**题目描述 Description**

有 N 堆纸牌，编号分别为 1，2，…, N。每堆上有若干张，但纸牌总数必为 N 的倍数。可以在任一堆上取若于张纸牌，然后移动。  
　　移牌规则为：在编号为 1 堆上取的纸牌，只能移到编号为 2 的堆上；在编号为 N 的堆上取的纸牌，只能移到编号为 N-1 的堆上；其他堆上取的纸牌，可以移到相邻左边或右边的堆上。  
　　现在要求找出一种移动方法，用最少的移动次数使每堆上纸牌数都一样多。  
  
　　例如 N=4，4 堆纸牌数分别为：  
　　①　9　②　8　③　17　④　6  
　　移动3次可达到目的：  
　　从 ③ 取 4 张牌放到 ④ （9 8 13 10） -> 从 ③ 取 3 张牌放到 ②（9 11 10 10）-> 从 ② 取 1 张牌放到①（10 10 10 10）。

**输入描述 Input Description**

第一行N（N 堆纸牌，1 <= N <= 100）  
第二行A1 A2 … An （N 堆纸牌，每堆纸牌初始数，l<= Ai <=10000）

**输出描述 Output Description**

输出至屏幕。格式为：  
所有堆均达到相等时的最少移动次数。‘

**样例输入 Sample Input**

4  
9 8 17 6

**样例输出 Sample Output**

3

**数据范围及提示 Data Size & Hint**

e

#include<iostream>

#include<cstring>

#include<stdio.h>

using namespace std;

int sum(0);

int n,m;

int f[110][110],a[110][110];

int main()

{

//freopen("input.txt","r",stdin);

int n;

int a[110];

cin>>n;

for(int i=1;i<=n;i++)

{

cin>>a[i];

sum+=a[i];

}

sum/=n;

int ans(0);

for(int i=1;i<=n;i++)

if(a[i]<sum)

{

a[i+1]-=sum-a[i];

a[i]=sum;

ans++;

}

else

if(a[i]>sum)

{

a[i+1]+=a[i]-sum;

a[i]=sum;

ans++;

}

cout<<ans<<"\n";

return 0;

}

算法就是从第一个依次向右扫

这个步骤是合理的。但是看不出来是最优的。可见，贪心法确实是比较容易实现，因为比较符合人类直觉，但是不好证明。

这对于每一步来说是局部最优

但是不是全局最优（也就是说，贪心就是要满足局部最优即是全局最优，而动态规划却不一定

）

贪心算法(又称贪婪算法)是指,在对问题求解时,总是做出在当前看来是最好的选择。也就是说,不从整体最优上加以考虑,他所做出的仅是在某种意义上的局部最优解。

0）假设纸牌数量可以是负数

1）对于最左边的纸牌，为了使它的纸牌数达到平均，只要还没有达到平均无论其余子情况如何移动，一定有一步是把自己多余的纸牌移动到右边，或者是从右边移动进来自己差了多少张纸牌

2）第一堆牌只有和右边进行交互是合法的，步骤1是必须的

3）处理好第一堆后，其余操作一定不涉及第一堆，否则答案更劣（经过前一堆是没有意义的）

4）无视第一堆，于是现在又是情况1了（子结构）

就从前往后扫，等到了最后一堆他肯定是已经是平均值了的，你就不需要去单独处理第一个和最后一个