1 zz 和肉

(meat.cpp)

zz 对肉上瘾！zz 的朋友 kk 想让她开心 n 天。为了在第 i 天快乐，她需要吃整整 a 斤的肉。

住宅区有一家大商店，kk 想从那里给她买肉。在第 i 天，他们以每公斤 p_i 元的价格出售肉类。kk 知道所有的数字 $a_1,...,a_n$ 和 $p_1,...,p_n$。每天，他可以买任意数量的肉，也可以保留一些肉以备将来食用。

kk 做饭有点累，所以他请求你的帮助。帮助他尽量减少花在 zz 上的钱，让他快乐 n 天。

输入

第一行输入包含整数 n（$1 \le n \le 10^5$）、天数。

在接下来的 n 行中，第 i 行包含两个整数 ai 和 pi（1≤ai、pi≤100）、zz 需要的肉量和当天的肉成本。

输出

在一行中打印出让达夫开心 n 天所需的最低金额。

paper.in	paper.out
3 1 3 2 2 3 1	10
3 1 3 2 1 3 2	8

说明：

在第一个样本案例中：最佳的方法是第一天购买 1 公斤，第二天购买 2 公斤，第三天购买 3 公斤。

在第二个样本案例中：最佳的方法是第一天买 1 公斤，第二天买 5 公斤（第二天和第三天需要的肉）。

思路：显然买后面的肉时，如果前面某天的肉更便宜，那么直接在那天提前买，因此遍历时，只需记录一个到当天为止肉的最低价格，之后的价格如果高于这个最低价格，就按最低价格买，如果低于，那就更新最低价格。

参考代码：

```
#include<iostream>
#include<cstring>
using namespace std;
const int N=1e5+5;
int a[N],b[N];
int main()
{
    int n,mi=110,xb;
    cin>>n;
    for(int i=1;i<=n;i++)cin>>a[i]>>b[i];
    int res=0;
    for(int i=1;i<=n;i++){
        if(mi>b[i]){
            mi=b[i];
        }
        res+=a[i]*mi;
    }
    cout<<res;
}
```

2 家庭作业

(homework.cpp)

ZZ 今天第一次去学校，由于老师给他的作业太难所有他需要你的帮助来完成：

老师给 Filya 一个仅包含正整数的数组 $a_1,a_2,...,a_n$。首先，他任意选择一个整数 x，然后他将 x 加到数组的某些元素上（仅有一次），再从其他一些元素中减去 x，（每个元素仅减一次）并留一些元素不进行操作。经过这些操作以后，他希望数组内所有的元素均相等。现在，他想知道他可不可以找到这样一个 x 并进行上述操作，使数组内的每一个元素相等。

输入格式

第一行包含一个整数 *n(1<=n<=100000)* 表示数组内的元素个数。

第二行包含 n 个整数 $a_1,a_2,...,a_n$ $(0\le a_i\le 10^9)$ 即为数组内的元素。

输出格式

输出仅为一行如果数组内的元素在经过题目内所述操作后不能全部相等，则输出 NO ； 否则，输出 YES 。

homework.in	homework.out
5 1 3 3 2 1	YES
5 1 2 3 4 5	NO

样例解释

在第一个样例中，应该选择第一个元素作为 x，添加到第一个和第五个，然后从第二个和第三个减去。

思路：首先可以将所有元素排序，然后重复的元素删去，若留下至少四个不同的数，可以发现无论如何操作都无法将全部的数都变相同，若有三个不同的数，肯定是最小的和最大的变到中间的，而中间的数不变，需满足 a[2]-a[1]==a[3]-a[2];若有两个或者一个相同的数，一定能满足所有的数都变相同，分类讨论即可。

参考代码：

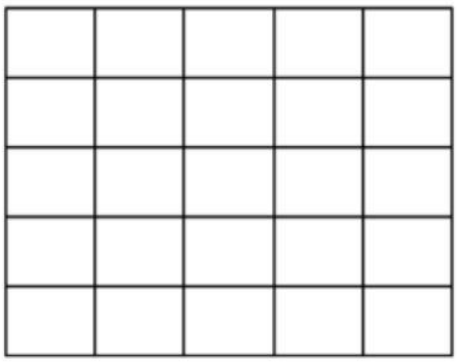
```
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=1e5+5;
int a[N],b[N],cnt;
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)cin>>a[i];
    sort(a+1,a+1+n);
    a[0]=-1;
    for(int i=1;i<=n;i++){
        if(a[i]!=a[i-1])b[++cnt]=a[i];
    }
    if(cnt>3)puts("NO");
```

