数论模板

1. 欧几里得定理
2. 扩展欧几里得定理
3. 质数、筛质数
4. 欧拉函数
5. 中国剩余定理
6. 组合数
7. 博弈论
8. 高斯消元
9. 矩阵乘法
10. 莫比乌斯函数
11. 卡特兰数
12. 快速幂、龟速乘

1.欧几里得定理

gcd（a，b）=gcd(b , a % b)

首先我们可以得出 d|a ， d|b ,那么一定有 d|ax+by

a%b相当于a-[a/b]*b

=a-kb

因为a能被d整除，b能被d整除，那a-kb一定能被d整除

//也可以这么理解：a相当于几倍的d，b也相当于几倍d，a-kb后还是会剩几倍d

```
int gcd(int a,int b)
{
    return b?gcd(b,a%b):a;
}
```

2.扩展欧几里得定理

裴蜀定理：对于任意正整数a、b，一定存在非零整数x、y，使得

$ax+by=gcd（a，b）$

附：可以证明出：对于所有非零整数的x，y的值，ax+by一定会等于gcd（a，b）的倍数，这个可以用脑子想一想。

对于裴蜀定理，这个可以用扩展欧几里得的构造法来证明：如果我们对于所有的a，b，都能构造出出一个x、y，那这个定理肯定就是正确的

思路，要求ax+by=gcd（a，b）

先求出子问题的gcd（a，b）的x和y。最终递归的答案一定能求出x和y

ax+by=gcd(a,b)=gcd(b,a%b)

则肯定也满足bx`+(a%b)y`

=bx`+(a-[a/b]*b)y`

=ay`+b(x`-[a/b]y`)

=gcd(a,b)=gcd(b,a%b)

可知下一层的x`,y`与x，y的关系： x=y` y=x`-[a/b]y`

递归到最后的那个关系可知。由于b=0 所以可求得一组解：x=1，y=0

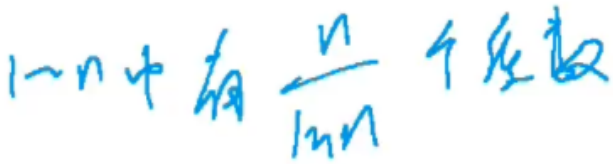
可根据这个作为下面一层的解来算上面这一层。。。直到算到当前这一层。（运用递归的方法

```
int exgcd(int a,int b,int &x,int &y)//ax+by=gcd（a，b）
{
  if(b==0)
  {
    x=1,y=0;
    return a;
  }
  else
  {
    int x1,y1;
    int gcd=exgcd(b,a%b,x1,y1);//用指针来解x1，y1
    x=y1;
```

```
    y=x1-(a/b)*y1;
    return gcd;
  }
}
```

3.质数、筛质数

质数基本定理：1~n中有$\frac{n}{\ln n}$个质数



埃氏筛（o（nloglogn））

```
#include<iostream>
using namespace std;
const int N=1000004;
int primes[N];
int cnt;
int st[N];
void get_primes(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;

            if(i<=n/i)//避免i*i爆long long
            for(int j=i*i;j<=n;j+=i)
            {
              //printf("j=%d\n",j);
                st[j]=true;
            }
        }
    }
}

int main()
{
    int n;
    scanf("%d",&n);

    get_primes(n);
```

```
    printf("%d",cnt);
}
```

线性筛 （o（n））

每个数x只会被它的最小质因子筛去

primes[j]*i会一直被筛去，直到i%primes[j]==0

为什么呢——

①当还没能i%primes[j]的时候，由于primes[j]小于i的最小质因子，所以primes[j] 是i*primes[j] 的的最小质因子

②直到i%primes[j]的时候，此时primes[j]就是i的最小质因子，所以有primes[j] 是i*primes[j] 的最小质因子

```
#include<iostream>
using namespace std;
const int N=1000004;
int primes[N];
int cnt;
int st[N];
void get_primes(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
          primes[cnt++]=i;
    }
                                    //这句话是确保primes[j]和i相乘不会爆n
    for(int j=0;primes[j]<=n/i;j++) //1.当i为质数的时候,primes[j]=i时会break
    {                               //2.当i是合数的时候，primes[j]为i的最小质因子的时候，会break
      st[primes[j]*i]=true;

      if(i%primes[j]==0)  break;
    }
    }
}

int main()
{
    int n;
    scanf("%d",&n);

    get_primes(n);

    printf("%d",cnt);
}
```

