模板

乱七八糟

模非质数时取余技巧

如果分母较小，可以先余模数乘分母，也就是(x/b)%a=x%(ab)/b，最后在结果出再除分母

01字典树

```
struct wordtree
{
    #define pb push_back
    vector<int>T,L,R;
    int tot;
    void init(){tot=0;T.pb(0);L.pb(-1);R.pb(-1);}
    void Clear(){vector<int>().swap(T);vector<int>().swap(L);vector<int>().swap(R);}
    void down(){T.pb(0);L.pb(-1);R.pb(-1);}
    void insert(int u,int v,int pos){
        for(int i=pos;;i--){
            T[u]++;
            if(!i)break;
            if(v&(1<<(i-1))){
                if(R[u]==-1){R[u]=++tot;down();}
                u=R[u];
            }else{
                if(L[u]==-1){L[u]=++tot;down();}
                u=L[u];
            }
        }
    }
    void delet(int u,int v,int pos)
    {
        for(int i=pos;;i--){
            T[u]--;
            if(!i)break;
            if(v&(1<<(i-1))) u=R[u];
            else u=L[u];
        }
    }
    void ask(int u,int pos,int & x)
    {
        for(int i=pos;;i--){
            if(!i||u==-1)break;
            if(x&(1<<(i-1))){
                if(L[u]!=-1&&T[L[u]])u=L[u];
                else u=R[u],x^=(1<<(i-1));
            }else{
                if(R[u]!=-1&&T[R[u]])u=R[u],x^=(1<<(i-1));
                else u=L[u];
            }
        }
```

```
    }
}tree;
```

线性基

```
struct Linear_Basis
{
    LL b[63],nb[63],tot;
    void init(){
        tot=0;
        memset(b,0,sizeof(b));
        memset(nb,0,sizeof(nb));
    }
    bool ins(LL x){
        for(int i=62;i>=0;i--)
            if (x&(1LL<<i)){
                if (!b[i]) {b[i]=x;break;}
                x^=b[i];
            }
        return x>0;
    }
    LL Max(LL x){
        LL res=x;
        for(int i=62;i>=0;i--)
            res=max(res,res^b[i]);
        return res;
    }
    LL Min(LL x)
    {
        LL res=x;
        for(int i=0;i<=62;i++)
            if (b[i]) res^=b[i];
        return res;
    }
    void rebuild()
    {
        for(int i=62;i>=0;i--)
            for(int j=i-1;j>=0;j--)
                if (b[i]&(1LL<<j)) b[i]^=b[j];
        for(int i=0;i<=62;i++)
            if (b[i]) nb[tot++]=b[i];
    }
    LL Kth_Max(LL k){
        LL res=0;
        for(int i=62;i>=0;i--)
            if (k&(1LL<<i)) res^=nb[i];
        return res;
    }
} LB;
```

树状数组

```
int lowbit(int x){return x&-x;}
void change(int x,int p){while(x<N)t[x]+=p,x+=lowbit(x);}
int ask(int x){int ans=0;while(x>0)ans+=t[x],x-=lowbit(x);return ans;}
int sum(int l,int r){return ask(r)-ask(l-1);}
int find_kth(int k)//第k小
{
    int ans=0,res=0;
    for(int i=20;i>=0;i--){
        ans+=(1<<i);
        if(ans>=N||res+t[ans]>=k) ans-=(1<<i);
        else res+=t[ans];
    }
    return ans+1;
}
```

字符串hash

$h[i] = h[i-1] * p + val[i]$

h[i...j] = $((h[j] - h[i-1] * p^{j-i+1})$%mod + mod)%mod

p = 233 mod = 1e9+9

ST表

```
int f[25][N],num[30];
void init_ST(int n,int *a)
{
    for(int i=1;i<=n;i++)f[0][i]=a[i];
    for(int i=0;i<=25;i++)num[i]=(1<<i);
    int k=log2(n)+1;
    for(int i=1;i<=k;i++)
        for(int j=1;j+num[i-1]<=n;j++)
            f[i][j]=max(f[i-1][j],f[i-1][j+num[i-1]]);
}
int ask(int l,int r){int k=log2(r-l+1);return max(f[k][l],f[k][r-num[k]+1]);}
```

Farey 序列

F1={0/1，1/1};

F2={0/1，1/2，1/1}

F3={0/1，1/3，1/2，2/3，1/1}

除了F1，其余Farey序列都有奇数个元素，并且中间值是1/2

Farey序列是一个对称序列，头尾之和为1

假如序列中有三个连续元素x1/y1, x2/y2, x3/y3，则有x2 = x1+x3; y2 = y1+y3;并且有x1*y2* – *x2*y1 = 1

Stern-Brocot树 （构成所有有理数）

算法的思想是由(0/1, 1/0)开始，然后不断地重复以下的操作：在相邻的两个分数m/n和m'/n'插入 (m+m')/(n+n')。

组合数学

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

$\sum_{i=0}^{k} C_n^i = C_{n+1}^{k+1}$

错排公式

D(n) = (n-1) [D(n-2) + D(n-1)]

D(n) = n! [(-1)^2/2! + ... + (-1)^(n-1)/(n-1)! + (-1)^n/n!].

圆排列

从n个不同元素中不重复地取出m（1≤m≤n）个元素在一个圆周上，叫做这n个不同元素的圆排列。如果一个m-圆排列旋转可以得到另一个m-圆排列，则认为这两个圆排列相同。

n个不同元素的m-圆排列个数N为 $\frac{n!}{(n-m)!\times m}$

lucas定理

C(n,m)%p=C(n/p,m/p)*C(n%p,m%p)%p

斯特林公式

$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$ 应用：求n!的位数

第一类斯特林数 s(p,k)

$s(p,k)$ ： p个人分成k组做环排列的方法数目

$s(p,k) = (p-1) * s(p-1,k) + s(p-1,k-1), 1 <= k <= p-1$

边界条件：$s(p,0) = 0, p >= 1$ $s(p,p) = 1, p >= 0$

应用：

求自然数幂和： $\sum_{i=0}^{n} i^k$

$P_n^k = n*(n-1)*...*(n-k+1)$

展开： $P_n^k = s(k,k)n^k - s(k,k-1)n^{k-1}...$

$P_n^k = \sum_{i=0}^{k}(-1)^{k-i}s(k,i)n^i$

所以第一类斯特林数是排列数公式展开式的系数

显然左边是k个只有0次和1次的系数为(1,-(i-1))的多项式相乘

可以用分治FFT在nlog^2(n)时间内求s(n,0),s(n,1),...,

设： $S_k(n) = \sum_{i=0}^{n} i^k$

$C_n^k = \frac{P_n^k}{k!} = \frac{\sum_{i=0}^{k}(-1)^{k-i}s(k,i)n^i}{k!}$

$n^k = C_n^k k! - \sum_{i=0}^{k-1}(-1)^{k-i}s(k,i)n^i$

$S_k(n) = \sum_{i=0}^{n} i^k$

$= \sum_{i=0}^{n}(C_i^k k! - \sum_{j=0}^{k-1}(-1)^{k-j}s(k,j)i^j)$

$= k!\sum_{i=0}^{n} C_i^k - \sum_{i=0}^{n}\sum_{j=0}^{k-1}(-1)^{k-j}s(k,j)i^j$

$= k!C_{n+1}^{k+1} - \sum_{j=0}^{k-1}(-1)^{k-j}s(k,j)S_j(n)$

$= \frac{P_{n+1}^{k+1}}{k+1} - \sum_{j=0}^{k-1}(-1)^{k-j}s(k,j)S_j(n)$

可以$O(k^2)$递推$s_k(n)$

第二类斯特林数 S(p,k)

$S(p,k)$： p个不同的小球放进k个相同的非空的盒子的方法数

$S(p,k) = k * S(p-1,k) + S(p-1,k-1), 1 <= k <= p-1$

边界条件： $S(p,p) = 1, p >= 0$ $S(p,0) = 0, p >= 1$

$S(n,m) = \frac{1}{m!}\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^n$

该式是个卷积，可以在nlogn时间内求出S(n,0),S(n,1),...

应用：

$n^k = \sum_{i=0}^{k} S(k,i)P_n^i$

$= \sum_{i=0}^{k} S(k,i)C_n^i i!$

$= \sum_{i=0}^{k} S(k,i)i!(C_{n-1}^i + C_{n-1}^{i-1})$

$= (n-1)^k + \sum_{i=1}^{k} S(k,i)i!C_{n-1}^{i-1}$

贝尔数

定义：第n个bell数表示集合{1,2,3,...,n}的划分方案数，$B[0] = 1$

$B_{n+1} = \sum_{k=0}^{n} C(n,k)B_k$

$B_n = \sum_{m=0}^{n} S(n,m)$

$= \sum_{m=0}^{n} \frac{1}{m!}\sum_{k=0}^{m}(-1)^k C(m,k)(m-k)^n$

$= \sum_{m=0}^{n}\sum_{k=0}^{m} \frac{(-1)^k}{k!} * \frac{(m-k)^n}{(m-k)!}$

设$A_i = \frac{(-1)^i}{i!}, B_i = \frac{i^n}{i!}$

$= \sum_{m=0}^{n}\sum_{k=0}^{m} A_k B_{m-k}$

$= \sum_{i=0}^{n} A_i \sum_{m=0}^{n-i} B_m$

nlogn预处理即可得到B_n，写法简便

指数生成函数：$\sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = e^{e^x-1}$

将$e^x - 1$泰特展开后多项式求exp即可在nlogn的时间内求贝尔数的前n项和

伯努利数

$B_0 = 1, B_n = -\frac{1}{n+1}(C_{n+1}^0 B_0 + C_{n+1}^1 B_1 + \ldots + C_{n+1}^{n-1} B_{n-1})$

指数生成函数：

$G_e(x) = \sum_{i=0}^{\infty} \frac{B_i}{i!} x^i = \frac{x}{e^x-1} = \frac{x}{\sum_{i=1}^{\infty} \frac{x^i}{i!}} = \frac{1}{\sum_{i=0}^{\infty} \frac{x^i}{(i+1)!}}$

