Special-五一赛题解

- `A` 仰望星空
  - 题目分析
  - 参考代码
- `B` 切切切
  - 题目分析
  - 参考代码
- `C` 数0！
  - 题目分析
  - 参考代码
- `D` 七巧板入门
  - 题目分析
  - 参考代码
- `E` 高精度阶乘
  - 题目分析
  - 参考代码
- `F` 排队打饭
  - 题目分析
  - 参考代码
- `G` 作业规划
  - 题目分析
  - 参考代码
- `H` 整数求和
  - 题目分析
  - 参考代码
- `I` 找零
  - 题目分析
  - 参考代码
- `J` cxccxc与Nim游戏
  - 题目分析
  - 参考代码
- `KLM` 位运算加、乘、幂
  - 题目分析
  - 参考代码
- `N` 最大平均字段
  - 题目分析
  - 参考代码
- `O` lxy不疑惑

Special-五一赛题解

A 仰望星空

题目分析

直接输出就行

参考代码

```
#include<stdio.h>
int main()
{
    printf("我仰望星空\n\
```

```
它是那样辽阔而深邃\n\
那无穷的真理\n\
让我苦苦地求索 追随\n\
我仰望星空\n\
它是那样庄严而圣洁\n\
那凛然的正义\n\
让我充满热爱 感到敬畏\n\
我仰望星空\n\
它是那样自由而宁静\n\
那博大的胸怀\n\
让我的心灵栖息 依偎\n\
我仰望星空\n\
它是那样壮丽而光辉\n\
那永恒的炽热\n\
让我心中燃起希望的烈焰\n\
响起春雷\n\
那永恒的炽热\n\
让我心中燃起希望的烈焰\n\
响起春雷");
return 0;
}
```

B 切切切

题目分析

本题是一个简单的数学计算题，可以不需要复杂的算法知识，结合简单的几何计算即可得出结果；

由相似三角形面积与边长的关系可以得到$\frac{x_0+a-c}{a}=\sqrt{\frac{1}{2}}$，解得$c=x_0+a-\sqrt{\frac{1}{2}}\times a$

ps:有的同学用二分去做也是可以的，可以，但没必要。

参考代码

```
#include <stdio.h>
#include <math.h>
int x,y,a,b;
int main(void)
{
    scanf("%d%d%d%d",&x,&y,&a,&b);
    double ans=x+a-sqrt(0.5)*a;
    printf("%.5f\n",ans);

    return 0;
}
```

c 数0!

题目分析

由基本的数学知识可知，x末尾零的个数是分解质因数后$min\{2$的因数，5的因数$\}$，显然因数2的个数少于因数5，因此这道题的本质就是求因数5的个数。在阶乘当中每5个数有一个因数5，每25个数有一个因数25。（以此类推）统计答案即可。

时间复杂度：$NlogN$

参考代码

```
#include<stdio.h>
int main()
{
  long long n,ans=0;
  scanf("%lld",&n);
  while(n!=0){
    ans+=n/5;
    n/=5;
  }
  printf("%lld",ans);
}
```

D 七巧板入门

题目分析

不难分析，最小的两种正方形是由两个直角三角形或者四个直角三角形拼成的。同时，由于这两个正方形的较大者边长是较小者边长的根号2倍，所以这两种正方形不可以互相混搭拼接，只能在他们各自的基础上组成更大的正方形。

那么答案就显而易见了，只有当三角形的个数为 $2 \times x \times x$ 或者 $4 \times x \times x$ (x 为正整数)的情况下，才能拼成三角形，其他时候均不可以。

参考代码

```
#include<stdio.h>
#include<math.h>
const double eps = 1e-8;
int t, n;
int main() {
    scanf("%d", &t);
    while (t--) {
        scanf("%d", &n);
        if (n % 2) puts("NO");
        else {
            int len = (int)(sqrt(n / 2) + eps);
```

```
            int len2 = (int)(sqrt(n / 4) + eps);
            puts((len * len == n / 2 || (n % 4 == 0 && len2 * len2 == n / 4)) ? "YES" : "NO");
        }
    }
}
```

