HZOJ-242. 最大平均值总结

> ## 题目描述
>
> 给定一个有 N 个元素的非负序列，求长度大于等于 M 的连续子序列的最大平均值。
>
> ## 输入
>
> 第一行输入两个数 N,M。（$1 \leq N, M \leq 100000$） 接下来 N 行，每行输入个数表示非负序列。
>
> ## 输出
>
> 输出一个整数表示最大平均值乘 1000 的结果。

算法设计

第一步：构建一个二分查找模型 （1111100000）

原问题可以转化为：假定可以构造一个函数，自变量为连续长度大于等于 M 的子序列的平均值 A ，因变量是一个布尔值，该值为 1 ，表示存在一个长度大于 M 的子序列的平均值大于 A ， 该值为 0 ，表示不存在一个长度大于 M 的子序列的平均值大 于 A。又假定最大平均值为 x ，则有自变量 $A \leq x$ 时， 函数值为 1，$A > x$ 时，函数值为 0 ，可见该函数为单调递减 函数。x 值可以用二分查找方式得到。

第二步：处理原数组

给原数组的每一位减去A。因为：
设 $[i, j]$ 的区间和为 sum ， $\frac{sum}{j-i+1} = L$ ， 有 $\frac{sum}{L} \geq A$ 则：

$$\frac{sum}{L} \geq A \; sum \qquad \geq A \times L \; sum - A \times L \geq 0$$

因为 $[i, j]$ 有 L 个数字，所以给原数组的每一位减去A。

第三步：为处理后的数组计算前缀和

为原数组计算前缀和的原因是为了快速计算区间和，假定前缀和数组为 sum， 区间为 $[i, j]$ ，则区间和为 $sum[j] - sum[i-1]$ 。 如果用遍历法计算区间和，时间复杂度为 $O(n)$ ， 如果用前缀和计算区间和，时间复杂度为 $O(1)$ ， 所以，应当使用前缀和计算区间和。

第四步：计算因变量

维护一个值 $minNum$ ，用于计算 数组的 $i-M$ 位前 的最小值，其中 i 表示当前处理的下标。

从 M 一直遍历到 N ，计算 $[minNum, i]$ 的区间和，根据 上面推导的公式可得：如果 $[minNum, i]$ 的区间和大于 0 则 返回 1 ，否则继续遍历。如果遍历结束还没有找到合适的结果， 则返回 0 。

第五步：得出答案

二分查找结束后，返回 $tail$ 。

代码展示

```cpp
#include <bits/stdc++.h>
using namespace std;
vector<double> gData;
vector<double> gSum;
int gN, gM;
bool check (double a) {
    gSum[0] = 0;
    for (int i = 1; i <= gN; i++)
        gSum[i] = gData[i] + gSum[i - 1] - a;
    double minNum = 0;
    for (int i = gM; i <= gN; i++) {
        minNum = min (minNum, gSum[i - gM]);
        if (gSum[i] - minNum > 0)
            return true;
    }
    return false;
}
double solve () {
    double head = 0, tail = gSum[gN], mid;
    while (tail - head > 0.001) {
        mid = (head + tail) / 2;
        if (check (mid))
            head = mid;
        else
            tail = mid;
    }
    return tail;
}
int main () {
    scanf ("%d%d", &gN, &gM);
    gData.resize (gN + 5, 0);
    gSum.resize (gN + 5, 0);
    for (int i = 1; i <= gN; i++) {
        scanf ("%lf", &gData[i]);
        gSum[i] = gData[i] + gSum[i - 1];
    }
    int ans = ( int )(solve () * 1000);
    printf ("%d\n", ans);
```

```
    return 0;
}
```