# 第七次上机题解

**Max.D.**

**前言**：这次因为种种原因，大家受到了3小时18题的摧残。如果打击到了大家的自信，那助教们必须抱歉了。不过还是希望大家能扎实提高自己的编程水平，不因为挫折影响自己的情绪。

下面进入正题。

## A 众数 简化版

题目很直白，让我们计算一组数中的众数。做法有很多，列举一下比较有价值的。

### 方法一

先将所有数字排序，在排序之后，所有的相同数字自然会挨在一起，最后用循环扫一次统计，可得答案。

```
#include<stdio.h>
#include<stdlib.h>
#define N 1005
#define max(a,b) (a>b?a:b)

int i,n,a[N];

int compare(const void* a, const void* b)
{
    return *(int*)a>*(int*)b;
}

int main()
{
    scanf("%d", &n);
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    qsort(a,n,sizeof(int),compare);
    int num=1,ans=0;
    for(i=1;i<n;i++)
    {
        if(a[i]==a[i-1])
            num++;
        else
            num=1;
        ans=max(ans,num);
    }
    printf("%d",ans);
    return 0;
}
```

这里先用qsort排序，效率O(nlogn)（因为数据范围不大，也可以用冒泡排序，选择排序等），num代表当前数字已经出现的次数，在循环过程中不断被更新，并用num更新ans。则最大值ans就是答案。

下面是另一种我很喜欢的风格，增加一个j指针，从起点扫到第一个不一样的位置，j-i就是a[i]出现的次数

```
#include<stdio.h>
#include<stdlib.h>
#define N 1005
#define max(a,b) (a>b?a:b)

int i,n,a[N],ans;

int compare(const void* a, const void* b)
{
    return *(int*)a>*(int*)b;
}

int main()
{
    scanf("%d", &n);
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    qsort(a,n,sizeof(int),compare);
    for(int i=0,j;i<n;i=j)
    {
        j=i;
        while(j<n&&a[j]==a[i])
            j++;
        ans=max(ans,j-i);
    }
    printf("%d",ans);
    return 0;
}
```

## 方法二

由于数据范围中要求a[i]绝对值不超过1000，那么，我们可以使用“桶”的方法，把数值当做数组下标，统计下标的出现次数。需要注意的是，由于数值有负数，我们可以加上一个一千多的偏移量，或者采用指针实现负数下标来进行处理。效率 O（m），m表示a[i]范围。

```
#include<stdio.h>
#define max(a,b) (a>b?a:b)

int b[2010],*h=b+1005,n,a,ans;

int main()
{
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d",&a);
        h[a]++;
    }
    for(int i=-1000;i<=1000;i++)
        ans=max(ans,h[i]);
    printf("%d",ans);
    return 0;
}
```

## 方法三

由于数据范围不大，可以直接枚举每一个出现的数，然后再在原数组里搜索和它相等的数的个数而不必经过排序。这样的效率是O(n^2)，和在冒泡排序前提下的方法一相同。

另外也不必要一定是出现的数字，直接-1000~1000枚举也行。

# B 好题

## 方法一

我们知道，实际上需要解决的是三只队伍的三次排序问题，只是排序中比较大小需要考虑字典序罢了。

```c
#include <stdio.h>
#include <string.h>

char name[4][20];
int p[4];
int q[4];

int o[4];

int main()
{
    int i,j;
    int t;
    for (i=1;i<=3;i++)
        scanf("%s%d%d",name[i],p+i,q+i);

    for(i=1;i<=3;i++)
        o[i]=i;

    for (i=1;i<3;i++)
        for (j=i+1;j<=3;j++)
            if (p[ o[i] ]<p[ o[j] ] || ( p[ o[i] ] == p[ o[j] ]  && strcmp(name[o[i]],
name[o[j]])>0 ) )
            {
                t = o[i];
                o[i] = o[j];
                o[j] = t;
            }

    for (i=1;i<=3;i++)
        if (p[ o[i] ] == p[o[1]]) printf("%s ",name[o[i]]);
    printf("\n");

    for (i=1;i<3;i++)
        for (j=i+1;j<=3;j++)
            if (q[ o[i] ]<q[ o[j] ] || ( q[ o[i] ] == q[ o[j] ]  && strcmp(name[o[i]],
name[o[j]])>0 ) )
            {
                t = o[i];
                o[i] = o[j];
                o[j] = t;
            }

    for (i=1;i<=3;i++)
        if (q[ o[i] ] == q[o[1]]) printf("%s ",name[o[i]]);
    printf("\n");

    for (i=1;i<3;i++)
        for (j=i+1;j<=3;j++)
            if (p[ o[i] ] + q[ o[i] ] <p[ o[j] ] + q[ o[j] ] || ( p[ o[i] ] + q[o[i]] == p[ o[j]
]  + q[ o[j] ] && strcmp(name[o[i]], name[o[j]])>0 ) )
            {
                t = o[i];
                o[i] = o[j];
                o[j] = t;
            }

    for (i=1;i<=3;i++)
        if (p[ o[i] ] + q[ o[i] ] == p[o[1]] + q[o[1]] ) printf("%s ",name[o[i]]);
    printf("\n");
```

```
    return 0;
}
```