E 高精度阶乘

题目分析

取个对数然后算就行

参考代码

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

int n, ans = 1;

int main()
{
    while (scanf("%d", &n) != EOF)
    {
        ans = 1;
        float sum = 0;
        for (int i = 1; i <= n; i++)
            sum += log(i) / log(10);
        printf("%d ", (int)sum + 1);
        while (n)
        {
            ans *= n;
            while (ans % 10 == 0)
                ans /= 10;
            ans = ans % 1000; //最开始只取末尾非零位开始的1位，结果143时WA，2位时也WA，3位就能通过
```

```
            --n;
        }
        printf("%d\n", ans % 10);
    }
    return 0;
}
```

F 排队打饭

题目分析

主要思路：在输入以后，首先把它们排序，但是需要自己写，因为我用了两个数组

然后用一个变量存放等待时间

注意：第i个人的等待时间为前i-1的人的接水时间总和

参考代码

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
int main(){
    int i,n,a[1005],b[1005],f;//定义数组，一个赋值，一个记录序号
    double s=0;//结果要是浮点数形式
    scanf("%d",&n);//输入
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);//输入
        b[i]=i+1;//如果i为0，要加一
    }
```

```
    int temp;
    for(i=0;i<n;i++){
        for(int j=n-1;j>0;j--){
            if(a[j]<a[j-1]){
                temp=a[j];//冒泡大法！！！
                a[j]=a[j-1];
                a[j-1]=temp;
                temp=b[j];//注意，这里要同时变换
                b[j]=b[j-1];
                b[j-1]=temp;
            }
        }
    }
    for(i=0;i<n;i++){
        printf("%d ",b[i]);//输出序号
    }
    for(i=0;i<n;i++){//这里有个公式，s=a1*n+a2*(n-1)+a3*(n-2)+a4*(n-3)……
一直到 an
        s+=a[i]*(n-i-1);
    }
    printf("\n");
    s/=n;
    printf("%.2lf",s);//保留两位输出
    return 0;
}
```

G 作业规划

题目分析

给出一个比较显然的贪心：在生活当中，大家每一天会选择做哪一项作业？一定是布置了的并且截至时间最早的。根据生活常识这样一定是最优的。所以我们枚举每一天t，在所有作业中找到一个没有做过且$l_i \leq t \leq ri$且r_i最小的，将他完成。统计答案。

此题可用堆优化时间复杂度，由于超纲在此略去。另外还有一些贪心方法，能通过即可。

另外后台看到了dinic，预流推进和匈牙利算法，真是百花齐放。**安利这些同学关注BCPC校赛。**

时间复杂度：N^2

参考代码

```
#include<stdio.h>
struct Work{
  int l,r,k;
};
struct Work a[5005];
int n,ans,m;
int main()
{
  scanf("%d %d",&n,&m);
  a[0].r=1e9;
  for(int i=1;i<=n;i++) scanf("%d%d",&a[i].l,&a[i].r);
  for(int t=1;t<=100000;t++){
    int pos=0;
    for(int j=1;j<=n;j++)
      if(a[j].k==0&&a[j].l<=t&&a[j].r>=t&&a[j].r<=a[pos].r)
        pos=j;
    if(pos!=0){
      a[pos].k=1;
      ans++;
    }
  }
  printf("%d",ans);
}
```

H 整数求和

题目分析

根据hint，对于给定的n，值不相同的 $\lfloor \frac{n}{i} \rfloor$ 不超过 $2\sqrt{n}$ 个。

只要能做到找出 满足$\lfloor \frac{n}{i} \rfloor = k$的$i$的范围即可通过。

由$\lfloor \frac{n}{k} \rfloor = a$以及$\lfloor \frac{n}{a+1} \rfloor \neq k$，可以看出，$a$就是$i$的上限。

时间复杂度：$\sqrt{N}$

参考代码

```
#include<stdio.h>

long long n,nt;
long long ans;

int main()
{
  scanf("%lld",&n);
  for(long long i=1;i<=n;){
    nt=n/(n/i)+1;
    ans+=(n/i)*(nt-i);
    i=nt;
  }
  printf("%lld",ans);
}
```