4.欧拉函数

欧拉函数的定义

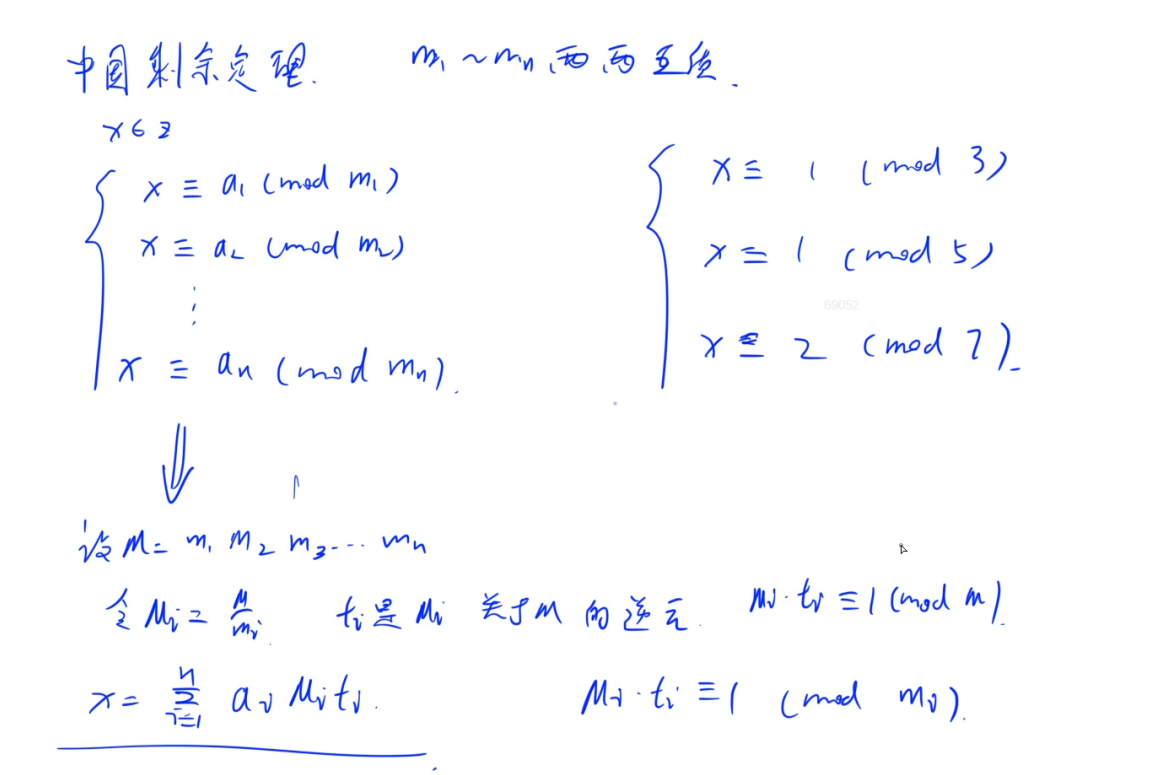
> $1 \sim N$ 中与 N 互质的数的个数被称为欧拉函数，记为 $\phi(N)$。
> 若在算数基本定理中，$N = p_1^{a_1} p_2^{a_2} \dots p_m^{a_m}$，则：
> $\phi(N) = N \times \frac{p_1-1}{p_1} \times \frac{p_2-1}{p_2} \times \dots \times \frac{p_m-1}{p_m}$

筛法求欧拉函数

```
void init(int n)
{
  phi[1]=1;
  for(int i=2;i<=n;i++)
  {
    if(!st[i])
    {
      phi[i]=i-1;
      primes[cnt++]=i;
    }

    for(int j=0;primes[j]<=n/i;j++)
    {
      st[primes[j]*i]=true;
      if(i%primes[j]==0)
      {
        phi[i*primes[j]]=phi[i]*primes[j];
        break;
      }
      else
      phi[i*primes[j]]=phi[i]*(primes[j]-1);
    }
  }
}
```