上面采用的是选择排序，需要注意的，一是排序条件 p[ o[i] ]<p[ o[j] ] || ( p[ o[i] ] == p[ o[j] ] && strcmp(name[o[i]], name[o[j]])>0 表示的是先考虑数值，数值相同的情况下，考虑字典序，这样相同数值情况下，字典序就能从小到大了；二是这里排序的不是队伍的信息本身，而是队伍的“编号”，这样的话，我们不必对队伍本身进行交换，这是一个很棒棒的技巧。

## 方法二

题意中我们需要对三个队伍进行挑选，求最大值以及对应的队伍很方便，但其中头疼的是字典序问题。经过考虑之后，我们发现，先把这三只队伍按名字的字典序排好，再求解问题，就可以直接顺序输出答案了。

虽然只有三个队伍，为了方便，还是需要排序。这次同样我们只排序编号，更为方便

```
#include<stdio.h>
#include<string.h>
#define max(a,b) (a>b?a:b)

char nam[3][25];
int p[3],q[3],o[3]={0,1,2},f=0,tmp;

int main()
{
    f=-1E9;
    for(int i=0;i<3;i++)
        scanf("%s%d%d",nam[i],&p[i],&q[i]);
    for(int t=0;t<2;t++)
        for(int i=0;i<2-t;i++)
            if(strcmp(nam[o[i]],nam[o[i+1]])>0)
                tmp=o[i],o[i]=o[i+1],o[i+1]=tmp;
    for(int i=0;i<3;i++)
        f=max(f,p[i]);
    for(int i=0;i<3;i++)
        if(p[o[i]]==f)
            printf("%s ",nam[o[i]]);
    puts("");
    f=-1E9;
    for(int i=0;i<3;i++)
        f=max(f,q[i]);
    for(int i=0;i<3;i++)
        if(q[o[i]]==f)
            printf("%s ",nam[o[i]]);
    puts("");
    f=-1E9;
    for(int i=0;i<3;i++)
        f=max(f,p[i]+q[i]);
    for(int i=0;i<3;i++)
        if(p[o[i]]+q[o[i]]==f)
            printf("%s ",nam[o[i]]);
    return 0;
}
```

最后补一句，strcmp函数在不同系统中设置不同，正数代表字典序的大于，而不是1，所以strcmp(s1,s2)==1的写法是不正确的，而是strcmp(s1,s2)>0。（我被这个坑了QwQ）

# C Ausar的字符串替换

## 方法一

题目理解很简单，就是字符串替换，需要注意是从左到右逐一寻找替换，不考虑替换之后对原有匹配情况的改变。　另外，我们没必要把答案整进一个数组，毕竟，你也不知道替换之后字符串长度（可能不止1000了）

```
#include <stdio.h>
#include <string.h>

char A[1005],B[1005],C[1005];

int main()
{
    scanf("%s%s%s",A,B,C);
    int Blen=strlen(B);
    char *p=A,*x;
    while(*p)
    {
        x=strstr(p,B);
        if(x==NULL)
        {
            printf("%s",p);
            break;
        }
        while(p<x)
        {
            putchar(*p);
            p++;
        }
        printf("%s",C);
        p+=Blen;
    }
    return 0;
}
```