I 找零

题目分析

这个题目有过很多版本了，在此给出一个最为简单的。

首先为了便于理解，稍微改动一下，把题目中"把区间 $[l,r]$ 上的值更改为1"，变成"把区间 $[l,r]$ 上的值全部加1"，最后还是数0的个数。

构造一个b序列，满足$b_i=\sum_{k=1}^{i}a_i$，a序列和b序列是一一对应的。也就是说最后我们可以通过b序列推出a序列的值。

a序列中"把区间 $[l,r]$ 上的值全部加1"的操作，对应着b序列“$b_l=b_l+1$和$b_{r+1}=b_{r+1}-1$”。

计算完b序列后推出a序列即可。

时间复杂度：N（想把log卡了不过没成功）

参考代码

```
#include <stdio.h>

int n, m, ans;
int a[10000005];

int main()
{
    int x, y;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &x, &y);
        a[x]++;
        a[y + 1]--;
    }
    for (int i = 1; i <= n; i++)
        a[i] += a[i - 1]; // printf("%d ",a[i]);
    for (int i = 1; i <= n; i++)
        if (a[i] == 0)
```

```
            ans++;
    printf("%d", ans);
}
```

J cxccxc与Nim游戏

题目分析

读完题后我们观察到，一个局面对当前操作者来说是必胜的，当且仅当存在一个操作使得下个局面对于对方来说是必败的。而当且仅当所有可能的下一局面对于对方来说都是必胜的，这个局面对当前操作者来说才是必败的。

因此我们可以从最后的状态开始倒推。

这时只剩下一枚硬币，通过简单的推理，我们可以得出

1. 这枚硬币在最左边的方格（即方格1）时，当前操作者直接拿走这枚硬币，获胜。这个局面对当前操作者来说是必胜的
2. 这枚硬币在方格2时，操作者只能将其向左移到方格1，让对方变成情况1的操作者，对方必胜，即当前操作者必败。
3. 这枚硬币在其他位置时，操作者可以将其移动到方格2，造成一个必败局面，从而使自己必胜

这个结论可以由下面的表格概括

x_3	1	2	3	4	5	6	7	...
	W1N	L0SE	W1N	W1N	W1N	W1N	W1N	...

现在我们增加一枚硬币。此时，我们发现只有在左边的硬币位于方格1，且操作者将其拿走，才能转换为我们已知的一枚硬币时的状态。因此我们从此状态出发。

对于左边的硬币位于方格1，右边硬币位于方格2时，操作者可以选择拿走左边的硬币将局势变为只有一枚硬币在方格2的必败局面。因此当前为必胜局面，如下表所示。

x_2, x_3	1	2	3	4	5	6	7	...
1	\	W1N						...

第i行j列表示左边硬币在方格i，右边硬币在方格j

如果右边硬币在方格3，根据规则，当前操作者只能选择拿走左边硬币或左移右边硬币，无论如何都只能导致对方的必胜局面，因此(1,3)的状态为必败。

x_2, x_3	1	2	3	4	5	6	7	...
1	\	W1N	L0SE					...

通过和之前相似的推理，推出剩下的输赢状态:

x_2, x_3	1	2	3	4	5	6	7	...
1	\	W1N	L0SE	W1N	W1N	W1N	W1N	...

如果左边的硬币不在方格1，那么当前操作者只能选择左移左边或右边硬币中的一枚，即下一个局面为表格中向上或向左可以到达的任意一个合法位置，因此，我们可以归纳地推出整张表格，每一格对应一种两枚硬币时的局面状态:

x_2, x_3	1	2	3	4	5	6	7	...
1	\	W1N	L0SE	W1N	W1N	W1N	W1N	...
2	\	\	W1N	L0SE	W1N	W1N	W1N	...
3	\	\	\	W1N	L0SE	W1N	W1N	...
4	\	\	\	\	W1N	L0SE	W1N	...
5	\	\	\	\	\	W1N	L0SE	...
6	\	\	\	\	\	\	W1N	...
7	\	\	\	\	\	\	\	...
...	\	\	\	\	\	\	\	...

我们加入最后一枚硬币。此时的边界情况是$x_1 = 1, x_2 = 2, x_3 = 3$，操作者只能拿走最左侧的硬币，局面转换为上表的(2,3)，为必胜局面，因此$x_1 = 1, x_2 = 2, x_3 = 3$是必败局面。

$x_1=1,x_2,x_3$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**...**
1	\	\	\	\	\	\	\	...
2	\	\	LOSE					...
3	\	\	\					...
4	\	\	\	\				...
5	\	\	\	\	\			...
6	\	\	\	\	\	\		...
7	\	\	\	\	\	\	\	...
...	\	\	\	\	\	\	\	...