5.中国剩余定理



上图x并未求出最优的解。因为扩展欧几里得只是求出t_i的一组解

(x%M+M)%M。如果是要求最小的解，那么x+kM都是一组解（加上的M求模m_i都是等于0）

```
#include<iostream>
#include<algorithm>
#include<string.h>
using namespace std;
const int N=20;
typedef long long ll;
int n;
ll a[N];
ll m[N];
void exgcd(ll a,ll b,ll &x,ll &y)
{
  if(b==0)
  {
    x=1;
    y=0;
    return ;
  }
  else
  {
    ll x1,y1;
    exgcd(b,a%b,x1,y1);
    x=y1;
    y=x1-(a/b)*y1;
  }
}
int main()
{
```

```
  ll M=1;
  scanf("%d",&n);
  for(int i=1;i<=n;i++)
  {
    scanf("%lld%lld",&m[i],&a[i]);
    M*=m[i];
  }
  ll res=0;
  for(int i=1;i<=n;i++)
  {
    ll Mi=M/m[i];
    ll ti,y;
    exgcd(Mi,m[i],ti,y);
    res+=a[i]*Mi*ti;
  }
  printf("%lld",(res%M+M)%M);
}
```

```
ll excrt(ll m1,ll m2,ll a1,ll a2)
{
    ll k1,k2;
    ll d=exgcd(a1,-a2,k1,k2);
    k1=k1*(m2-m1)/d;
    k1=(k1%(a2/d)+(a2/d))%(a2/d);
    m1=m1+a1*k1;
    a1=abs(a1/d*a2);

    m1=(m1%a1+a1)%a1;
    if(m1==0)
    m1+=a1;
    return m1;
}
```

6.组合数

▼ I 利用递推求组合数

时间复杂度为$O(n^2)$，只能处理5000以内的组合数

利用公式 $C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$

```
void enit()
{
  for(int i=0;i<=N;i++)
  {
    for(int j=0;j<=i;j++)
    {
      if(j==0) c[i][j]=1;
      else c[i][j]=(c[i-1][j-1]+c[i-1][j])%mod;
    }
  }
}
```

▼ II预处理阶乘，利用逆元求组合数

时间复杂度为$O(nlogn)$

可以用这个来算组合数和排列数 $C_n^m = \frac{n!}{m!(n-m)!}$ $A_n^m = \frac{n!}{(n-m)!}$

先预处理下所有数的阶乘，以及阶乘的逆元。然后再用这个算组合数就可以了

```
#include<iostream>
#include<algorithm>
using namespace std;
typedef long long ll;
const int N=100005,mod=1e9+7;
int fact[N],infact[N];
int qmi(int a,int b)
{
    int res=1;
    while(b)
    {
        if(b&1)
        res=(ll)res*a%mod;

        a=(ll)a*a%mod;
        b>>=1;
    }
    return res;
}
void init()
{
    fact[0]=infact[0]=1;
    for(int i=1;i<=N;i++)
    {
        fact[i]=(ll)fact[i-1]*i%mod;
        infact[i]=qmi(fact[i],mod-2);
    }
}
int C(int n,int m)
{
    return (ll)fact[n]*infact[m]%mod*infact[n-m]%mod;
}
int main()
{
    init();

    int n;
    scanf("%d",&n);
    while(n--)
    {
        int a,b;
        scanf("%d%d",&a,&b);
        printf("%d\n",C(a,b));
    }
}
```

▼ III利用卢卡斯定理求组合数

时间复杂度为$O(p\,log_n^p)$

$C_n^m = C_{n \mod p}^{m \mod p} \times C_{n/p}^{m/p}$ (p要为质数)

```
#include<iostream>
#include<algorithm>
using namespace std;
typedef long long ll;
int p;
int qmi(int a,int b)
{
  int res=1;
  while(b)
  {
    if(b&1)
    res=(ll)res*a%p;

    a=(ll)a*a%p;
    b>>=1;
  }
  return res;
}
int C(int n,int m)
{
  int res=1;
  for(int i=n,j=1;j<=m;j++,i--)
  {
    res=(ll)res*i%p;
    res=(ll)res*qmi(j,p-2)%p;
  }
  return res;
}
ll lucas(ll n,ll m)
{
  if(n<p&&m<p) return C(n,m);
  return (ll)C(n%p,m%p)*lucas(n/p,m/p)%p;
}
int main()
{
  int T;
  scanf("%d",&T);
  while(T--)
  {
    ll n,m;
    scanf("%lld%lld%d",&n,&m,&p);
    printf("%d\n",lucas(n,m));
  }
}
```