可以用FFT多项式求逆在nlogn的时间内求前n项

应用：自然数幂求和

$S_m(n) = \sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} (-1)^k C(m+1, k) B_k n^{m+1-k}$

五边形数定理

$\phi(x) = \prod_{n=1}^{\infty}(1-x^n) = \sum_{-\infty}^{\infty}(-1)^k x^{\frac{k(3k-1)}{2}} = 1 + \sum_{k=1}^{\infty}(-1)^k x^{\frac{k(3k\pm1)}{2}}$

整数拆分问题

$P(x) = (1 + x^1 + x^2 + ..)(1 + x^2 + x^4 + \ldots)(1 + x^3 + x^6 + \ldots)\ldots$

$= \prod_{i=0}^{\infty} \frac{1}{1-x^i} = P_x(0)x^0 + P_x(1)x^1 + P_x(2)x^2 + \ldots + P_x(n)x^n + \ldots$

n的拆分数为$P_x(n)$

$\frac{1}{\phi(x)} = \sum_{i=0}^{\infty} P_x(i)x^i$

$1 = \phi(x) \sum_{i=0}^{\infty} P_x(i)x^i$

$= (1 - x - x^2 + x^5 + \ldots)(P_x(0) + P_x(1)x + P_x(2)x^2 + P_x(3)x^3 + \ldots)$

$P_x(n) = P_x(n-1) + P_x(n-2) - P_x(n-5) - P_x(n-7) + \ldots$

$\frac{i(3i-1)}{2} \le n$的i的个数不超过$\sqrt{n}$个 计算$P_x(n)$复杂度为$O(n\sqrt{n})$

约束条件1： 拆分出来的每种数的个数不能大于等于k

$(1 + x^1 + x^2 + \ldots + x^{k-1})(1 + x^2 + x^4 + \ldots + x^{2(k-1)}) = \prod_{i=0}^{\infty} \frac{1-x^{ki}}{1-x^i} = \frac{\phi(x^k)}{\phi(x)} = \phi(x^k)P(x)$

$\phi(x^k)P(x) = (1 - x^k - x^{2k} + x^{5k})(P_x(0) + P_x(1)x^1 + P_x(2)x^2 + P_x(3)x^3 + \ldots)$

$ans = F_k(n) = P_x(n) - P_x(n-k) - P_x(n-2k) + P_x(n-5k) + \ldots$

球盒问题（n个球，m个盒）

约束条件1： 球无区别 盒无区别 允许为空

整数拆分问题 答案为x^n的系数

约束条件2：球无区别 盒无区别 不允许为空

整数拆分问题 答案为x^{n-m}的系数

约束条件3：球无区别 盒有区别 允许为空

隔板法 答案为 $C(m+n-1,m-1)$

约束条件4：球无区别 盒有区别 不允许为空

隔板法 答案为 $C(n-1,m-1)$

约束条件5：球有区别 盒有区别 允许为空

指数型母函数 答案为 m^n

约束条件6：球有区别 盒有区别 不允许为空

指数型母函数 答案为 $\sum_{k=0}^{m}(-1)^kC(m,k)(m-k)^n$

约束条件7：球有区别 盒无区别 不允许为空

第二类斯特林数 答案为 $S(n,m)=\frac{1}{m!}\sum_{k=0}^{m}(-1)^kC(m,k)(m-k)^n$

约束条件8：球有区别 盒无区别 允许为空

第二类斯特林数 答案为 $S(n,1)+S(n,2)+S(n,3)+..+S(n,min(n,m))$

生成函数

当 $A(n+k)=C_1A(n+k-1)+C_2A(n+k-2)+...+C_kA(n)$

$A(n)$成为k阶常系数线性递推数列

多项式$x^k=C_1x^{k-1}+C_2x^{k-2}+...+C_k$为特征多项式

$A(n)=d_1x_1^n+d_2x_2^n+...+d_kx_k^n$

$d_1,d_2,...,d_k$为待定系数

如果有重根 $x_1=x_2=x_3$ 对应项为 $(d_1+d_2n+d_3n^2)x_1^n$

斐波那契数列

$S_n=f(0)+f(1)+...+f(n)=f(n+2)-1$

斐波那契数f(n)是偶数当且仅当n能被3整除

$f(n)=\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n-\frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$

$f(n)=C(n-1,0)+C(n-2,1)+...+C(n-t,t-1)\ t=\lfloor\frac{n+1}{2}\rfloor$

、

$(1-x)^{-k}=\sum_{n=0}^{\infty}C(k+n-1,k-1)x^n$

$(1+x^1+x^2+\ldots+x^{n-1})^m=(\frac{1-x^n}{1-x})^m=\frac{(1-x^n)^m}{(1-x)^m}=(1-x^n)^m(1-x)^{-m}$

$g_n(x)=1(1+x)(1+x+x^2)\ldots(1+x+x^2+\ldots+x^{n-1})=\frac{\prod_{j=1}^{n}(1-x^j)}{(1-x)^n}$

指数生成函数常用公式 $e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}+\ldots$

$g^{(e)}(x)=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots=e^x$

$g^{(e)}(x)=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\ldots=e^{-x}$

$g^{(e)}(x)=1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\ldots=\frac{e^x+e^{-x}}{2}$

$g^{(e)}(x)=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+\ldots=\frac{e^x-e^{-x}}{2}$

$ln(1+x)=x-\frac{1}{2}x^2+\frac{1}{3}x^3+o(x^3)$

$sinx=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5+o(x^5)$

$arcsinx=x+\frac{1}{2}\times\frac{x^3}{3}+\frac{1\times3}{2\times4}\times\frac{x^5}{5}+\frac{1\times3\times5}{2\times4\times6}\times\frac{x^7}{7}+o(x^7)$

$cosx=1-\frac{1}{2!}x^2+\frac{1}{4!}x^4+o(x^4)$

$\frac{1}{1-x}=1+x^2+x^3+o(x^3)$

$(1+x)^a=1+\frac{a}{1!}x+\frac{a(a-1)}{2!}x^2+\frac{a(a-1)(a-2)}{3!}x^3+o(x^3)$

一道例题：用红黄蓝三色给1*n的棋盘着色，要求红格数是偶数，且至少有一个蓝格的着色方法数

$h_1(x)=1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\ldots$

$h_2(x)=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots$

$h_3(x)=x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots$

$g^{(e)}(x)=h_1(x)h_2(x)h_3(x)$

$=(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\ldots)(1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots)(x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots)$

$=\frac{e^x+e^{-x}}{2}e^x(e^x-1)=\frac{e^{3x}-e^{2x}+e^x+1}{2}=-\frac{1}{2}+\sum_{n=0}^{\infty}(\frac{3^n-2^n+1}{2})\frac{x^n}{n!}$

$ans=h_n=\frac{3^n-2^n+1}{2}$

数论

区间素数筛

```
const int MAXN = 1e6+1e3;   //待筛的区间[L,R]长度
const int N = 50001;//保证大于(2^31-1)的算数平方根
bool prime[MAXN];
bool seive[N];
typedef long long ll;
int len;
void init()
```

```
{
        for(int i=2;i<N;i++) seive[i]=1;
        for(int i=2;i*i<N;i++)  //预处理
            if(seive[i])
                for(int j=2*i;j<N;j+=i)
                    seive[j]=false;
}
void seg_seive(ll L,ll R)   //区间筛法
{
    len=R-L+1;
    for(int i=0;i<len;i++) prime[i]=1;
    if(1-L>=0) prime[1-L]=0;   //易错因为1不是素数也不是合数，这也是区间筛的一个易错bug
    for(ll i=2; i*i<=R ;i++)
    {
        if(seive[i])
        {
            for(ll j=max((ll)2,(L-1+i)/i)*i;j<=R;j+=i)  //第二个易错点，j必须从大于1，因为L可能小于i，但是seive[i]是素数。
                prime[j-L]=false;
        }
    }
}
```

大质数判定与大数分解

```
typedef long long LL;
LL ans;
LL modmul(LL a,LL b,LL mod)
{
    LL ret=0;
    for(;b;b>>=1,a=(a+a)%mod)
        if(b&1)ret=(ret+a)%mod;
    return ret;
}
LL qpow(LL x,LL u,LL mod)
{
    LL ret=1LL;
    for(;u;u>>=1,x=modmul(x,x,mod))
        if(u&1)ret=modmul(ret,x,mod);
    return ret;
}
LL gcd(LL a,LL b)
{
    return b?gcd(b,a%b):a;
}
LL Pollard_Rho(LL n,LL c)
{
    LL i=1,j=2,x=rand()%(n-1)+1,y=x;
    while(1)
    {
        i++;
        x=(modmul(x,x,n)+c)%n;
```

```
        LL p=gcd((y-x+n)%n,n);
        if(p!=1&&p!=n)return p;
        if(y==x)return n;
        if(i==j)
        {
            y=x;
            j<<=1;
        }
    }
}
bool Miller_Rabin(LL n)//判断是否为质数
{
    LL x,pre,u=n-1;
    int i,j,k=0;
    if(n==2||n==3||n==5||n==7||n==11)return 1;
    if(n==1||!(n%2)||!(n%3)||!(n%5)||!(n%7)||!(n%11))return 0;
    while(!(u&1))
    {
        k++;
        u>>=1;
    }
    srand((long long)12234336);
    for(i=1;i<=50;i++)
    {
        x=rand()%(n-2)+2;
        if(!(n%x))return 0;
        x=qpow(x,u,n);
        pre=x;
        for(j=1;j<=k;j++)
        {
            x=modmul(x,x,n);
            if(x==1&&pre!=1&&pre!=n-1)return 0;
            pre=x;
        }
        if(x!=1)return 0;
    }
    return 1;
}
LL prime[1000];int tot=0;
void find(LL n,LL c)//大数分解
{
    if(n==1)return;
    if(Miller_Rabin(n))
    {
        prime[tot++]=n;
        return;
    }
    LL x=n,k=c;
    while(x==n)x=Pollard_Rho(x,c--);
    find(n/x,k);
    find(x,k);
}
```