而$x_1=1$时的其他局面也同样可以到达其上方、左方的局面状态。除此之外，还可以到达只有两枚硬币的局面状态表格中正对的位置。由此我们推得整个表格：

$x_1=1,x_2,x_3$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**...**
1	\	\	\	\	\	\	\	...
2	\	\	LOSE	W1N	W1N	W1N	W1N	...
3	\	\	\	LOSE	W1N	W1N	W1N	...
4	\	\	\	\	LOSE	W1N	W1N	...
5	\	\	\	\	\	LOSE	W1N	...
6	\	\	\	\	\	\	LOSE	...
7	\	\	\	\	\	\	\	...
...	\	\	\	\	\	\	\	...

同理，$x_1=n$时表格中任意位置对应的局面，其下一个局面可以是向上、向右、向$x_1=1,2,\ldots,n-1$表格的正对位置所对应的局面，分别代表左移三枚硬币中的一枚。继续推导可得：

$x_1=2,x_2,x_3$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**...**
1	\	\	\	\	\	\	\	...
2	\	\	\	\	\	\	\	...
3	\	\	\	W1N	L0SE	W1N	W1N	...
4	\	\	\	\	W1N	L0SE	W1N	...
5	\	\	\	\	\	W1N	L0SE	...
6	\	\	\	\	\	\	W1N	...
7	\	\	\	\	\	\	\	...
...	\	\	\	\	\	\	\	...

$x_1=3,x_2,x_3$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**...**
1	\	\	\	\	\	\	\	...
2	\	\	\	\	\	\	\	...
3	\	\	\	\	\	\	\	...
4	\	\	\	\	W1N	W1N	L0SE	...
5	\	\	\	\	\	W1N	W1N	...
6	\	\	\	\	\	\	W1N	...
7	\	\	\	\	\	\	\	...
...	\	\	\	\	\	\	\	...

.....

通过归纳法不难得出，当$x_1=n$时，只有$x_3-x_2=n$这条线上的局面为必败，其他情况全为必胜局面。

因此我们只需判断$x_1+x_2=x_3$是否成立即可，成立即为必败，否则为必胜。

参考代码

```
#include<stdio.h>
int x1,x2,x3;
int main(){
    scanf("%d%d%d",&x1,&x2,&x3);
    printf("%s",x1+x2==x3?"L0SE":"W1N");
    return 0;
}
```

KLM 位运算加、乘、幂

题目分析

首先考虑加法怎么完成。

或运算也叫半加运算，我们可以得到不考虑进位时$a+b$的结果。

当两个数某一位都是1，那么这一位进位。很巧的是这对应与运算。

于是我们得到一个可以递归的关系$a+b=(a\oplus b)+((a\&b)<<1)$。

有了加法，乘法（a个b相加）就可以仿照快速幂的方式完成.

次幂再用一次普通快速幂即可

时间复杂度：log^3N（M题）

参考代码

```
#include<stdio.h>
unsigned long long a,b;
//三个函数分别对应+，*，^

unsigned long long getplus(unsigned long long x,unsigned long long y)
{
  if(x==0) return y;
  if(y==0) return x;
```

```
  getplus(x^y,(x&y)<<1);
}
unsigned long long getmulti(unsigned long long x,unsigned long long
y){
  unsigned long long ret=0;
  while(y!=0){
    if((y&1)==1) ret=getplus(ret,x);
    y>>=1;
    x=x<<1;
  }
  return ret;
}

unsigned long long getmi(unsigned long long x,unsigned long long y){
  unsigned long long ret=1;
  while(y!=0){
    if((y&1)==1) ret=getmulti(ret,x);
    y>>=1;
    x=getmulti(x,x);
  }
  return ret;
}

int main()
{
  scanf("%llu%llu",&a,&b);
  printf("%llu",getmi(a,b));
}
```