▼ IV高精度求组合数

将$C_n^m = \frac{n!}{m!(n-m)!}$本身需要算高精度乘法、高精度除法。可通过转化为求$p_1^{c1}\ p_2^{c2}\ p_3^{c3}.....p_k^{ck}$，即质数的相乘，就只需要实现一个高精度乘法就行了

①用线性筛来储存每一个质数

②用$\lfloor \frac{n}{p} \rfloor + \lfloor \frac{n}{p^2} \rfloor + \lfloor \frac{n}{p^3} \rfloor + ...$来算每个质数的次幂

③利用高精度算每个质数的乘法就行了

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
const int N=5005;
int primes[N];
bool st[N];
int cnt;
int sum[N];//储存每个质数的次幂
void get_primes(int n)
{
  for(int i=2;i<=n;i++)
  {
    if(!st[i])
    primes[cnt++]=i;

    for(int j=0;primes[j]<=n/i;j++)
    {
      st[primes[j]*i]=true;

      if(i%primes[j]==0)
      break;
    }
  }
}
int get(int n,int p)
{
  int res=0;
  while(n)
  {
    res+=n/p;
    n/=p;
  }
  return res;
}
vector<int> num(vector<int> a,int b)
{
  vector<int> c;
  int t=0;
  for(int i=0;i<a.size();i++)
  {
    t+=a[i]*b;
    c.push_back(t%10);
    t/=10;
  }

  while(t)
  {
    c.push_back(t%10);
```

```
    t/=10;
  }

  return c;
}
int main()
{
  int n,m;
  scanf("%d%d",&n,&m);
  get_primes(n);

  for(int i=0;i<cnt;i++)
  {
    int p=primes[i];
    sum[i]=get(n,p)-get(m,p)-get(n-m,p);
  }
  vector<int> res;
  res.push_back(1);
  for(int i=0;i<cnt;i++)
  {
    for(int j=0;j<sum[i];j++)
    {
      res=num(res,primes[i]);
    }
  }

  for(int i=res.size()-1;i>=0;i--)
  printf("%d",res[i]);
}
```

7.博弈论

简单取石子的结论：

初始时每一堆石子数的异或值等于0，先手胜；反之，后手胜

sg函数

893. 集合-Nim游戏

题目 | 提交记录 | 讨论 | 题解 | 视频讲解

给定 n 堆石子以及一个由 k 个不同正整数构成的数字集合 S。

现在有两位玩家轮流操作，每次操作可以从任意一堆石子中拿取石子，每次拿取的石子数量必须包含于集合 S，最后无法进行操作的人视为失败。

问如果两人都采用最优策略，先手是否必胜。

```
#include<iostream>
#include<algorithm>
#include<string.h>
#include<unordered_set>
using namespace std;
const int N=105,M=10005;
int n,m;
int a[N],f[M];
int sg(int x)
{
    if(f[x]!=-1) return f[x];

    unordered_set<int> S;
    for(int i=0;i<m;i++)
    {
        if(x-a[i]>=0)
        S.insert(sg(x-a[i]));
    }

    for(int i=0;;i++)
    {
        if(!S.count(i))
        return f[x]=i;
    }
}
int main()
{
    cin>>m;
    for(int i=0;i<m;i++)    cin>>a[i];

    memset(f,-1,sizeof f);
    int res=0;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        int x;
        cin>>x;
        res^=sg(x);
    }
    if(res) puts("Yes");
    else puts("No");
}
```

8.高斯消元

高斯消元解线性方程组