威尔逊定理

当且仅当p为质数，p可整除(p-1)!+1 ， 即p|((p-1)!+1)

欧拉定理

若a,p为正整数且a,p互质， 则$a^{\varphi(p)} \equiv 1(modp)$

扩展欧拉定理

$gcd(a,p)=1$时， $a^b = a^{b\%\varphi(p)}(modp)$

$gcd(a,p) \neq 1$且 $b >= \varphi(p)$时， $a^b = a^{b\%\varphi(p)+\varphi(p)}(modp)$

费马小定理

若p是质数，则$a^{p-1} \equiv 1(modp)$

费马大定理

$a^n + b^n = c^n$ 当n>2时无整数解

中国剩余定理

```
//模数互质的情况，m[] 存模数，a[] 存模后的结果
ll m[N],a[N];
ll exgcd(ll a,ll b,ll &x,ll &y)
{
    if(b==0){x=1LL;y=0;return a;}
    ll r=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return r;
}
ll china(ll n,ll *m,ll *a)
{
    ll M=1,d,y,x=0;
    for(int i=0;i<n;i++) M*=m[i];
    for(int i=0;i<n;i++){
        ll w=M/m[i];
        exgcd(m[i],w,d,y);
        x=(x+y*w*a[i])%M;
    }
    return (x+M)%M;
}
```

```
//模数不互质
struct China
{
    #define ll long long
```

```
    ll gcd(ll a,ll b){return b==0?a:gcd(b,a%b);}
    ll exgcd(ll a,ll b,ll &x,ll &y)
    {
        if(b==0){x=1LL;y=0;return a;}
        ll r=exgcd(b,a%b,y,x);
        y-=a/b*x;
        return r;
    }
    //a在模n乘法下的逆元，没有则返回-1
    ll inv(ll a,ll n)
    {
        ll x,y;
        ll t=exgcd(a,n,x,y);
        if(t!=1)return -1;
        return(x%n+n)%n;
    }
    //合并两个方程
    bool merge(ll a1,ll n1,ll a2,ll n2,ll& a3,ll& n3)
    {
        ll d=gcd(n1,n2);
        ll c=a2-a1;
        if(c%d)return false;
        c=(c%n2+n2)%n2;
        c/=d;n1/=d;n2/=d;
        c*=inv(n1,n2);
        c%=n2;c*=n1*d;
        c+=a1;
        n3=n1*n2*d;
        a3=(c%n3+n3)%n3;
        return true;
    }
    //求模线性方程组x=ai(mod ni) 可以不互质
    ll china(int len,ll* a,ll* n)
    {
        ll a1=a[0],n1=n[0];
        ll a2,n2;
        for(int i=1;i<len;i++){
            ll aa,nn;
            a2=a[i],n2=n[i];
            if(!merge(a1,n1,a2,n2,aa,nn))return -1;
            a1=aa;n1=nn;
        }
        return (a1%n1+n1)%n1;
    }
}china;
```

扩展欧几里得

```
ll exgcd(ll a,ll b,ll &x,ll &y)
{
    if(b==0){x=1LL;y=0;return a;}
    ll r=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return r;
}
// by+(a-a/b*b)x = ax+b(y-a/b*x)
```

欧拉函数

```
//o(n)
void init_euler()
{
    memset(p1,0,sizeof(p1));
    memset(phi,0,sizeof(phi));
    memset(vis,0,sizeof(vis));
    for(int i=2;i<N;i++){
        if(!vis[i]){p1[++p1[0]]=i;phi[i]=i-1;}
        for(int j=1;j<=p1[0];j++){
            if(i*p1[j]>=N)break;
            vis[i*p1[j]]=true;
            if(!(i%p1[j])){phi[i*p1[j]]=phi[i]*p1[j];break;}
            phi[i*p1[j]]=phi[i]*(p1[j]-1);
        }
    }phi[1]=1;
    pre[0]=0;
    for(int i=1;i<N;i++)pre[i]=(pre[i-1]+phi[i])%p;
}
```

莫比乌斯函数

```
//o(n)
void init_mus()
{
    cnt=0;memset(vis,0,sizeof(vis));
    mus[1]=1;
    for(int i=2;i<N;i++){
        if(!vis[i]){prime[cnt++]=i;mus[i]=-1;}
        for(int j=0;j<cnt&&i*prime[j]<N;j++){
            vis[i*prime[j]]=1;
            if(i%prime[j]==0){mus[i*prime[j]]=0;break;}
            else mus[i*prime[j]]=-mus[i];
        }
    }
}
```

数论函数

积性函数：对于**任意互质的整数**a和b有性质$f(ab)=f(a)f(b)$的数论函数

完全积性函数：对任意整数a和b有性质$f(ab)=f(a)f(b)$的数论函数

约数函数：$\sigma_k=\sum_{d|n}d^k$

原函数：$e=[n=1]$

恒等函数：$I=1$

单位函数：$ID=n$

幂函数：$ID^k=n^k\ ID^0=I$

狄利克雷卷积及其他：

卷积：$(a\times b)(n)=\sum_{d|n}a(d)b(\frac{n}{d})$

乘：$(a\cdot b)=a(n)b(n)$

加：$(a+b)(n)=a(n)+b(n)$

交换律：$a\times b=b\times a$

结合律：$(a\times b)\times c=a\times(b\times c)$

分配率：$(a+b)\times c=a\times c+b\times c$

单位元：$a\times e=a$

注意：$a\cdot e=e$

单位元的逆元(莫比乌斯函数)：$\mu\times I=e$

莫比乌斯反演：$f=g\times I\Leftrightarrow g=f\times\mu$

欧拉函数的卷积：$\phi\times I=ID\Leftrightarrow\phi=ID\times\mu$

恒等函数的卷积：$I\times I=\sigma_0\Leftrightarrow I=\sigma_0\times\mu$

幂函数的卷积：$ID^k\times I=\sigma_k\Leftrightarrow ID^k=\sigma_k\times\mu$

若：$b(x)b(\frac{n}{x})=c(n)$

则：$(a\cdot b^k)\times b^k=\sum_{d|n}a(d)b(d)b(\frac{n}{d})=(a\times I)\cdot c^k(n)$

另外一种表述：g在完全积性条件下：

$(f\cdot g)\times g=(f\times I)\cdot g$

例子：由 $ID^k(d)ID^k(\frac{n}{d})=ID^k(n)$

可得：$(a\cdot ID^k)\times ID^k=(a\times I)\cdot ID^k$

$(\phi\cdot ID^k)\times ID^k=(\phi\times I)\cdot ID^k=ID\cdot ID^k=ID^{k+1}$

$(\mu\cdot ID^k)\times ID^k=(\mu\times I)\cdot ID^k=e\cdot ID^k=e$

$ID^k\times ID^k=(I\cdot ID^k)\times ID^k=(I\times I)\cdot ID^k=\sigma_0\cdot ID^k$

常用转换技巧

前n个正整数的约数之和：

$\sum_{i=1}^{n}\sigma_1(i)=\sum_{i=1}^{n}\sum_{j=1}^{n}[j|i]*j=\sum_{i=1}^{n}i*\sum_{j=1}^{n}[i|j]=\sum_{i=1}^{n}i*\lfloor\frac{n}{i}\rfloor=\sum_{i=1}^{n}\frac{\lfloor\frac{n}{i}\rfloor*(\lfloor\frac{n}{i}\rfloor+1)}{2}$

前n个正整数的欧拉函数之和：

$\varphi(n)=n-\sum_{d|n,d<n}\varphi(d)\ \phi(n)=\sum_{i=1}^{n}\varphi(i)$

$\phi(n)=\sum_{i=1}^{n}\varphi(i)=\sum_{i=1}^{n}(i-\sum_{d|i,d<i}\varphi(d))=\frac{n*(n+1)}{2}-\sum_{i=2}^{n}\sum_{d|i,d<i}\varphi(d)$

$=\frac{n*(n+1)}{2}-\sum_{i=2}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\varphi(d)=\frac{n*(n+1)}{2}-\sum_{i=2}^{n}\phi(\lfloor\frac{n}{i}\rfloor)$

$\frac{n*(n+1)}{2}=\sum_{i}^{n}i=\sum_{i=1}^{n}\sum_{d|i}\varphi(d)=\sum_{\frac{i}{d}=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{\frac{i}{d}}\rfloor}\varphi(d)=\sum_{i=1}^{n}\phi(\lfloor\frac{n}{i}\rfloor)$

前n个正整数的莫比乌斯函数之和（梅滕斯函数）：

$1=\sum_{i=1}^{n}[i=1]=\sum_{i=1}^{n}\sum_{d|i}\mu(d)=\sum_{i=1}^{n}\sum_{d=1}^{\lfloor\frac{n}{i}\rfloor}\mu(d)=\sum_{i=1}^{n}M(\lfloor\frac{n}{i}\rfloor)$

$M(n)=1-\sum_{i=2}^{n}M(\lfloor\frac{n}{i}\rfloor)$

$\mu^2(d)=\sum_{k^2|d}\mu(k)$：

证明：当d没有平方因子时 $\sum_{k^2|d}\mu(k)=1$，$\mu^2(d)=1$

当d有平方因子x^2时 $\sum_{k^2|d}\mu(k)=\sum_{k^2|x^2}\mu(k)=\sum_{k|x}\mu(k)=0,\mu^2(d)=0$

前n个正整数的莫比乌斯函数平方之和：

$S(n)=\sum_{i=1}^{n}\mu^2(i)=\sum_{d^2|i}\mu(d)=\sum_{i=1}^{n}\sum_{d=1}^{\sqrt{\lfloor\frac{n}{i}\rfloor}}\mu(d)$

前n个正整数的因数的莫比乌斯函数平方之和：

$S(n)=\sum_{i=1}^{n}\sum_{d|i}\mu^2(d)=\sum_{i=1}^{n}\sum_{d|i}\sum_{k^2|d}\mu(k)=\sum_{i=1}^{n}\mu(i)\sum_{j=1}^{\frac{n}{i^2}}\lfloor\frac{n}{i^2j}\rfloor$