N 最大平均字段

题目分析

一个结论：答案子段长度不超过3

反证法：如果存在一个长度为$n(n>3)$的答案序列，那么把他分成长度为2和n-2的两段，其中必有一段的平均值大于等于它，矛盾。证毕。

简单模拟即可。

造数据的时候碰巧把二分精度卡了，不是故意的：）

时间复杂度：N

参考代码

```
#include<stdio.h>

double a[100005],ans;
int n;

int main()
{
  scanf("%d",&n);
  for(int i=1;i<=n;i++)
    scanf("%lf",&a[i]);
  for(int i=1;i<=n-1;i++)
    if((a[i]+a[i+1])/2>ans)
      ans=(a[i]+a[i+1])/2;
  for(int i=1;i<=n-2;i++)
    if((a[i]+a[i+1]+a[i+2])/3>ans)
      ans=(a[i]+a[i+1]+a[i+2])/3;
  printf("%.4f",ans);
}
```

o lxy不疑惑

题目分析

结论题：$ans = a * b - a - b$

此题出自NOIP2017小凯的疑惑，证明可自行百度。

这里推荐找规律的做法，规律比较明显。

时间复杂度：1

参考代码

```
#include<stdio.h>
int a,b;
int main()
{
  scanf("%d%d",&a,&b);
  printf("%lld",1ll*a*b-a-b);
}
```

P 自习安排

题目分析

依旧是结论题：$ans=(n-1)*a/b+1$

此题出自2022ICPC昆明站签到题。

给出一个简单证明。

假设答案是ans。那么一定存在第k_1天自习天数从$ans-1$变成ans。则需要满足存在k_1，使得$\frac{ans-1}{k_1-1}\leq x$，

并且不存在第k_2天，使自习天数从ans变成$ans+1$。则需要满足任意k_2，使得$\frac{ans}{k_2-1}>x$

根据k的范围，恒等变换可得$(n-1)x\leq ans<(n-1)x+1$。发现ans只能取到一个整数，即$(n-1)*a/b+1$。

时间复杂度：T

这两个结论题包括前面位运算等题目出的我觉得很有意思并且没什么门槛的题目，希望全部同学能体会到写代码的乐趣而不仅限于oi选手，不过尝试的同学不是很多呢。：(

参考代码

```
#include<stdio.h>
int main()
{
  long long n,a,b;
  while(~scanf("%lld%lld%lld",&n,&a,&b))
    printf("%lld\n",(n-1)*a/b+1);
}
```

Q 异或求和

题目分析

考虑一个最简单的情况：

$l=0,r=2^k-1,m\leq r$

此时的答案是$\sum_{i=0}^{r}i$，因为异或的值一定可以把所有数不重复地取完

当$l=0,r=2^k-1$

此时的答案是$\lfloor\frac{m}{r+1}\rfloor*(r+1)+\sum_{i=0}^{r}i$，也就是说$m$低位部分统计如上，超出$r$的高位部分统计了$r+1$次

当$l=t*2^k,r=t*2^k+2^k-1$

此时的答案是$(\lfloor\frac{m}{2^k}\rfloor\oplus l)*(r-l+1)+\sum_{i=0}^{2^k}i$，也就是超出$2^k-1$的高位部分还要和$r$的高位部分异或再统计。

当$l=0$时，$[l,r]$可以分成若干个$[l_i,r_i](l=t_i*2^i,r=t_i*2^i+2^i-1)$，将$[l,r]$全部覆盖。也就是把区间拆分按上述方法计算。

当$l\neq0$时，答案为$[0,r]-[0,l-1]$

时间复杂度：$TlogN$

参考代码

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

int m,T;

long long sol(long long x,long long y){
  if(x==-1) return 0;
  long long pre=0,ret=0;
  for(int i=40;i>=0;i--){
    pre|=(((x^y)>>i)&1)<<i;
    if(((x>>i)&1)==0) continue;
    long long l=pre^(1<<i),r=(pre^(1<<i))|((1<<i)-1);
    ret+=(l+r)*(r-l+1)/2;

  }
  return ret+(x^y);
}

int main()
{
  scanf("%d",&T);
  while(T--){
    int l,r;
    scanf("%d%d%d",&l,&r,&m);
    long long pre=0;
    printf("%lld\n",sol(r,m)-sol(l-1,m));
  }
}
```

R 因数求和

题目分析

先用一个数组，每个位置记录该下标有多少个数，令$n = max\{a_i\}$。

考虑因子x是多少个数的因子：$\sum_{i=1}^{\frac{n}{x}} a_{i*x}$。

那么答案中的贡献就是$x * \sum_{i=1}^{n/x} a_{i*x}$。

从1到n枚举x，统计答案。

时间复杂度：

每个x需要循环$\frac{n}{x}$次，所以一共循环次数是$\sum_{x=1}^{n} \frac{n}{x}$，这个级数和$nlogn$等价

$NlogN$

有的同学优化枚举因数，先筛出质数再根号求答案，我觉得很不错。

参考代码