```
#include<iostream>
#include<algorithm>
#include<math.h>
using namespace std;
const int N=105;
const double eps=1e-6;
int n;
double a[N][N];
int gauss()
{
  int c=0,r=0;//列，行
  for(int c=0;c<n;c++)//每一列开始遍历
  {
    int t=r;
    for(int i=r;i<n;i++)
    {
      if(fabs(a[i][c])>fabs(a[t][c]))
      t=i;
    }
    if(fabs(a[t][c])<eps)
    continue;

    for(int j=c;j<=n;j++)
    swap(a[r][j],a[t][j]);

    for(int j=n;j>=c;j--)
    a[r][j]/=a[r][c];

    for(int i=r+1;i<n;i++)
    {
      if(fabs(a[i][c])>eps)
      for(int j=n;j>=c;j--)
      {
        a[i][j]-=a[r][j]*a[i][c];
      }
    }
    r++;
  }
  if(r<n)
  {
    for(int i=r;i<n;i++)
    if(fabs(a[i][n])>eps)
    return 2;//代表无解
    else
    return 1;//代表有无数多个解
  }

  for(int i=n-1;i>=0;i--)
```

```
  {
    for(int j=i+1;j<n;j++)
    {
      a[i][n]-=a[i][j]*a[j][n];
    }
   }
   return 0;//代表有唯一解
}
int main()
{
  cin>>n;
  for(int i=0;i<n;i++)
    for(int j=0;j<n+1;j++)
      cin>>a[i][j];

  int t=gauss();

  if(t==0)
  {
    for(int i=0;i<n;i++)
    {
      printf("%.2lf\n",a[i][n]);
    }
  }
  else if(t==1)//有无数多个解
  puts("Infinite group solutions");
  else if(t==2)//无解
  puts("No solution");
}
```

高斯消元解异或线性方程组（如解决开关问题）

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N=105;
int a[N][N];
int n;
int gauss()
{
  int r=0,c=0;
  for(int c=0;c<n;c++)//从第0列到第n-1列
  {
    int t=r;
    for(int i=r;i<n;i++)
    {
      if(a[i][c]==1)
      {
        t=i;
        break;
      }
    }

    if(a[t][c]==0)
    continue;
```

```
    for(int j=c;j<=n;j++)
    {
      swap(a[r][j],a[t][j]);
     }
    //swap(a[t][c],a[r][c]);

    for(int i=r+1;i<n;i++)
    {
      if(a[i][c]==1)
      for(int j=n;j>=c;j--)
      {
        a[i][j]=a[i][j]^a[r][j];
      }
    }
    r++;
  }

  if(r<n)
  {
    for(int i=r;i<n;i++)
    {
      if(a[i][n]==1)
      return 2;//无解
    }
    return 1;//无限多个解
  }

  for(int i=n-1;i>=0;i--)
  {
    for(int j=i+1;j<n;j++)
    {
      a[i][n]=a[i][n]^(a[i][j]*a[j][n]);
    }
  }
  return 0;
}
int main()
{
  cin>>n;
  for(int i=0;i<n;i++)
    for(int j=0;j<n+1;j++)
      cin>>a[i][j];

  int t=gauss();
  if(t==0)
  {
    for(int i=0;i<n;i++)
    printf("%d\n",a[i][n]);
  }
  else if(t==1)
  puts("Multiple sets of solutions");
  else
  puts("No solution");
}
```

9.矩阵乘法

斐波那契数列的前n项和

```
#include<iostream>
#include<algorithm>
#include<string.h>
using namespace std;
typedef long long ll;
const int N=3;
int n,m;
void mul(int c[],int a[],int b[][N])
{
  int temp[N]={0};
  for(int i=0;i<N;i++)
  {
    for(int j=0;j<N;j++)
    {
      temp[i]=(temp[i]+(ll)a[j]*b[j][i])%m;
    }
  }
  memcpy(c,temp,sizeof temp);
}
void mul(int c[][N],int a[][N],int b[][N])
{
  int temp[N][N]={0};
  for(int i=0;i<N;i++)
  {
    for(int j=0;j<N;j++)
    {
      for(int k=0;k<N;k++)
      {
        temp[i][j]=(temp[i][j]+(ll)a[i][k]*b[k][j])%m;
      }
    }
   }
   memcpy(c,temp,sizeof temp);
}
int main()
{
  int F[N]={1,1,1};//[f1,f2,s1]
  int A[N][N]={
    {0,1,0},
    {1,1,1},
    {0,0,1}
  };
  //可得Fn=f1 * A^(n-1)
  scanf("%d%d",&n,&m);
  n--;
  while(n)
  {
    if(n&1)
    mul(F,F,A);

    mul(A,A,A);
    n>>=1;
  }
  printf("%d\n",F[2]);
}
```