用p来不断跟进。如果strstr函数找到子串B，就输出C，p增进Blen。这种做法很有技巧，利用了数组和指针之间的关系，需要大家细细体会。

## 方法二

如果你不是很喜欢指针与string.h里的库函数，写起来也不麻烦。

```c
#include<stdio.h>
#include<string.h>

char s[1005],a[1005],b[1005];
int n,x,y,ma,mb;

int main()
{
    scanf("%s%s%s",s,a,b);
    n=strlen(s);
    ma=strlen(a);
    mb=strlen(b);
    for(int i=0;i<n;i++)
    {
        int f=ma+i<=n;
        for(int j=0;j<ma&&f;j++)
            if(a[j]!=s[i+j])
                f=0;
        if(f)
        {
            printf("%s",b);
            i+=ma-1;
        }
        else
            putchar(s[i]);
    }
    return 0;
}
```

f是一个标记，代表着是否从i开始存在和B相同的一个匹配。如果匹配的话，输出C字符串，否则输出本字符。

# D Ausar的子串翻转

熟练掌握循环的话这题当然不难，翻转我们可以枚举左半边，和右半边逐一交换,或者可以倒序放置入一个新数组，再放回来。直接贴上代码。

```
#include <stdio.h>

char ch[1010],tmp;

int main()
{
    fgets(ch,1005,stdin);
    int n,l,r;
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d%d",&l,&r);
        while(l<r)
        {
            tmp=ch[l],ch[l]=ch[r],ch[r]=tmp;
            l++,r--;
        }
    }
    printf("%s",ch);
    return 0;
}
```

# E X进制A+B

做法很简单，先计算a+b，然后再用X进制格式输出。一个小小的坑点是0+0，很多同学没有输出。

```
#include<stdio.h>

int a, b, x, cnt;
char t[20] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9','A', 'B', 'C', 'D', 'E', 'F' };
char ans[50];

int main()
{
    while (scanf("%d %d %d", &a, &b, &x) == 3)
    {
        a += b;
        cnt = 0;
        while(1)
        {
            ans[cnt++] = t[a%x];
            if (a/x==0) break;
            a /= x;
        }
        while(cnt--)
            printf("%c", ans[cnt]);
        printf("\n");
    }
    return 0;
}
```

再来一个递归风格的~

```
#include<stdio.h>

int a,b,x;

void print(int a)
{
    if(a==0)
        return ;
    print(a/x);
    putchar(a%x<10?a%x+'0':a%x-10+'A');
}

int main()
{
    while(scanf("%d%d%d",&a,&b,&x)==3)
    {
        a+=b;
        if(a>0)
            print(a);
        else
            putchar('0');
        puts("");
    }
    return 0;
}
```

# F 多简单

这题主要目的其实就是让大家体验体验指针...确实挺简单的。照搬题目的步骤做，然后下面四个式子自己推一下就行了。

```
#include <stdio.h>

void fun(int *a,int *b)
{
    *a = *a + *b;
    *b = *a - *b;
}

int a,b,c,d;

int main()
{
    scanf("%d%d%d%d",&a,&b,&c,&d);
    fun(&a,&b);
    fun(&c,&d);
    fun(&b,&c);
    printf("%d %d %d %d\n",a,b,c,d);
    printf("na=a+b\n");
    printf("nb=a+c+d\n");
    printf("nc=a\n");
    printf("nd=c\n");
    return 0;
}
```

# G 鞍点

一道考察大家循环控制的经典题。方法很简单，先找每行最大值，再判断它是不是这个列的最小值，如果是的话就加入答案数组中。需要注意的是这里最大值最小值都是严格的。

```
#include<stdio.h>
#define max(a,b) (a>b?a:b)

int a[2005][2005],n,m,ax[2005],ay[2005],an;

int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            scanf("%d",&a[i][j]);
    for(int i=1;i<=n;i++)
    {
        int mv=-1E9,cnt=0,p;
        for(int j=1;j<=m;j++) //求最大值
            mv=max(mv,a[i][j]);
        for(int j=1;j<=m;j++)
            if(mv==a[i][j])
            {
                cnt++;
                p=j;
            }
        if(cnt==1) //判断最大值唯一性
        {
            int f=1;
            for(int j=1;j<=n&&f;j++)
                if(j!=i&&a[j][p]<=a[i][p])
                    f=0;
            if(f)
                ax[an]=i,ay[an]=p,an++; //记录鞍点
        }
    }
    printf("%d\n",an);
    for(int i=0;i<an;i++)
        printf("%d %d %d\n",ax[i]-1,ay[i]-1,a[ax[i]][ay[i]]);
    return 0;
}
```