可发现i的取值范围可以缩小到$\sqrt{n}$，令$f(n)=\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor$

则 $S(n)=\sum_{i=1}^{\sqrt{n}}\mu(i)f(\lfloor\frac{n}{i^2}\rfloor)$

小于等于n的两两数最小公倍数之和：

$ans=\sum_{d=1}^{n}d\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}i^2\varphi(i)$

第二个求和式卷积即可求

小于等于n的两两数最大公约数之和：

$ans=\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor*\lfloor\frac{n}{i}\rfloor*\varphi(i)$

求lcm(a,b)+lcm(a+1,b)+...+lcm(b,b):

$ans=\frac{b}{2}\sum_{T|b}(\frac{b}{T}+\lceil\frac{a}{T}\rceil)(\frac{b}{T}-\lceil\frac{a}{T}\rceil+1)\sum_{d|T}\mu(d)d$

最右边和式用积性函数性质算即可

求最小公倍数在某一区间的(x<=y)二元组数量：

区间[l,r]=[1,r]-[1,l-1]

$G(n)=\sum_{i=1}^{n}\sum_{y=1}^{n}[\frac{xy}{gcd(x,y)}<=n]=\sum_{k=1}^{\sqrt{n}}\mu(k)H(\lfloor\frac{n}{k^2}\rfloor)$

$H(n)=\sum_{a=1}^{n}\sum_{b=1}^{\lfloor\frac{n}{a}\rfloor}\sum_{c=1}^{\lfloor\frac{n}{ab}\rfloor}1$

H(n)枚举前两项(a<=b)用组合数求即可

FFT （快速傅里叶变换）

```
#define L(x) (1 << (x))
const double PI = acos(-1.0);
const int N = 2e5+5;
double ax[N],ay[N],bx[N],by[N];
int revv(int x, int bits)
{
    int ret=0;
    for(int i=0;i<bits;i++){ret<<=1;ret|=x&1;x>>=1;}
    return ret;
}
void fft(double * a, double * b, int n, bool rev)
{
    int bits=0;
    while((1<<bits)<n)++bits;
    for(int i=0;i<n;i++){
        int j=revv(i,bits);
        if(i<j)swap(a[i],a[j]),swap(b[i],b[j]);
    }
    for(int len=2;len<=n;len<<=1){
        int half=len>>1;
        double wmx=cos(2*PI/len),wmy=sin(2*PI/len);
        if(rev)wmy=-wmy;
        for(int i=0;i<n;i+=len){
            double wx=1,wy=0;
            for(int j=0;j<half;j++){
                double cx=a[i+j],cy=b[i+j];
                double dx=a[i+j+half],dy=b[i+j+half];
                double ex=dx*wx-dy*wy,ey=dx*wy+dy*wx;
                a[i+j]=cx+ex,b[i+j]=cy+ey;
                a[i+j+half]=cx-ex,b[i+j+half]=cy-ey;
                double wnx=wx*wmx-wy*wmy,wny=wx*wmy+wy*wmx;
                wx=wnx,wy=wny;
            }
        }
    }
    if(rev){for(int i=0;i<n;i++) a[i]/=n,b[i]/=n;}
}
int solve(int a[],int na,int b[],int nb,int ans[]) //两个数组求卷积,有时ans数组要开成long long
{
    int len=max(na,nb),ln;
    for(ln=0;L(ln)<len;++ln);
    len=L(++ln);
    for(int i=0;i<len;++i){
```

```
        if(i>=na)ax[i]=0,ay[i]=0;
        else ax[i]=a[i],ay[i]=0;
    }
    fft(ax,ay,len,0);
    for(int i=0;i<len;++i){
        if(i>=nb) bx[i]=0,by[i]=0;
        else bx[i]=b[i],by[i]=0;
    }
    fft(bx,by,len,0);
    for(int i=0;i<len;++i){
        double cx=ax[i]*bx[i]-ay[i]*by[i];
        double cy=ax[i]*by[i]+ay[i]*bx[i];
        ax[i]=cx,ay[i]=cy;
    }
    fft(ax,ay,len,1);
    for(int i=0;i<len;++i) ans[i]=(int)(ax[i]+0.5);
    return len;
}
```

NTT （快速数论变换）

```
//选好素数
const int M = 22;
ll G=3,P=998244353,wn[M];//P=7*2^26+1, M=k-1
ll modexp(ll a,ll b,ll p){
    ll ans=1LL;a%=p;
    while(b){if(b&1)ans=ans*a%p;b>>=1;a=a*a%p;}
    return ans;
}
void getwn(){for(int i=0;i<M;i++) wn[i]=modexp(G,(P-1)/(1<<i),P);}//!!!记得用
void NTT(ll x[],int n,int rev) {
    int i,j,k,ds;ll w,u,v;
    for(i=1,j=n>>1,k=n>>1;i<n-1;i++,k=n>>1){
        if(i<j)swap(x[i],x[j]);
        while(j>=k){j-=k;k>>=1;}
        if(j<k)j+=k;
    }
    for(i=2,ds=1;i<=n;i<<=1,ds++)
        for(j=0;j<n;j+=i){
            w=1;
            for(k=j;k<j+i/2;k++){
                u=x[k]%P;v=w*x[k+i/2]%P;
                x[k]=(u+v)%P;
                x[k+i/2]=(u-v+P)%P;
                w=w*wn[ds]%P;
            }
        }
    if(rev==-1){
        for(i=1;i<n/2;i++) swap(x[i],x[n-i]);
        w=modexp(n,P-2,P);
        for(i=0;i<n;i++) x[i]=x[i]*w%P;
    }
```

```
}
int solve(ll a[],int lena,ll b[],int lenb) {//结果在a数组
    int n=1;
    while (n<lena+lenb) n<<=1;
    for(int i=lena;i<n;i++)a[i]=0;
    for(int i=lenb;i<n;i++)b[i]=0;
    NTT(a,n,1);NTT(b,n,1);
    for (int i=0;i<n;i++) a[i]=a[i]*b[i]%P;
    NTT(a,n,-1);
    return n;
}
```

三模数NTT所需函数

```
inline ll mul(ll a,ll b,ll mod){
    a%=mod;b%=mod;
    return ((a*b-(ll)((ll)((long double)a/mod*b+1e-3)*mod))%mod+mod)%mod;
}
ll modexp(ll a,ll b,ll p)
{
    ll ans=1LL;a%=p;
    while(b){if(b&1)ans=ans*a%p;b>>=1;a=a*a%p;}
    return ans;
}
ll M[]={998244353,1004535809,469762049};
ll _M=(ll)M[0]*M[1];
ll m1=M[0],m2=M[1],m3=M[2];
ll inv1=modexp(m1%m2,m2-2,m2),inv2=modexp(m2%m1,m1-2,m1),inv12=modexp(_M%m3,m3-2,m3);
inline int CRT(int a1,int a2,int a3){//注意顺序
    ll A=(mul((ll)a1*m2%_M,inv2,_M)+mul((ll)a2*m1%_M,inv1,_M))%_M;
    ll k=((ll)a3+m3-A%m3)*inv12%m3;
    return (k*(_M%P)+A)%P;
}
```

NTT 素数表

如果模数P 任意，取的模数必须超过n(P-1)^2 ， 可以取多个模数（乘积>n(P-1)^2）做NTT。再用中国剩余定理合并,每次NTT转化为三个NTT再合并即可

```

r·2^k+1 r   k   g
3   1   1   2
5   1   2   2
17  1   4   3
97  3   5   5
193 3   6   5
257 1   8   3
7681    15  9   17
12289   3   12  11
```

11	40961	5	13	3	
12	65537	1	16	3	
13	786433	3	18	10	
14	5767169	11	19	3	
15	7340033	7	20	3	
16	23068673	11	21	3	
17	104857601	25	22	3	
18	167772161	5	25	3	
19	469762049	7	26	3	
20	998244353	119	23	3	
21	1004535809	479	21	3	
22	2013265921	15	27	31	
23	2281701377	17	27	3	
24	3221225473	3	30	5	
25	75161927681	35	31	3	
26	77309411329	9	33	7	
27	206158430209	3	36	22	
28	2061584302081	15	37	7	
29	2748779069441	5	39	3	
30	6597069766657	3	41	5	
31	39582418599937	9	42	5	
32	79164837199873	9	43	5	
33	263882790666241	15	44	7	
34	1231453023109121	35	45	3	
35	1337006139375617	19	46	3	
36	3799912185593857	27	47	5	
37	4222124650659841	15	48	19	
38	7881299347898369	7	50	6	
39	31525197391593473	7	52	3	
40	180143985094819841	5	55	6	
41	1945555039024054273	27	56	5	
42	4179340454199820289	29	57	3	