```
#include<stdio.h>

int n;
int a[1000005];
long long ans;

int main()
{
  scanf("%d",&n);
  for(int i=1,x;i<=n;i++){
    scanf("%d",&x);
    a[x]++;
  }
  for(int i=2;i<=1000000;i++)
    for(int j=i;j<=1000000;j+=i)
      ans+=1ll*i*a[j];
  ans+=n;
  printf("%lld",ans);
```

```
}
```

s 时间与空间

题目分析

本题只是想给大家介绍介绍主定理。学会了主定理，代码其实非常简单。

参考代码

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

void output(double x,double y)
{
    if((x == 0) && (y == 0))
    {
        printf("1\n");
    }
    else if(x == 0)
    {
        if(y == 1)
        {
            printf("log(n)\n");
        }
        else
        {
            printf("log^%.3lf(n)\n", y);
        }
    }
    else if(y == 0)
    {
        if(x == 1)
        {
            printf("n\n");
        }
```

```
        else
        {
            printf("n^%.3lf\n", x);
        }
    }
    else
    {
        if(x == 1)
        {
            printf("n");
        }
        else
        {
            printf("n^%.3lf", x);
        }
        if(y == 1)
        {
            printf("log(n)\n");
        }
        else
        {
            printf("log^%.3lf(n)\n", y);
        }
    }
}

int main()
{
    double a,b,x,y;
    scanf("%lf%lf%lf%lf",&a,&b,&x,&y);
    double ans=log(b)/log(a);
    if(x<ans)
    {
        output(ans,0);
    }
    else if(x==ans)
    {
        output(x,y+1);
    }
```

```
    else
    {
        output(x,y);
    }
}
```

T 计算连分数

题目分析

只要熟悉连分数算法“取倒数取整”的过程，这道题目就不难写出。

参考代码

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

long long n;
double sqn;

long long gcd(long long a,long long b)
{
    if(b==0)
    {
        return a;
    }
    return gcd(b,a%b);
}

struct rational
{
    long long p,q;
};

struct quaderic
{
```

```
    struct rational a,b;
};

int main()
{
    int length;
    while(~scanf("%d%d",&n,&length))
    {
        sqn=sqrt((double)n);
        struct quaderic r;
        r.a.p=0;
        r.a.q=1;
        r.b.p=1;
        r.b.q=1;
        int i;
        for(i=0;i<=length;i++)
        {
            long long a=(long long)((double)r.a.p/(double)r.a.q+
(double)r.b.p/(double)r.b.q*sqn);
            printf("%lld ",a);
            struct quaderic s;
            s.a.p=r.a.p-a*r.a.q;
            s.a.q=r.a.q;
            s.b=r.b;
            struct rational no1;
            no1.p=s.a.p*s.a.p;
            no1.q=s.a.q*s.a.q;
            struct rational no2;
            no2.q=s.b.q*s.b.q;
            long long gc1=gcd(n,no2.q);
            no2.q=no2.q/gc1;
            no2.p=n/gc1*s.b.p*s.b.p;
            struct rational no;
            no.q=no1.q*no2.q;
            no.p=no1.p*no2.q-no2.p*no1.q;
            long long gc2=gcd(no.p,no.q);
            no.p=no.p/gc2;
            no.q=no.q/gc2;
            long long gc3=gcd(s.a.p,no.p);
```

```
            long long gc4=gcd(s.a.q,no.q);
            long long temp1=s.a.p/gc3;
            long long temp2=no.p/gc3;
            long long temp3=s.a.q/gc4;
            long long temp4=no.q/gc4;
            r.a.p=temp1*temp4;
            r.a.q=temp3*temp2;
            long long gc5=gcd(s.b.p,no.p);
            long long gc6=gcd(s.b.q,no.q);
            long long temp5=s.b.p/gc5;
            long long temp6=no.p/gc5;
            long long temp7=s.b.q/gc6;
            long long temp8=no.q/gc6;
            r.b.p=-temp5*temp8;
            r.b.q=temp7*temp6;
        }
        printf("\n");
    }
}
```

U 苍穹一粟

题解链接