# H login的淑芬半期复习

似乎大家都不喜欢按常理出牌，用二分法或者直接循环的人不少QwQ。

很好的期中复习题，假如不熟悉牛顿法，请翻淑芬书，或者查看链接。牛顿法和泰勒公式有密切联系。

至于不幸运选择极值点,或者极值点附近的点导致迭代速度慢的话，我们可以用随机的方式重新挑选。

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

const double eps=1e-6;

int chd(double w)
{
    if(w>eps)
        return 1;
    else if(w<-eps)
        return -1;
    else
        return 0;
}

double n,m,A;

double f(double x)  //f(x)
{
    return pow(x,n)/m+cos(x)-A;
}

double fp(double x) //f'(x)
{
    return n*pow(x,n-1)/m-sin(x);
}

int main()
{
    scanf("%lf%lf%lf",&n,&m,&A);
    double x=0;

    while(chd(fp(x))==0)    //确保不是极值点
        x=((double)rand()/(double)RAND_MAX-0.5)*6; //随机
    while(chd(f(x))!=0)  //寻找答案
        x=x-f(x)/fp(x);
    printf("%.8lf",x);
    return 0;
}
```

PS:其实大部分时候，直接随机一个点，不用检查，基本上可以过本题的~。

# I HugeGun学姐的梦魇1

组合数我们可以通过二维数组进行一个递推，这个大家必须学会。另外，可以知道，整除等价于取模为0，因此，我们预处理出组合数数组之后，把所有值为0的地方置1，而其余情况置0。再对这个新数组求一个二维前缀和（sum[i][j]代表该处左上方阵元素之和）。这个前缀和的值就是我们想要的答案了。每次询问都可以直接回答。

二维前缀和的求法(大家可以细心体会一下为什么成立）：

$$sum[i][j] = sum[i][j-1] + sum[i-1][j] - sum[i-1][j-1] + num[i][j]$$

```
#include<stdio.h>

int a[2005][2005],sum[2005][2005],mo,t,n,m;

int main()
{
    scanf("%d%d",&t,&mo);
    a[0][0]=1;
    for(int i=1;i<=2000;i++)
    {
        a[i][0]=1;
        for(int j=1;j<=i;j++)
            a[i][j]=(a[i-1][j]+a[i-1][j-1])%mo;
    }
    for(int i=0;i<=2000;i++)
        for(int j=0;j<=2000;j++)
            sum[i][j]=(i?sum[i-1][j]:0)+(j?sum[i][j-1]:0)-(i*j?sum[i-1][j-1]:0)+(j<=i&&a[i]
[j]==0);
    while(t--)
    {
        scanf("%d%d",&n,&m);
        printf("%d\n",sum[n][m]);
    }
    return 0;
}
```

**小结**：前九题中，真正的难题只有I，而B，H，G属于较难，需要花点心思，其余五题大家都应该掌握。大家写循环的程序，熟练度还有待增加。

# JKLMNOPQR By悠唯

# J HugeGun学姐的梦魇1(129/208)

## 解题思路

题意很简单，没什么好说的，直接按照题目所说的模拟即可，没有卡时间，没有卡空间，可以说看懂题就能AC。这题通过人数较少主要原因可能是题目编号偏后以及系列命名混淆视听，所以建议同学们仔细读题之后再确定做题顺序，至少保证比赛结束前每个题目都看过。

## 代码

```
//Author: 彭毛小民
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int n,st,T;
int total;
int hole[101010];