多项式开方、求逆、积分、求exp、求ln、求幂

多项式求逆： $B(x) = 2G(x) - A(x)G^2(x)(modx^n)$

多项式开方： $B(x) = \frac{A(x)+G^2(x)}{2G(x)}(modx^n)$

$G(x) = (1 + ln(1 + \frac{1}{exp(\int \frac{1}{\sqrt{F(x)}})}))^k$

```
1  //给定F(x) 求G'(x)
2  # include <bits/stdc++.h>
3  # define RG register
4  # define IL inline
5  # define Fill(a, b) memset(a, b, sizeof(a))
6  using namespace std;
7  typedef long long ll;
8
9  IL int Input(){
10     RG int x = 0, z = 1; RG char c = getchar();
11     for(; c < '0' || c > '9'; c = getchar()) z = c == '-' ? -1 : 1;
```

```
    for(; c >= '0' && c <= '9'; c = getchar()) x = (x << 1) + (x << 3) + (c ^ 48);
    return x * z;
}

const int maxn(8e5 + 5);
const int mod(998244353);
const int phi(998244352);
const int inv2(499122177);
const int g(3);

int a[maxn], b[maxn], c[maxn], d[maxn], r[maxn];

IL int Pow(RG ll x, RG ll y){
    RG ll ret = 1;x%=mod;
    for(; y; y >>= 1, x = x * x % mod)
        if(y & 1) ret = ret * x % mod;
    return ret;
}
IL void NTT(RG int *p, RG int m, RG int opt){
    RG int n = 1, l = 0;
    for(; n < m; n <<= 1) ++l;
    for(RG int i = 0; i < n; ++i) r[i] = (r[i >> 1] >> 1) | ((i & 1) << (l - 1));
    for(RG int i = 0; i < n; ++i) if(r[i] < i) swap(p[i], p[r[i]]);
    for(RG int i = 1; i < n; i <<= 1){
        RG int t = i << 1, wn = Pow(g, phi / t);
        if(opt == -1) wn = Pow(wn, mod - 2);
        for(RG int j = 0; j < n; j += t)
            for(RG int k = 0, w = 1; k < i; ++k, w = 1LL * w * wn % mod){
                RG int x = p[k + j], y = 1LL * w * p[k + j + i] % mod;
                p[k + j] = (x + y) % mod, p[k + j + i] = (x - y + mod) % mod;
            }
    }
    if(opt == -1){
        RG int inv = Pow(n, mod - 2);
        for(RG int i = 0; i < n; ++i) p[i] = 1LL * p[i] * inv % mod;
    }
}
IL void Inv(RG int *p, RG int *q, RG int len){//多项式求逆,p为A(x), q为G(x)
    if(len == 1){
        q[0] = Pow(p[0], mod - 2);
        return;
    }
    Inv(p, q, len >> 1);
    for(RG int i = 0; i < len; ++i) a[i] = p[i], b[i] = q[i];
    RG int tmp = len << 1;
    NTT(a, tmp, 1), NTT(b, tmp, 1);
    for(RG int i = 0; i < tmp; ++i) a[i] = 1LL * a[i] * b[i] % mod * b[i] % mod;
    NTT(a, tmp, -1);
    for(RG int i = 0; i < len; ++i) q[i] = ((2 * q[i] - a[i]) % mod + mod) % mod;
    for(RG int i = 0; i < tmp; ++i) a[i] = b[i] = 0;
}
IL void Sqrt(RG int *p, RG int *q, RG int len){//多项式开方,p为A(x),q为G(x)
    if(len == 1){
```

```
65          q[0] = sqrt(p[0]); //???
66          return;
67      }
68      Sqrt(p, q, len >> 1), Inv(q, c, len);
69      RG int tmp = len << 1;
70      for(RG int i = 0; i < len; ++i) a[i] = p[i];
71      NTT(a, tmp, 1), NTT(c, tmp, 1);
72      for(RG int i = 0; i < tmp; ++i) a[i] = 1LL * a[i] * c[i] % mod;
73      NTT(a, tmp, -1);
74      for(RG int i = 0; i < len; ++i) q[i] = 1LL * (q[i] + a[i]) % mod * inv2 % mod;
75      for(RG int i = 0; i < tmp; ++i) a[i] = c[i] = 0;
76  }
77  IL void ICalc(RG int *p, RG int *q, RG int len){//多项式求导
78      q[len - 1] = 0;
79      for(RG int i = 1; i < len; ++i) q[i - 1] = 1LL * p[i] * i % mod;
80  }
81  IL void Calc(RG int *p, RG int *q, RG int len){//多项式积分
82      q[0] = 0;
83      for(RG int i = 1; i < len; ++i) q[i] = 1LL * Pow(i, mod - 2) * p[i - 1] % mod;
84  }
85  IL void Ln(RG int *p, RG int *q, RG int len){//多项式求ln
86      Inv(p, c, len), ICalc(p, a, len);
87      RG int tmp = len << 1;
88      NTT(c, tmp, 1), NTT(a, tmp, 1);
89      for(RG int i = 0; i < tmp; ++i) c[i] = 1LL * c[i] * a[i] % mod;
90      NTT(c, tmp, -1), Calc(c, q, len);
91      for(RG int i = 0; i < tmp; ++i) a[i] = c[i] = 0;
92  }
93  IL void Exp(RG int *p, RG int *q, RG int len){//多项式求exp
94      if(len == 1){
95          q[0] = 1;
96          return;
97      }
98      Exp(p, q, len >> 1), Ln(q, b, len);
99      for(RG int i = 0; i < len; ++i) b[i] = (mod - b[i] + p[i]) % mod, c[i] = q[i];
100     b[0] = (b[0] + 1) % mod;
101     RG int tmp = len << 1;
102     NTT(b, tmp, 1), NTT(c, tmp, 1);
103     for(RG int i = 0; i < tmp; ++i) b[i] = 1LL * b[i] * c[i] % mod;
104     NTT(b, tmp, -1);
105     for(RG int i = 0; i < len; ++i) q[i] = b[i];
106     for(RG int i = 0; i < tmp; ++i) b[i] = c[i] = 0;
107 }
108 IL void CalcPow(RG int *p, RG int *q, RG int len, RG int y){//多项式求幂
109     Ln(p, d, len);
110     for(RG int i = 0; i < len; ++i) d[i] = 1LL * d[i] * y % mod;
111     Exp(d, q, len);
112     for(RG int i = 0; i < len; ++i) d[i] = 0;
113 }
114 int f[maxn], h[maxn], n, k, len;
115
116 IL void Record(){
117     for(RG int i = 0; i < len; ++i) f[i] = h[i], h[i] = 0;
```

```
}
int main(){
    freopen("polynomial.in", "r", stdin);
    freopen("polynomial.out", "w", stdout);
    n = Input() - 1, k = Input();
    for(RG int i = 0; i <= n; ++i) f[i] = Input();
    for(len = 1; len <= n; len <<= 1);
    Sqrt(f, h, len), Record();
    Inv(f, h, len), Record();
    Calc(f, h, n + 1), Record();
    Exp(f, h, len), Record();
    Inv(f, h, len), Record();
    f[0] = (f[0] + 1) % mod;
    Ln(f, h, len), Record();
    f[0] = (f[0] + 1) % mod;
    CalcPow(f, h, len, k), Record();
    ICalc(f, h, n + 1);
    for(RG int i = 0; i <= n; ++i) printf("%d ", h[i]);
    return 0;
}
```

多项式除法及取模（待补）

BSGS算法（求最小的x满足 a^x = b (mod n) n为质数）

```
const int MOD = 76543;
int hs[MOD],head[MOD],Next[MOD],id[MOD],top;
void insert(int x,int y)
{
    int k = x%MOD;
    hs[top] = x, id[top] = y, Next[top] = head[k], head[k] = top++;
}
int find(int x)
{
    int k = x%MOD;
    for(int i = head[k]; i != -1; i = Next[i])
        if(hs[i] == x)
            return id[i];
    return -1;
}
int BSGS(int a,int b,int n)//a^x = b(mod n)
{
    a%=n;b%=n;
    memset(head,-1,sizeof(head));
    top = 1;
    if(b == 1)return 0;
    int m = sqrt(n*1.0), j;
    long long x = 1, p = 1;
    for(int i = 0; i < m; ++i, p = p*a%n)insert(p*b%n,i);
    for(long long i = m; ;i += m)
```

```
    {
        if( (j = find(x = x*p%n)) != -1 )return i-j;
        if(i > n)break;
    }
    return -1;
}
```

ex_BSGS算法 （n不为质数）

```
const int N = 1e5+3;//一定为素数
ll point[N*2],Next[N*2],val[N*2],pla[N*2],tot;
void add(ll x,ll y){
    ll xx=x%N;
    Next[++tot]=point[xx];
    val[tot]=x;
    pla[tot]=y;
    point[xx]=tot;
}
ll find(ll x){
    ll i;
    ll xx=x%N;
    for (i=point[xx];i;i=Next[i]){
        if (val[i]==x)return pla[i];
    }
    return -1;
}
ll gcd(ll a,ll b){
    return b==0?a:gcd(b,a%b);
}
ll qpow(ll a,ll b,ll mod){
    ll ans;
    for (ans=1;b;a=a*a%mod,b>>=1){
        if (b&1)ans=ans*a%mod;
    }
    return ans;
}
ll ex_BSGS(ll a,ll b,ll p){// a^x = b (mod p)
    a%=p;b%=p;
    ll i;
    if (b==1)return 0;
    ll t=gcd(a,p),d=1,k=0;
    while (t!=1){
        if (b%t)return -1;
        b/=t;
        p/=t;
        d=d*(a/t)%p;
        k++;
        if (b==d)return k;
        t=gcd(a,p);
    }
    memset (point,0,sizeof(point));
    memset (Next,0,sizeof(Next));
```

```
    tot=0;
    ll m=ceil(sqrt(p));
    ll am=qpow(a,m,p);
    ll mul=b;
    add(mul,0);
    for (i=1;i<=m;i++){
        mul=mul*a%p;
        add(mul,i);
    }
    for (i=1;i<=m+1;i++){
        d=d*am%p;
        ll place=find(d);
        if (place!=-1)return i*m-place+k;
    }
    return -1;
}
```

博弈论

打表

```
//f[]: 可以取走的石子个数
//sg[]:0~n的SG函数值
//hash[]:mex{}
int f[N],sg[N],hash[N];
void getSG(int n)
{
    int i,j;
    memset(sg,0,sizeof(sg));
    for(i=1;i<=n;i++)
    {
        memset(hash,0,sizeof(hash));
        for(j=1;f[j]<=i;j++)
            hash[sg[i-f[j]]]=1;
        for(j=0;j<=n;j++)    //求mes{}中未出现的最小的非负整数
        {
            if(hash[j]==0)
            {
                sg[i]=j;
                break;
            }
        }
    }
}

```

DFS

```
//注意 S数组要按从小到大排序 SG函数要初始化为-1 对于每个集合只需初始化1遍
//n是集合s的大小 S[i]是定义的特殊取法规则的数组
int s[110],sg[10010],n;
int SG_dfs(int x)
{
    int i;
    if(sg[x]!=-1)
        return sg[x];
    bool vis[110];
    memset(vis,0,sizeof(vis));
    for(i=0;i<n;i++)
    {
        if(x>=s[i])
        {
            SG_dfs(x-s[i]);
            vis[sg[x-s[i]]]=1;
        }
    }
    int e;
    for(i=0;;i++)
        if(!vis[i])
        {
            e=i;
            break;
        }
    return sg[x]=e;
}