10.莫比乌斯函数

▼ 题目大意

达达正在破解一段密码，他需要回答很多类似的问题：

对于给定的整数 a,b 和 d，有多少正整数对 x,y，满足 x≤a，y≤b，并且 gcd(x,y)=d。

作为达达的同学，达达希望得到你的帮助。

输入格式

第一行包含一个正整数 n，表示一共有 n 组询问。

接下来 n 行，每行表示一个询问，每行三个正整数，分别为 a,b,d。

输出格式

对于每组询问，输出一个正整数，表示满足条件的整数对数。

数据范围

1≤n≤50000,1≤d≤a,b≤50000

输入样例：

```
2
4 5 2
6 4 3
```

输出样例：

```
3
2
```

(做映射：x`=x/d,y`=y/d)

题意可以传化为$x \leq \frac{a}{d}, y \leq \frac{b}{d}$,并且$gcd\ (x, y)\ = 1$

设a=a/d，b=b/d

$x \le a, y \le b,$并且$gcd\ (x,y)\ = 1$

利用容斥原理：a×b-最大公约数不为1的整数对数

目标1：求最大公约数不为1的整数对数

莫比乌斯函数：

$$\mu(n) = \begin{cases} 1 & \text{若 } n = 1\ ; \\ (-1)^k & \text{若 } n \text{ 无平方数因数，且 } n = p_1 p_2 \ldots\ldots p_k\ ; \\ 0 & \text{若 } n \text{ 有大于 } 1 \text{ 的平方数因数。} \end{cases}$$

答案=$ab - \frac{a}{2}\frac{b}{2} - \frac{a}{3}\frac{b}{3} + \ldots\ldots + \frac{a}{6}\frac{b}{6}\ldots\ldots$

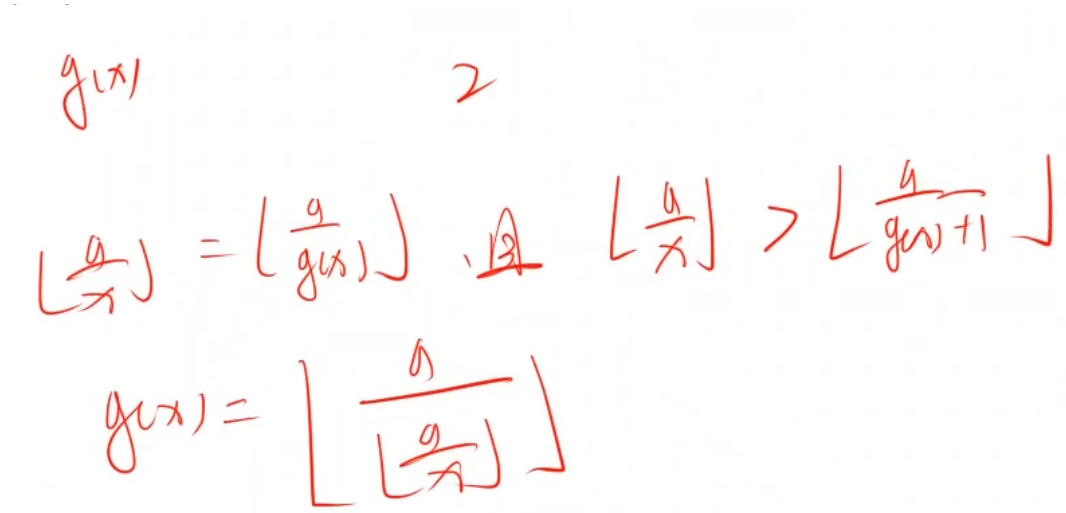
~~（ps：不能除以4，因为最大公约数为4的整数对数在2里面已经除过的）~~

> $\frac{a}{1}\ \frac{a}{2}\ \frac{a}{3}\ldots\ldots\frac{a}{a}$ **中最多会有** $2\sqrt{a}$ **个不同的数**
>
> 证明如下：
> $\frac{a}{1}\ \frac{a}{2}\ldots\ldots\frac{a}{\lfloor\sqrt{a}\rfloor}$中最多有$\sqrt{a}$个不同的数。
> 又$\frac{a}{\lfloor\sqrt{a}\rfloor+1}\ \frac{a}{\lfloor\sqrt{a}\rfloor+2}\ldots\ldots\frac{a}{a}$中的值最多是从$\sqrt{a}-1$到1的，所以这之中最多也只有$\sqrt{a}$个不同的数。
> 所以命题得证。