int main(){
    int i;
    memset(hole,0,sizeof hole);
    scanf("%d%d%d",&n,&st,&T);
    hole[st]=1;
    for (i=2; i<=T ;i++)
    {
        st = (st+i-1)%n +1;
        hole[st]=1;
    }
    total=0;
    for (i=1;i<=n;i++) total += hole[i] == 0;
    if (total)
    {
        printf("%d\n",total);
        for (i=1;i<=n;i++)
            if (hole[i]==0) printf("%d ",i);
    }
    else
        printf("have a pokemon");
    return 0;
}
```

---

# K HugeGun学姐的夢魘1(9/36)

## 解题思路

本题属于数学推论+高精度乘法。实际上高精度数乘低精度数写起来不算复杂，所以本题基本是一个推出结论就可以AC的题。TLE的同学基本上是没想出正确结论的，WA拿了20分的同学多半是没想到要写高精度乘法，事实上只需要考虑极限情况$a^{50}$是肯定会超过long long范围的。

下面给出一个简单的推导思考过程：

**考虑第n个数a[n]，如果选了a[n]，则问题转化成了对于序列a[1]~a[n-1]一共n-1个数计算权值，最后乘上a[n]的值。如果不选a[n]，则问题转化成了剩下的n-1个数计算权值，然后乘上1。**这两部分的结果加起来就行了。写成递归式的话就像这样，记S(x)为前x个数计算出的权值和：

$$S(x)=\begin{cases}1 & (x==0\text{，题意中全不选结果为}1)\\ S(x-1)*a[x]+S(x-1) & (x>0)\end{cases}$$

很容易看出a[n]对答案的贡献是a[n]+1。那么最后的答案其实就是

$$Ans=\prod_{i=1}^{n}(a_i+1)$$

写一个高精度计算Ans即可。

## 代码

```
//Author: 翁翾辰
#include<stdio.h>

int main()
{
    long long a[50],n,i,j,s[100]={1},jw;
    scanf("%lld",&n);
    for(i=0;i<n;i++)
    {
        scanf("%lld",&a[i]);
        jw=0;
        for(j=0;j<100;j++)
        {
            s[j]*=(a[i]+1);
```

```
            s[j]+=jw;
            jw=s[j]/10;
            s[j]%=10;
        }
    }
    for(i=99;i>=0;i--)
    {
        if(s[i]!=0) break;
    }
    for(;i>=0;i--) printf("%lld",s[i]);
    return 0;
}
```

---

# L HugeGun学姐的梦魇2(3/5)

## 解题思路

较难数学逻辑题，但是其实仔细读题理顺逻辑的话不难想到。
这里有一个性质，那就是按顺序从两端往中间选点一定是最优的。即选择的点一定是这种情况：（用1表示选择这个点，0表示不选）
1111...111000000000000...0000000001111...111
我们可以用数学归纳法简单证明一下，首先两个点肯定是选择端点。然后新加入一个点，如果某个点不在最左边一团或者最右边一团里，即如果出现了1111...111000000000001000...0000000001111...111这样的情况，我们可以考虑如下步骤：

如果这个点左边和右边的“1”点数量相同，那么选择这个点左边的(或者右边的)点不会改变距离和的结果。那么我们可以把它移到左边或者右边一团里。
如果这个点左边的“1”点数量较多，那么把这个点往右边移动，距离和一定会增加，因为对于所有左边的点来说，距离增加了num(left)*dis，右边减少了num(right)*dis，而num(left)>
如果右边的“1”比左边多，则应该往左边移动。

总之每加入两个点，这两个点一定会被放在新的最左边和最右边。那么代码也就很容易就写出来了。

**我的描述可能比较绕，同学们自己画图多试几个点应该就能明白了。**

## 代码

```
//Author: 刘登元
//这段代码看起来就清爽--By 悠唯
#include<stdio.h>
#define inf 2005
#define ll long long
ll a[inf];
int main(){
    int t;scanf("%d",&t);
    while(t--){
        int n;scanf("%d",&n);
        for (int i=2;i<=n;i++){
            scanf("%lld",&a[i]);
            a[i]+=a[i-1];
        }
        printf("0");
        for (int i=2;i<=n;i++){
            long long ans=0;
            for (int l=1,r=n,k=i-1;l<=i/2;l++,r--,k-=2)
                ans+=(a[r]-a[l])*k;
            printf(" %lld",ans);
        }putchar('\n');
    }return 0;
}
```

---

# M 悠唯的阅读理解题(382/508)

## 解题思路

这才是正常的通过率嘛，纯模拟水题，即使放到M题也掩盖不了签到题的光芒（被打死）。按照题意循环就行了。有一些同学估计是上网或者别的地方找的幻方构造的程序，和样例都不一样就直接交，所以加粗了一句按照题目要求构造幻方。
没什么好说的，注意数组不要越界就行了，然后右下角是特殊情况需要特判。没了。

## 代码