```

巴什博奕

只有一堆n个物品，两个人轮流从中取物，规定每次最少取一个，最多取m个，最后取光者为胜

答案：先手必败 n%(m+1)=0

威佐夫博弈

有两堆各若干的物品，两人轮流从其中一堆取至少一件物品，至多不限，或从两堆中同时取相同件物品，规定最后取完者胜利。

答案：若两堆物品的初始值为（x，y），且x<y，则另z=y-x；记w=（int）[（（sqrt（5）+1）/2）*z]；若w=x，则先手必败，否则先手必胜。

斐波那契博弈

有一堆物品，两人轮流取物品，先手最少取一个，至多无上限，但不能把物品取完，之后每次取的物品数不能超过上一次取的物品数的二倍且至少为一件，取走最后一件物品的人获胜。

答案：先手胜当且仅当n不是斐波那契数

取走-分割游戏

Lasker's Nim游戏：每一轮允许两种操作之一。（1）从一堆石子中取走任意多个（2）将一堆数量不少于2的石子分成都不为空的两堆。

答案：对于所有的k>=0，有g(4k+1)=4k+1；g(4k+2)=4k+2；g(4k+3)=4k+4；g(4k+4)=4k+3。

POJ 1740

有N堆石子，两人轮流进行操作，每一次为"操作者指定一堆石子，先从中扔掉一部分（至少一颗，可以全部扔掉），然后可以将该堆剩下的石子中的任意多颗任意移到其他未取完的堆中"，操作者无法完成操作时为负。

答案：必败态的条件为"堆数为偶数（不妨设为2N），并且可以分为两两相等的N对"

阶梯博弈

从左到右有一排石子，给出石子所在的位置。规定每个石子只能向左移动，且不能跨过前面的石子。最左边的石子最多只能移动到1位置。每次选择一个石子按规则向左移动，问先手是否能赢。

答案：我们把棋子按位置升序排列后，从后往前把他们两两绑定成一对。如果总个数是奇数，就把最前面一个和边界（位置为0）绑定。 在同一对棋子中，如果对手移动前一个，你总能对后一个移动相同的步数，所以一对棋子的前一个和前一对棋子的后一个之间有多少个空位置对最终的结果是没有影响的。 于是我们只需要考虑同一对的两个棋子之间有多少空位。 这样一来就成了N堆取石子游戏了.

翻硬币游戏

N 枚硬币排成一排。有的正面朝上。有的反面朝上。我们从左開始对硬币按1 到N 编号。 游戏者依据某些约束翻硬币，但他所翻动的硬币中，最右边那个硬币的必须是从正面翻到反面。 谁不能翻谁输。

局面的 SG 值为局面中每个正面朝上的棋子单一存在时的 SG 值的异或和。

SG(HHTHTTHT)=SG(H)$^{SG(TH)}$SG(TTTH)^SG(TTTTTTH)

约束条件1：每次仅仅能翻一个硬币。

有奇数个正面硬币。局面的SG值==1，先手必胜，有偶数个正面硬币，局面的SG值==0。先手必败。

约束条件2：每次能翻转一个或两个硬币。 不用连续

每一个硬币的SG值为它的编号。初始编号为0。与NIM游戏是一样的。 (只有一个正面硬币且在0位置特判)

约束条件3：每次必须连续翻转k个硬币。

我们计算的是个数为N的硬币中，当中最后一个硬币为正面朝上,的sg值。

sg的形式为000...01 000...01，当中一小段0的个数为k-1。

约束条件4：每次翻动一个硬币后。必须翻动其左侧近期三个硬币中的一个，即翻动第x个硬币后。必须选择x-1。x-2，x-3中的当中一个硬币进行翻动，除非x是小于等于3的。（Subtraction Games）

考虑单一硬币，这个与每次最多仅仅能取3个石子的取石子游戏的SG分布一样，

1，2，3，4，5（1，2，3，0，1）

约束条件5：每次必须翻动两个硬币，并且这两个硬币的距离要在可行集S={1,2,3}中。硬币序号从0開始。(Twins游戏)

与约束条件4一样，0，1，2，3，4，5，（0，1，2，3，0，1）

约束条件6：每次能够翻动一个、二个或三个硬币。（Mock Turtles游戏）

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 （1 2 4 7 8 11 13 14 16 19 21 22 25 26 28 ）

sg值为2x或者2x+1。我们称一个非负整数为odious，当且仅当该数的二进制形式的1出现的次数是奇数，否则称作evil。所以1，2。4，7是odious由于它们的二进制形式是1,10,100,111.而0,3,5,6是evil，由于它们的二进制形式是0,11,101,110。上面 sg值都是odious数。所以当2x为odious时，sg值是2x，当2x是evil时。sg值是2x+1.

约束条件7：每次能够连续翻动随意个硬币，至少翻一个。（Ruler游戏）

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 （1 2 1 4 1 2 1 8 1 2 1 4 1 2 1 16 ）

sg值为x的因数其中2的能达到的最大次幂。比方14=2*7，最大1次幂。即2；16=2222*。最大4次幂，即16。

约束条件8：每次必须翻转4个对称的硬币，最左与最右的硬币都必须是从正翻到反。（開始的时候两端都是正面）（Grunt游戏）

这是Grundy游戏的变种，初始编号从0開始。 当首正硬币位置为0,1,2时是terminal局面，即 终结局面，sg值都是0。当首正硬币位置n大于等于3的时候的局面能够通过翻0,x,n-x,n四个位置得到(当中x<n/2可保证胜利)。

这就像是把一堆石子分成两堆不同大小石子的游戏，也就是Grundy游戏。

Grundy游戏

一开始有大小为n的一个堆，每次可以将大小大于2的堆拆成两个大小不同的堆,

可以使用Sprague-Grundy定理分析游戏。这需要将游戏中的堆大小映射到等效的nim堆大小。

Heap Size: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Equivalent NIM heap: 0 0 0 1 0 2 1 0 2 1 0 2 1 3 2 1 3 2 4 3 0

目前没有一个公式可以计算sg值

```
#include <algorithm>
#include <array>
#include <iostream>
int main() {
    constexpr int bound = 10000;
    std::array<int, bound+1> gnumbers;
    std::array<bool, bound/2+1> excluded;
    for (int i = 0; i <= bound; ++i) {
        auto e_begin = excluded.begin();
        auto e_end = e_begin + i/2;
        std::fill(e_begin, e_end, false);
        for (int j = 1; j < (i+1)/2; ++j) {
            int const k = i - j;
            excluded[gnumbers[j] ^ gnumbers[k]] = true;
        }
        gnumbers[i] = std::find(e_begin, e_end, false) - e_begin;
    }
    for (int i = 0; i <= bound; ++i)
        std::cout << i << ' ' << gnumbers[i] << '\n';
}
```

计算几何

```
#include <bits/stdc++.h>
using namespace std;
const double eps=1e-8;
#define ll long long
#define sz(x) (int)(x).size()
const double PI=acos(-1.0);
ll gcd(ll x,ll y){return y==0?x:gcd(y,x%y);}
int sgn(double x){if(abs(x)<eps)return 0;return x<0?-1:1;}
struct point{
    double x,y;

    point(double _x=0,double _y=0){
        x=_x;
        y=_y;
    }
    double norm(){return sqrt(x*x+y*y);}
    point operator - (const point & b)const {return {x-b.x,y-b.y};}
    point operator + (const point & b)const {return {x+b.x,y+b.y};}
    point operator / (const double & b)const {return {x/b,y/b};}
    point operator * (const double & b)const {return {x*b,y*b};}
    double operator * (const point & b)const {return x*b.y-b.x*y;}//叉乘
    bool operator < (const point & b)const {return sgn(x-b.x)?x<b.x:sgn(y-b.y)?y<b.y:0;}
    bool operator == (const point & b)const {return !sgn(x-b.x) && !sgn(y-b.y);}
};
struct line{
    point a,b;
    double angle;
    line(point _a=0,point _b=0){
        a=_a;b=_b;
        angle=atan2(b.y-a.y,b.x-a.x);
    }
};
double dot( point a , point b ) { return a.x * b.x + a.y * b.y ; }
double det( point a , point b ) { return a.x * b.y - a.y * b.x ; }
double dist( point a , point b ) { return ( a - b ).norm() ; }
double area(const point& a,const point& b,const point& c){return abs((b-a)*(c-a))/2;}
pair<bool,point> line_intersection(const line & a,const line & b){//线段交
    if(!sgn((a.a-a.b)*(b.a-b.b)))return {0,{}};
    if(sgn((b.a-a.a)*(b.a-b.b))*sgn((b.a-a.b)*(b.a-b.b))>0)return {0,{}};
    if(sgn((a.a-b.a)*(a.a-a.b))*sgn((a.a-b.b)*(a.a-a.b))>0)return {0,{}};
    double sa=area(a.a,b.a,a.b);
    double sb=area(a.b,a.a,b.b);
    return {1,b.a+(b.b-b.a)*sa/(sa+sb)};
}
vector<point> graham(vector<point> p){
    sort(p.begin(),p.end(),[](const point & a,const point & b){return a.x==b.x?
a.y<b.y:a.x<b.x;});
    p.erase(unique(p.begin(),p.end()),p.end());
```

```
    int n=sz(p),m=0,k,i;
    vector<point> res(n+1);
    for(i=0;i<n;i++){
        while(m>1&&(res[m-1]-res[m-2])*(p[i]-res[m-2])<=0) m--;
        res[m++]=p[i];
    }
    k=m;
    for(i=n-2;i>=0;i--){
        while(m>k&&(res[m-1]-res[m-2])*(p[i]-res[m-2])<=0) m--;
        res[m++]=p[i];
    }
    if(n>1) m--;
    res.resize(m);
    return res;
}

int main(){

    return 0;
}

```

多边形面积

```
double getArea(point *p,int n)//多边形面积
{
    if (n< 3) return 0;
    double area = 0;
    for (int i = 1; i < n-1; i++)
        area += (p[i]-p[0])*(p[i+1]-p[0]);//利用p数组求面积
    return fabs(area/2.0);
}
```