目标2：找一段中有多少数是相同

$\frac{a}{x}\ \frac{a}{x+1}\ \ldots\ \frac{a}{g(x)}$都为同一个数

如$\frac{24}{9}\ \frac{24}{10}\ \frac{24}{11}\ \frac{24}{12}$



```
#include<iostream>
#include<algorithm>
using namespace std;
typedef long long ll;
const int N=50005;
int primes[N];
bool st[N];
int cnt;
int mobius[N];
int sum[N];
void init(int n)
{
  mobius[1]=1;
  for(int i=2;i<=n;i++)
  {
    if(!st[i])
    {
      primes[cnt++]=i;
      mobius[i]=-1;
    }
    for(int j=0;primes[j]<=n/i;j++)
    {
      int t=primes[j]*i;
      st[t]=true;
      if(i%primes[j]==0)//说明t中的pj质数肯定>1，所以莫比乌斯函数应该等于0
      {
        mobius[t]=0;
        break;
      }
      else
      mobius[t]=mobius[i]*-1;//相比于i多乘了个质数pj，所以质数的个数肯定是i多一个。
    }
  }

  for(int i=1;i<=n;i++)
  sum[i]=mobius[i]+sum[i-1];
}
```

```
int main()
{
  init(N-1);
  int T;
  scanf("%d",&T);
  while(T--)
  {
    int a,b,d;
    scanf("%d%d%d",&a,&b,&d);
    a/=d;
    b/=d;
    int n=min(a,b);
    ll res=0;
    for(int l=1,r;l<=n;l=r+1)
    {
      r=min(n,min(a/(a/l),b/(b/l)));

      res+=(sum[r]-sum[l-1])*(ll)(b/r)*(a/r);
    }
    printf("%lld\n",res);
  }
}
```