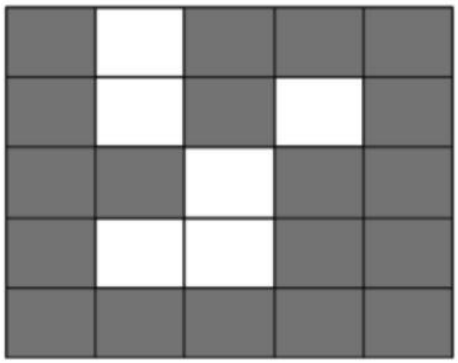
```
    else if(cnt==3){
        if(b[3]-b[2]==b[2]-b[1])puts("YES");
        else puts("NO");
    }
    else puts("YES");
}
```

3 方格稿纸

(paper.cpp)

小猪在小学中认识了很多的字，终于会写一点作文了。某天小猪买了一张方格稿纸来写作文，n 行 m 列。形状如下所示：





某天小猪的邻居小小猪来小猪家玩，用黑墨水笔把小猪新买的方格稿纸涂黑了很多格子。每个格子不是完全黑色就是完全白色。如上图所示。

小猪不能责怪小小猪。作文写不成了，他觉得很无聊，就开始数里面有多少魔幻方阵。

如果稿纸中一个 k×k 的正方形区域满足以下两个条件，那么它就是魔幻方阵：

1．黑白格子的数量差不能超过 1；

2．k 不能小于 2。

现在请你帮小猪求出他被染色的稿纸里面有多少个魔幻方阵。

输入格式

第一行有二个正整数 n 和 m（互相之间以一个空格分隔），表示稿纸共有 n 行 m 列。

接下来 n 行，每行有 m 个 0 或 1 的整数（互相之间以一个空格分隔），代表每个格子的颜色。如果这个数是 1 则为黑色，是 0 则为白色。

输出格式

仅有一行，该行只有一个整数，表示稿纸中魔幻方阵的个数。

paper.in	paper.out
5 5 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1	9

说明/提示

50%的数据，1≤n≤10，1≤m≤10；
75%的数据，1≤n≤180，1≤m≤180；
100%的数据，1≤n≤300，1≤m≤300。

思路：

直接枚举每个正方形，判断是否符合题目要求，使用二维前缀和即可。

参考代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[305][305];
int main(){
    int n,m,x;
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>x;
            a[i][j]=a[i-1][j]+a[i][j-1]-a[i-1][j-1]+x;
        }
    }
    int num,s=0;
    for(int k=2;k<=n&&k<=m;k++){
        for(int i=k;i<=n;i++){
            for(int j=k;j<=m;j++){
                num=a[i][j]-a[i-k][j]-a[i][j-k]+a[i-k][j-k];
                if(abs(k*k-2*num)<=1){
                    s++;
                }
            }
        }
    }
```

```
    cout<<s;
}
```

4 [5503]乔治买手机 dp

题目描述：

新的 华为 mate50 最近发布了， 乔治非常想买它。不幸的是， 他没有足够的钱，所以乔 治打算做一名程序员。现在他在工作中面临以下问题。

给定 n 个整数 p1、p2、… 、pn 的序列。您要选择 k 组整数，每组数据是一段连续的序 列：[l1，r1]，[l2，r2]，…，[lk，rk] (1≤l_1≤r_1<l_2≤r_2<…<l_k≤r_k≤n;r_i-l_i+1=m)，使得和的值 $\sum_{i=1}^{k}\sum_{j=l_i}^{r_i} p_j$ 是最大可能的。帮助乔治完成任务。

输入

第一行包含三个整数 n、m 和 k(1≤ (m×k) ≤n≤10000). 第二行包含 n 个整数 p1、p2、…、 pn (0≤p_i≤10^9)。

输出

单行打印整数- 总和的最大可能值。

输入样例 1：

5 2 1

1 2 3 4 5

输出样例 1：

9

输入样例 2：

7 1 3

2 10 7 18 5 33 0

输出样例 2：

61

题目类型：

dp

分析：

题目大意： 从 n 个数中选出 k 段长度为 m 的连续数组，使得这 k 段加和最大。所选取的每段之间可以是连续的，也可以不是连续的，但不能有重叠。我们使 dp[i][j]代表着 从

前 i 个数字中选取 j 段的最大值。对于 dp[i][j]来说，它可以继承前 i-1 个数选取 j 段的最大 值，也可以是前 i-m （因为不重叠）个数选取 j-1 段之后的最大值，再加上这段得到的值。
状态转移方程： dp[i][j]=max(dp[i-1][j],dp[i-m][j-1]+cnt);//cnt 为从 i-m 到 i 的数字和。
参考代码：

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

long long a[100005],dp[5005][5005];

int main(){
    int n,m,k;
    scanf("%d%d%d",&n,&m,&k);
    for(int i=1;i<=n;i++){
        scanf("%lld",&a[i]);
        a[i]+=a[i-1];////前缀和方便求和
    }
    long long cnt;
    for(int i=m;i<=n;i++){
        cnt=a[i]-a[i-m];
        for(int j=1;j<=k;j++){
            dp[i][j]=max(dp[i-1][j],dp[i-m][j-1]+cnt);
        }
    }
    printf("%lld\n",dp[n][k]);
    return 0;
}
```

