中有 ai 颗糖果。

小 A 每次可以从其中一盒糖果中吃掉一颗，他想知道，要让任意两个相邻的盒子中糖的个数之和都不大于 x，至少得吃掉几颗糖。

输入格式

输入的第一行是两个用空格隔开的整数，代表糖果盒的个数 n 和给定的参数 x。

第二行有 n 个用空格隔开的整数，第 i 个整数代表第 i 盒糖的糖果个数 ai。

输出格式

输出一行一个整数，代表最少要吃掉的糖果的数量。

输入样例

```
3 3
2 2 2
```

输出样例

```
1
```

解析

核心问题：当相邻的两盒大于x到时候，应该先吃那一盒呢？

答案：正着遍历，吃后面；反着遍历，吃前面！

为什么是这样的呢？

比如一个样例：

```
5 6
4 5 3 6 2
```

此时，若正着遍历，4+5>6，如果吃4（a[1]），就会忽略3(a[3])，吃5(a[2])则一举两得。

编码

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
const int Maxn = 1e5 + 1;

long long n, x;
//吃的总糖数
long long countNum = 0;
//糖的数组
int sugar[Maxn];

int main() {
    scanf("%d %d", &n, &x);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &sugar[i]);
    }
    //处理首个糖盒就超过目标数量的情况
    if (sugar[0] > x) {
        int gap = sugar[0] - x;
        countNum += gap;
        sugar[0] -= gap;
    }
    //处理剩余糖盒，全部是优先吃后面的糖
    for (int i = 1; i < n; i++) {
        //求出相邻之和
        long long sum = sugar[i] + sugar[i - 1];
        //求出需要吃掉的数量
        if (sum > x) {
            long long gap = sum - x;
            //记录已经吃的数量
            countNum += gap;
            //把相邻的豆子减掉
            sugar[i] -= gap;
        }
    }
    cout << countNum;
    return 0;
}
```