莫比乌斯反演：见lwy的数论笔记，略

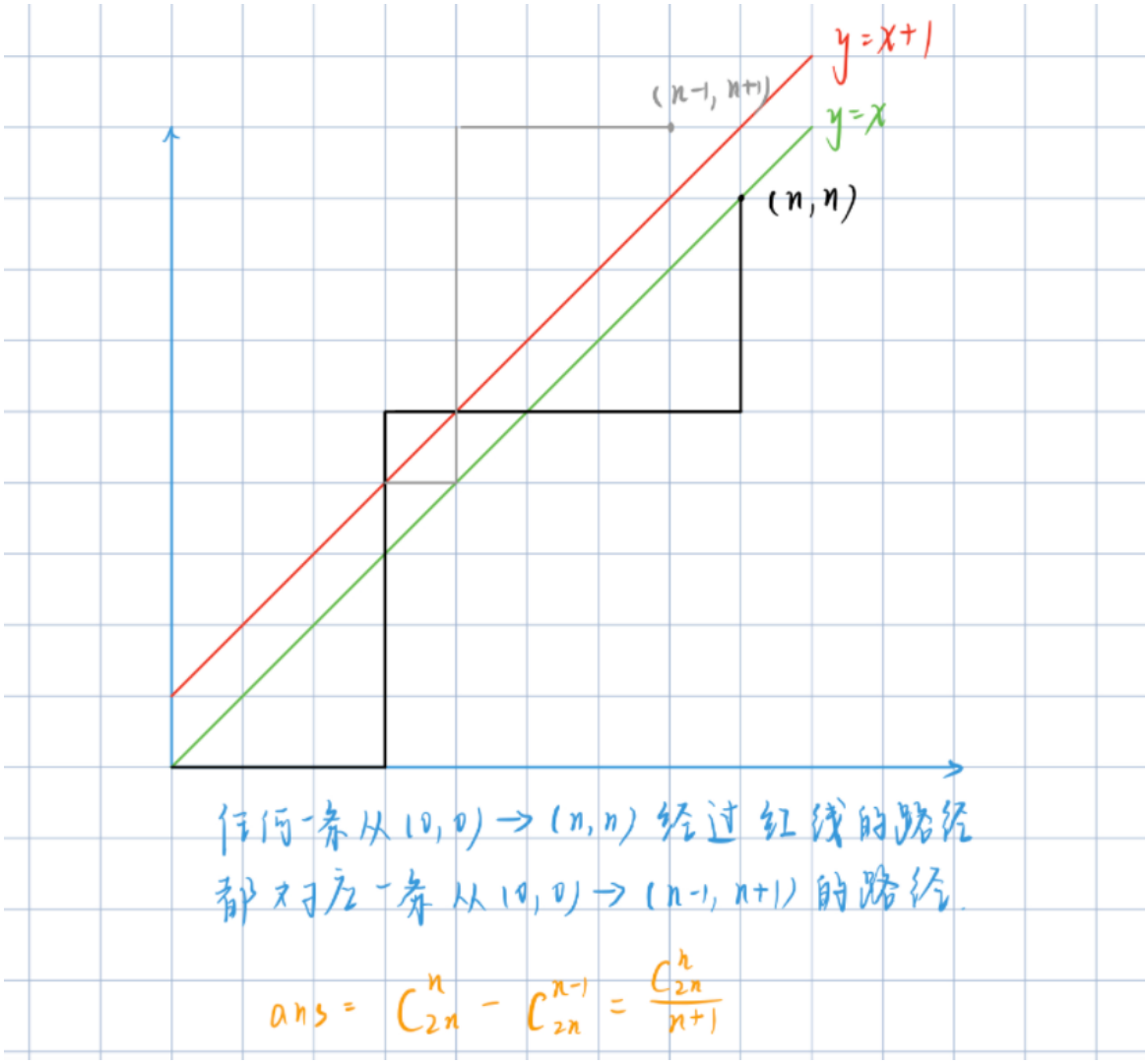
11.卡特兰数

题目描述：给定 n 个 0 和 n 个 1，它们将按照某种顺序排成长度为 2n 的序列，求它们能排列成的所有序列中，能够满足任意前缀序列中 0 的个数都不少于 1 的个数的序列有多少个。答案对 10^9+7 取模。

数据范围：1≤n≤10^5
组合计数，卡特兰数

解法
将 01 序列置于坐标系中，起点定于原点。若 0 表示向右走，1 表示向上走，那么任何前缀中 0 的个数不少于 1 的个数就转化为，路径上的任意一点，横坐标大于等于纵坐标。题目所求即为这样的合法路径数量。

下图中，表示从 (0,0) 走到 (n,n) 的路径，在绿线及以下表示合法，若触碰红线即不合法。



由图可知，任何一条不合法的路径（如黑色路径），都对应一条从 (0,0) 走到 (n−1,n+1)(n−1,n+1) 的一条路径（如灰色路径）。而任何一条 (0,0) 走到 (n−1,n+1)的路径，也对应了一条从 (0,0)走到 （n,n） 的不合法路径。

答案如图，即卡特兰数。

12.快速幂、龟速乘

快速幂

```
#include <iostream>
#include <algorithm>

using namespace std;

typedef long long LL;
```

```
LL qmi(int a, int b, int p)
{
    LL res = 1 % p;
    while (b)
    {
        if (b & 1) res = res * a % p;
        a = a * (LL)a % p;
        b >>= 1;
    }
    return res;
}


int main()
{
    int n;
    scanf("%d", &n);
    while (n -- )
    {
        int a, b, p;
        scanf("%d%d%d", &a, &b, &p);
        printf("%lld\n", qmi(a, b, p));
    }

    return 0;
}
```

龟速乘

```
ll qadd(ll a,ll b,ll p)
{
  ll res=0;
  ll base=a;
  while(b)
  {
    if(b&1)
    res=(res+base)%p;

    base=base*2%p;

    b>>=1;
  }

  return res;
}
```

数论更多的还是靠自己推结论，加油鸭~！