半平面交

```
point getIntersect(line& l1, line& l2){//求直线交点
    double A1 = l1.b.y - l1.a.y;
    double B1 = l1.a.x - l1.b.x;
    double C1 = (l1.b.x - l1.a.x) * l1.a.y - (l1.b.y - l1.a.y) * l1.a.x;
    double A2 = l2.b.y - l2.a.y;
    double B2 = l2.a.x - l2.b.x;
    double C2 = (l2.b.x - l2.a.x) * l2.a.y - (l2.b.y - l2.a.y) * l2.a.x;
    point p;
    p.x = (C2 * B1 - C1 * B2) / (A1 * B2 - A2 * B1);
    p.y = (C1 * A2 - C2 * A1) / (A1 * B2 - A2 * B1);
    return p;
}
bool cmp(const line& l1, const line& l2)
{
    int d = sgn(l1.angle-l2.angle);

```

```
    if (!d) return sgn((l2.a-l1.a)*(l2.b-l1.a)) < 0;
    return d < 0;
}
bool judge(line& l0, line& l1, line& l2)
{
    point p=getIntersect(l1,l2);
    return sgn((l0.a-p)*(l0.b-p)) > 0;
    //与上面的注释处的大于小于符号相反，大于0，是p在向量l0.a->l0.b的左边，小于0是在右边，当p不在半平面l0内时，返回true
}
void HalfPlaneIntersect(point *p,int n,point *p1,int& n1)//半平面交,p1放答案
{
    p[n]=p[0];
    vector<line>l;
    for(int i=0;i<n;i++) l.push_back(line(p[i],p[i+1]));
    sort(l.begin(),l.end(), cmp); //极角排序
    int i,j;
    for (i = 0, j = 0; i < n; i++)
        if (sgn(l[i].angle-l[j].angle) > 0)
            l[++j] = l[i];//排除极角相同（从了l[1]开始比较)
    n = j + 1;//个数
    vector<int>dq(n*2);
    dq[0] = 0;//双端队列
    dq[1] = 1;//开始入队列两条直线
    int top=1,bot=0;

    for (int i = 2; i < n; i++)
    {
        while (top > bot && judge(l[i], l[dq[top]], l[dq[top-1]])) top--;
        while (top > bot && judge(l[i], l[dq[bot]], l[dq[bot+1]])) bot++;
        dq[++top] = i;
    }
    while (top > bot && judge(l[dq[bot]], l[dq[top]], l[dq[top-1]])) top--;
    while (top > bot && judge(l[dq[top]], l[dq[bot]], l[dq[bot+1]])) bot++;
    dq[++top] = dq[bot];n1=0;
    for (int i = bot; i < top; i++){
        p1[n1++]=getIntersect(l[dq[i+1]], l[dq[i]]);//更新重复利用p数组
    }
}
```

点到线段，线段到线段

```
double PointTOline( point const&a,point const&b,point const&p){
    double ap_ab = (b.x - a.x)*(p.x - a.x)+(b.y - a.y)*(p.y - a.y);//cross( a , p , b );
    if ( ap_ab <= 0 )
        return sqrt( (p.x-a.x)*(p.x-a.x) + (p.y-a.y)*(p.y-a.y) );

    double d2 = ( b.x - a.x ) * ( b.x - a.x ) + ( b.y-a.y ) * ( b.y-a.y ) ;
    if ( ap_ab >= d2 ) return sqrt( (p.x - b.x )*( p.x - b.x ) + ( p.y - b.y )*( p.y - b.y));
```

```
    double r = ap_ab / d2;
    double px = a.x + ( b.x - a.x ) *r;
    double py = a.y + ( b.y - a.y ) *r;
    return sqrt( (p.x - px)*(p.x - px) + (p.y - py)*(p.y - py) );
}
double LineToline(line const& A,line const& B)
{
    if(line_intersection(A,B).fi)return 0.0;
    double ans1=min(PointTOline(A.a,A.b,B.a),PointTOline(A.a,A.b,B.b));
    double ans2=min(PointTOline(B.a,B.b,A.a),PointTOline(B.a,B.b,A.b));
    return min(ans1,ans2);
}
```

点是否在简单多边形内

```
bool isOnline( point const&a,point const&b, point const&po ){//点是否在线段上
    return po.x >= min( a.x , b.x ) &&
           po.x <= max( a.x , b.x ) &&
           po.y >= min( a.y , b.y ) &&
           po.y <= max( a.y , b.y ) &&
           ( po.x - a.x ) * ( b.y - a.y ) == ( po.y - a.y ) * ( b.x - a.x );
}
bool isInSimple( point * p ,int n , point const&po ){//点是否在简单多边形内

    p[n] = p[0];
    bool flag = 0;
    int tmp;
    for ( int i = 0; i < n;++i ){
        if ( isOnline( p[i] , p[i+1] , po ) ) return true;
        if ( p[i].y == p[i+1].y ) continue;
        p[i].y < p[i+1].y ? tmp = i+1 : tmp = i ;
        if ( po.y == p[tmp].y && po.x < p[tmp].x ) flag ^= 1;
        p[i].y > p[i+1].y ? tmp = i+1 : tmp = i ;
        if ( po.y == p[tmp].y && po.x < p[tmp].x ) continue ;

        if ( po.x < max( p[i].x , p[i+1].x ) &&
             po.y > min( p[i].y , p[i+1].y ) &&
             po.y < max( p[i].y , p[i+1].y ) ) flag ^= 1;
    }
    return flag;
}
```

点到凸多边形，凸多边形到凸多边形

```
double pointTopolygon(point * p,int n,point const& po)
{
    double ans=1e18;
    if(isInSimple(p,n,po))return 0.0;
    p[n]=p[0];
    for(int i=1;i<=n;i++) ans=min(ans,PointTOline(p[i-1],p[i],po));

```

```
    return ans;
}
double polygonTopolygon(point * p1,int n1,point * p2,int n2)
{
    double ans=1e18;
    for(int i=0;i<n1;i++)ans=min(ans,pointTopolygon(p2,n2,p1[i]));
    for(int i=0;i<n2;i++)ans=min(ans,pointTopolygon(p1,n1,p2[i]));
    return ans;
}
```

点到圆圆到圆

```
double pointTocircle(pair<point,double> &c,point &po)
{
    return max(0.0,dist(po,c.fi)-c.se);
}
double CircleTocircle(pair<point,double> a,pair<point,double> b)
{
    return max(0.0,dist(a.fi,b.fi)-a.se-b.se);
}
```

线段-圆-凸多边形

```
double lineTocircle(line const & l,pair<point,double> const & c)
{
    return max(0.0,PointTOline(l.a,l.b,c.fi)-c.se);
}
double lineTopolygon(line const & l,point *p,int n)
{
    int flag=0;
    if(isInSimple(p,n,l.a)||isInSimple(p,n,l.b))flag=1;
    p[n]=p[0];
    for(int i=1;i<=n;i++) if(line_intersection(l,{p[i],p[i-1]}).fi)flag=1;
    if(flag)return 0.0;
    double ans=INF;
    for(int i=1;i<=n;i++) ans=min(ans,PointTOline(p[i-1],p[i],l.a));
    for(int i=1;i<=n;i++) ans=min(ans,PointTOline(p[i-1],p[i],l.b));
    for(int i=0;i<n;i++) ans=min(ans,PointTOline(l.a,l.b,p[i]));
    return ans;
}
double CircleTopolygon(pair<point,double> const & c,point *p,int n)
{
    return max(0.0,pointTopolygon(p,n,c.fi)-c.se);
}
```