5 [8249] Alchemy 炼金术

题目描述：
总是热衷于培养新的爱好的奶牛 Bessie 正在学习如何转化金属。 对于 1≤i≤N≤100 ，她有 ai 单位的金属 i。
此外，她知道 K 个配方，她可以融合若干种金属各一单位，制造一单位编号大 于所有被融合金属的金属。
另外保证，对于每种金属，Bessie 最多知道一种制造该金属的配方。 计算经过一系列转化后，Bessie 可能拥有的金属 N 的最大单位数。 输入格式
输入的第一行包含 N。

第二行包含 N 个整数 aiai。
第三行包含 K。
以下 K 行每行包含两个整数 L 和 M ，随后是 M 个整数。后 M 个整数表示配方 中用于制造一单位金属 L 所需要被融合的金属。输入保证 L 大于这 M 个数。 输出格式
输出在应用一系列零次或多次转化后，Bessie 可能拥有的金属 N 的最大单位数。
数据范围
$1 \le N \le 100$,
$0 \le ai \le 104$,
$1 \le K < N$,
$2 \le L \le N$,
$1 \le M < L$。
输入样例：

5

2 0 0 1 0

3

5 2 3 4

2 1 1

3 1 2

输出样例：

1

样例解释
在这个例子中，以下是一种最优的转化方式：
将一单位金属 1 转化为金属 2。
将一单位金属 2 转化为金属 3。
将一单位金属 3 和金属 4 转化为金属 5。
现在 Bessie 还有一单位金属 1 和一单位金属 5。
她无法再制造更多的金属 5。

分析：

先把原来就有的 n 号金属拿去，然后考虑合成。遍历制作 n 号金属的合成表，能做几个就做几个。如果哪个金属没有了，再看它的合成表，可以做的话就做一 个（注意：只做一个！），能再做一个 n 号就再做一个。如果还能再做一个，就 继续做，以此类推。

那该如何判断能不能做呢？这就要用到深搜的思想了。如果 n 号金属缺材料， 那么就看缺的材料的合成表。如果还缺，就继续看；如果不缺，就做一个。如果 缺的材料无法制作，那就做不下去了。这时就可以输出答案了。

```cpp
#include <iostream>
using namespace std;
int a[110];
int b[110][110];//b[i][j]，合成金属 i  需要金属 j

bool dfs(int k){//dfs 代表我的 k 元素能不能被制造
    if(a[k]>0)
        return true;
    if(b[k][0]==0)//k 元素用完了，且无法被制造
        return false;
    for(int i=b[k][0];i>0;i--){
        if(!dfs(b[k][i]))//b[x][i]元素 无法被制造了，说明 k 也无法被制造
            return false;
    }
    for(int i=1;i<=b[k][0];i++){
        a[b[k][i]]--;
    }
    a[k]++;//合成了一个金属 k
    return true;
}
int main() {
    int n,k,t;
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    cin>>k;
    for(int i=1;i<=k;i++){
        cin>>t;//
        cin>>b[t][0];//合成 t 需要 b[t][0]种金属
        for (int j=1;j<=b[t][0];j++)
            cin>>b[t][j];
    }


    int ans=a[n];
    a[n]=0;
    while(dfs(n)){
        ans++;
        a[n]=0;
    }
    cout<<ans;
    return 0;
}
```