```
//Author: 曾宥崴
#include <stdio.h>
#include <string.h>

int a[1005][1005];

int main()
{
    int n;
    scanf("%d",&n);
    int now = 1, x = n, y = n/2+1;
    a[n+1][n+1] = 999999999;
    while (now <= n*n)
    {
        a[x][y] = now++;

        x++; y++;
        if (a[x][y] != 0) x-=2,y--;
        if (x > n) x = 1;
```

```
        if (y > n) y = 1;
    }

    for (x = 1; x <= n; x++)
    {
        for (y = 1; y <= n; y++)
            printf("%d ",a[x][y]);
        printf("\n");
    }
    return 0;
}
```

---

# N Max的排列（羊蹄）(7/17)

## 解题思路

结论题+找规律题。很多同学估计被劝退的原因一个是不会求逆序对，还有可能是觉得太复杂，估计还有的觉得标了羊蹄的肯定是狼题就跑了（呜呜呜）。
实际上按照顺序对排列进行编号的话，逆序对的个数是可以直接计算出来的。因为n个数的全排列有n!个。而n-1个数的全排列是(n-1)!个，这就意味着如果p小于等于(n-1)!，那么第一位数是不会变的，问题转化成了求剩下的n-1个数的逆序对数量。而如果p大于(n-1)!，则根据定义，p/(n-1)!可以表示第1个数往后移动了几位，那么前面的数全都大于它，把p/(n-1)!加入答案以后，继续计算剩下的n-1个数的逆序对数量。

看代码可能比较清楚。

## 代码

```
//Author: 陈铭煊
#include<stdio.h>

int T,n,ans;
long long fac[25],p;

int main()
{
    fac[1]=1;
    for(int i=2;i<=19;i++)
        fac[i]=fac[i-1]*i;
    scanf("%d",&T);
    while(T--)
    {
        ans=0;
        scanf("%d%lld",&n,&p);
        for(int i=n-1;i>0;i--)
            ans+=p/fac[i],p%=fac[i];
        printf("%d\n",ans);
    }
    return 0;
}
```

---

# O 何花花连泰勒展开都不会就跑来出上机题真是太烂了靠（简单版）(14/23)

## 解题思路

数学计算其实并不难（毕竟泰勒展开你们比我懂多了）。虽然题面比较复杂，不过仔细看完应该不是问题，可能是放到后面的缘故，碰的人较少，但是通过率惊人的有50%以上足以说明问题。

## 代码

```
//Author: 苏星熠
#include<stdio.h>
#include<math.h>
#include<string.h>

int main()
{
    int a,b;
    long m=0,n=0,p;
    while((scanf("%dx%d",&a,&b))==2)
    {
        m+=a*b;
        n+=a;
    }
    p=n-m;
    if(p!=0&&m!=0)
    printf("%ld%+ldx+R(x)",p,m);
    else if(m==0&&p==0)
    printf("R(x)");
    else if(p==0)
    printf("%ldx+R(x)",m);
    else
    printf("%ld+R(x)",p);
}
```

---

# P 摸鱼助教Mogg Ⅲ(3/17)

## 解题思路

没接触过的很难做出来，半个算法题吧，二维前缀和的思想。但是这不是同学们不动这道题的理由，因为事实证明4次方暴力+合理的break可以AC。再不济这题可以捞很多分。

正解：用s[i][j]表示从左上角到[i,j]格的总和，然后对于每个qxq的矩阵可以用很短时间判断出这个矩阵内有没有qxq个1。详情见代码。

## 代码

```
//Author: 陈铭煊
#include<stdio.h>

int n,m,q,cnt,a[2505][2505];

int main()
{
    scanf("%d%d%d",&n,&m,&q);
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
        {
            scanf("%d",&a[i][j]);
            a[i][j]+=a[i-1][j]+a[i][j-1]-a[i-1][j-1];
            //仔细想想，为什么这样可以计算前缀和？ --By 悠唯
            cnt+=(i>=q&&j>=q&&-a[i-q][j]-a[i][j-q]+a[i][j]+a[i-q][j-q]==q*q);
            //我加了个括号便于同学们理解，这里增加的是一个布尔量0或1--By 悠唯
        }
    printf("%d",cnt);
    return 0;
}
```

---

# Q 酸奶击鼓传花(28/87)

## 解题思路

约瑟夫问题，详情查看[链接:https://www.cnblogs.com/cmmdc/p/7216726.html]

不放代码，留个悬念。

# R zt的难题(8/25)

## 解题思路

算法题，防AK用，没有前提知识的同学们可以稍微了解一点。
https://www.cnblogs.com/aabbcc/p/6504605.html