1243：月度开销

题目描述

农夫约翰是一个精明的会计师。他意识到自己可能没有足够的钱来维持农场的运转了。他计算出并记录下了接下来 N (1 ≤ N ≤ 100,000) 天里每天需要的开销。

约翰打算为连续的M (1 ≤ M ≤ N) 个财政周期创建预算案，他把一个财政周期命名为fajo月。每个fajo月包含一天或连续的多天，每天被恰好包含在一个fajo月里。

约翰的目标是合理安排每个fajo月包含的天数，使得开销最多的fajo月的开销尽可能少。

输入

第一行包含两个整数N,M，用单个空格隔开。

接下来N行，每行包含一个1到10000之间的整数，按顺序给出接下来N天里每天的开销。

输出

一个整数，即最大月度开销的最小值。

输入样例

```
7 5
100
400
300
100
500
101
400
```

输出样例

```
500
```

提示

若约翰将前两天作为一个月，第三、四两天作为一个月，最后三天分别作为一个月，则最大月度开销为500。其他任何分配方案都会比这个值更大。

解析

这道题的意思就是尽可能的让每个财务月的支出相对平衡，大家学会了就可以来做账了。怎么样，是不是又感觉学会了一门技术？

本题与网线主管一样，本质上都是一个求边界最值的计算，可以直接套用我们边界二分法的模板。

在编写代码前需要搞清楚以下几个问题：

1、两个边界值分别是什么？
2、搜索条件是什么？

我们现在开始一一分析。

首先，最小值应该是全部花销中最大那个月的数值，你无论如何组合你的花销，都不可能小于这个值。

其次，最大值则是全部开销的总和，即当目标的拆分月份只有一个月的时候，所有的开销统归于这一个月。

最后，搜索条件就应该是按照每次尝试的预算额进行月份的分组，当月份的分组数大于目标分组时，这说明预算太小了，需要增加预算；当月份的分组数小于目标分组时，则说明预算额大了，需要降低。

特别需要注意的是，即使达到了目标拆分组数，也不要急于返回，别忘了我们是要找尽可能小的边界值，所以，当达到了目标分组数后，我们依然需要右移边界！

编码

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

using namespace std;
int n, m, sum, a[100010];

//按预算划分财务月
int CountMonth(int x) {
    long long sum = 0, cnt = 1;
    for (int i = 1; i <= n; i++) {
        sum += a[i];
        if (sum > x) {
            //分割出一个月
            cnt++;
            sum = a[i];//相加超过最大值时重新计数
        }
    }
    return cnt;
}

//二分搜索
int Search(int left, int right) {
    int mid;
```

```
    while (left <= right) {
        mid = left + (right - left) / 2;
        int countNum = CountMonth(mid);
        //等于目标月份，尽可能的缩减预算
       if (countNum == m) {
            right = mid - 1;
        }
            //预算太大，缩减
       else if (countNum < m) {
            right = mid - 1;
        }
            //预算太小，增加
       else if (countNum > m) {
            left = mid + 1;
        }
    }

    //尽可能小的数据
    return left;
}

int main() {
    cin >> n >> m;
    int l = 0, r = 0;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        sum += a[i];  //寻找最大的右边界，预算再大也不能大于该值
       l = max(l, a[i]);  //寻找最大的左边界，预算再小也不可能小于该值
   }
    r = sum;
    cout << Search(l, r);
    return 0;
}
```