费马点

```cpp
//三角形内一点到三点距离之和最短
#include <bits/stdc++.h>

using namespace std;
#define pb push_back
#define fi first
#define se second

struct Vec
{
    double x,y;
    Vec(double xx=0,double yy=0)
    {
        x=xx;
        y=yy;
    }
};

struct Point
{
    double x,y;
    Point(double xx=0,double yy=0)
    {
        x=xx;
        y=yy;
    }
};

double ddot(Vec A,Vec B)
{
    return A.x*B.x+A.y*B.y;
}
double getlen(Vec A)
{
    return sqrt(A.x*A.x+A.y*A.y);
}

double getlen(Point A,Point B)
{
    return sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y));
}

bool ask(double xa,double ya,double xb,double yb,double xc,double yc)//判断费马点是否在顶点上
{
    Vec ab(xb-xa,yb-ya),ac(xc-xa,yc-ya);
    if (ddot(ab,ac)/getlen(ab)/getlen(ac) < -0.5)
    {
        return true;
    }
    return false;
}

inline void swap(double &a,double &b)
```

```
54  {
55      double t;
56      t=a;
57      a=b;
58      b=t;
59  }
60
61
62  Point getAnotherPoint(Point A,Point B,Point C)
63  {
64      Point r,C1,C2;
65      Vec AB(B.x-A.x,B.y-A.y);
66      double len,len2;
67      double sqrt3=sqrt(3.0);
68      Vec AB2,crossAB,crossAB2;
69
70      AB2.x=AB.x/2; AB2.y=AB.y/2;
71      crossAB.x=AB2.y; crossAB.y=-AB2.x;
72      crossAB2.x=-AB2.y; crossAB2.y=AB2.x;
73
74      len=getlen(AB2);
75
76      crossAB.x*=sqrt3; crossAB.y*=sqrt3;
77      crossAB2.x*=sqrt3; crossAB2.y*=sqrt3;
78
79      C1.x=A.x+AB2.x+crossAB.x;C1.y=A.y+AB2.y+crossAB.y;
80      C2.x=A.x+AB2.x+crossAB2.x;C2.y=A.y+AB2.y+crossAB2.y;
81
82      if (getlen(C,C1)<getlen(C,C2))
83      {
84          return C2;
85      }else
86          return C1;
87
88  }
89
90
91  Point getCrossPoint(Point A,Point A1,Point B,Point B1)//得到费马点
92  {
93      Point r;
94      Vec AA(A1.x-A.x,A1.y-A.y),BB(B1.x-B.x,B1.y-B.y);
95      double i,j,tmp,tmp2;
96      double Ax=A.x,Ay=A.y,AAx=AA.x,AAy=AA.y,Bx=B.x,By=B.y,BBx=BB.x,BBy=BB.y;
97
98      if (AAx==0)
99      {
100         j=(Ax-Bx)/BBx;
101         i=(By+BBy*j-Ay)/AAx;
102     }else if (BBx==0)
103     {
104         i=(Bx-Ax)/AAx;
105     }else if (AAy==0)
106     {
```

```
        j=(Ay-By)/BBy;
        i=(Bx-Ax-BBx*j)/AAx;
    }else if (BBy==0)
    {
        i=(By-Ay)/AAy;
    }
    else
    {
        tmp=AAx;
        tmp2=AAy;
        Ax*=AAy;AAx*=AAy;Bx*=AAy;BBx*=AAy;
        Ay*=tmp;AAy*=tmp;By*=tmp;BBy*=tmp;
        j=((Ax-Ay)-(Bx-By))/(BBx-BBy);
        i=(Bx+BBx*j-Ax)/AAx;
    }

    r.x=(Ax+AAx*i)/tmp2;
    r.y=(Ay+AAy*i)/tmp;
    return r;
}
double xa,ya,xb,yb,xc,yc;
double l(pair<double,double>a,pair<double,double>b)
{
    a.fi-=b.fi;a.se-=b.se;
    return sqrt(a.fi*a.fi+a.se*a.se);
}
double fun()
{
    double ans=0.0;
    if(ask(xa,ya,xb,yb,xc,yc)||ask(xb,yb,xa,ya,xc,yc)||ask(xc,yc,xa,ya,xb,yb)){
        vector<double>tmp;
        tmp.pb(l({xa,ya},{xb,yb}));
        tmp.pb(l({xc,yc},{xb,yb}));
        tmp.pb(l({xa,ya},{xc,yc}));
        sort(tmp.begin(),tmp.end());
        return tmp[1];
    }
    Point C1,A1,R;
    C1=getAnotherPoint(Point(xa,ya),Point(xb,yb),Point(xc,yc));
    A1=getAnotherPoint(Point(xc,yc),Point(xb,yb),Point(xa,ya));
    R=getCrossPoint(Point(xa,ya),A1,Point(xc,yc),C1);
    ans=max(max(l({R.x,R.y},{xa,ya}),l({R.x,R.y},{xb,yb})),l({R.x,R.y},{xc,yc}));
    return ans;
}
int main()
{
    int T;scanf("%d",&T);
    while(T--)
    {
        scanf("%lf%lf%lf%lf%lf%lf",&xa,&ya,&xb,&yb,&xc,&yc);
        printf("%.9lf\n",fun());
    }
    return 0;
```

```
}
```

判断三个圆是否有公共点

```
#define point complex<double>
/*
real 返回实部
imag 返回虚部
abs 返回复数的模
arg 返回辐角
*/
bool inter(point a, double r_a, point b, double r_b, point c, double r_c)
{
    if(abs(c - a) <= r_a && abs(c - b) <= r_b)
        return true;
    b -= a, c -= a;
    point r = point(b.real() / abs(b), b.imag() / abs(b));
    b /= r, c /= r;
    double d = (r_a * r_a - r_b * r_b + abs(b) * abs(b)) / (2 * abs(b)),
           h = sqrt(max(r_a * r_a - d * d, 0.0));
    if(abs(h * h + (d - abs(b)) * (d - abs(b)) - r_b * r_b) > eps)
        return false;
    if(abs(point(d, h) - c) <= r_c)
        return true;
    if(abs(point(d, -h) - c) <= r_c)
        return true;
    return false;
}
bool check(point a, double r_a, point b, double r_b, point c, double r_c)
{
    if(r_a <= -eps || r_b <= -eps || r_c <= -eps)
        return false;
    r_a = max(r_a, 0.0), r_b = max(r_b, 0.0), r_c = max(r_c, 0.0);
    if(inter(a, r_a, b, r_b, c, r_c))
        return true;
    if(inter(b, r_b, c, r_c, a, r_a))
        return true;
    if(inter(c, r_c, a, r_a, b, r_b))
        return true;
    return false;
}
```

简单多边形与圆面积交

```
int CircleInterLine( point a, point b, point o, double r, point *p )
{
    point p1 = a - o ;
    point d = b - a ;
    double A = dot( d, d ) ;
    double B = 2 * dot( d, p1 ) ;
```

```
    double C = dot( p1, p1 ) - r*r ;

    double delta = B*B - 4*A*C ;
    if ( sgn(delta) < 0 ) return 0 ;//相离
    if ( sgn(delta) == 0 ) { //相切
        double t = -B / (2*A) ; // 0 <= t <= 1说明交点在线段上
        if ( sgn( t - 1 ) <= 0 && sgn( t ) >= 0 ) {
            p[0] = a + d*t ;
            return 1 ;
        }
    }
    if ( sgn(delta) > 0 ) { //相交
        double t1 = ( -B - sqrt(delta) ) / (2*A) ;
        double t2 = ( -B + sqrt(delta) ) / (2*A) ; //0 <= t1, t2 <= 1说明交点在线段上
        int k = 0 ;
        if ( sgn( t1 - 1 ) <= 0 && sgn( t1 ) >= 0 )
            p[k++] = a + d*t1 ;
        if ( sgn( t2 - 1 ) <= 0 && sgn( t2 ) >= 0 )
            p[k++] = a + d*t2 ;
        return k ;
    }
    return 0;
}
double Triangle_area( point a, point b )
{
    return fabs( det( a , b ) ) / 2.0  ;
}
double Sector_area( point a, point b ,double r)
{
    double ang = atan2( a.y , a.x ) - atan2( b.y, b.x  ) ;
    while ( ang <= 0 ) ang += 2 * PI ;
    while ( ang > 2 * PI ) ang -= 2 * PI ;
    ang = min( ang, 2*PI - ang ) ;
    return r*r * ang/2 ;
}
double calc( point a , point b , double r )
{
    point pi[2] ;
    if ( sgn( a.norm() - r ) < 0 ) {
        if ( sgn( b.norm() - r ) < 0 ) {
            return Triangle_area( a, b ) ;
        }
        else {
            CircleInterLine( a, b, point(0,0), r, pi) ;
            return Sector_area( b, pi[0],r) + Triangle_area( a, pi[0] ) ;
        }
    }
    else {
        int cnt = CircleInterLine( a, b, point(0,0), r, pi ) ;
        if ( sgn( b.norm() - r ) < 0 ) {
            return Sector_area( a, pi[0],r ) + Triangle_area( b, pi[0] ) ;
        }
        else {
```

```
            if ( cnt == 2 )
                return Sector_area( a, pi[0],r ) + Sector_area( b, pi[1],r ) +
Triangle_area( pi[0], pi[1] ) ;
            else
                return Sector_area( a, b ,r) ;
        }
    }
    return 0;
}
double area_CircleAndPolygon( point *p , int n , point o , double r )
{
    double res = 0 ;
    p[n] = p[0] ;
    for ( int i = 0 ; i < n ; i++ ) {
        int tmp = sgn( det( p[i] - o , p[i+1] - o ) ) ;
        if ( tmp )
            res += tmp * calc( p[i] - o , p[i+1] - o , r ) ;
    }
    return fabs( res ) ;
}
// 简单多边形与圆面积交
//p 简单多边形(有序),n 多边形个数，o 圆心 r 半径
```

网络流及图相关

最大权闭合子图

设s为源点，t为汇点。

使s连向所有的正权点（非负权点），边权为点权。

使所有非正权点（负权点）连向t，边权为点权的绝对值。

若需要选y才能选x，连一条由x到y的边，容量是∞。

最大点权和=正权点和-最小割(最大流)

最大流

```
#include <bits/stdc++.h>

using namespace std;
#define ll long long
#define pb push_back
#define mp make_pair
#define fi first
#define se second
const ll INF=1e18+5;
const int N = 1005;
struct Edge
{
    int from,to;
```

```
    ll cap,flow;
};
struct Dinic
{
    int n,m,s,t;
    vector<Edge> edge;
    vector<int> G[N];
    bool vis[N];
    ll d[N];
    int cur[N];

    void init()
    {
        for(int i=0;i<N;i++)
            G[i].clear();
        edge.clear();
        memset(d,0,sizeof(d));
        memset(vis,0,sizeof(vis));
        memset(cur,0,sizeof(cur));
    }

    void addEdge (int from,int to,ll cap)
    {
        edge.push_back((Edge){from,to,cap,0});
        edge.push_back((Edge){to,from,0,0});
        m = edge.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool BFS()
    {
        memset(vis,0,sizeof(vis));
        queue<int> Q;
        Q.push(s);
        d[s] = 0;
        vis[s] = 1;
        while(!Q.empty())
        {
            int x = Q.front();
            Q.pop();
            for(int i=0; i<(int)G[x].size(); i++)
            {
                Edge & e = edge[G[x][i]];
                if(!vis[e.to]&&e.cap>e.flow)
                {
                    vis[e.to] = 1;
                    d[e.to] = d[x] + 1;
                    Q.push(e.to);
                }
            }
        }
        return vis[t];
```

```
    }

    ll DFS(int x,ll a)
    {
        if(x==t||a==0) return a;
        ll flow = 0,f;
        for(int & i = cur[x]; i<(int)G[x].size(); i++)
        {
            Edge & e = edge[G[x][i]];
            if(d[x] + 1==d[e.to]&&(f=DFS(e.to,min(a,e.cap-e.flow)))>0)
            {
                e.flow +=f;
                edge[G[x][i]^1].flow -=f;
                flow +=f;
                a-=f;
                if(a==0) break;
            }
        }
        return flow;
    }

    ll Maxflow (int s,int t) {
        this->s = s;this->t = t;
        ll flow = 0;
        while(BFS()) {
            memset(cur,0,sizeof(cur));
            flow+=DFS(s,INF);
        }
        return flow;
    }

    //求最小割S,T;
    void new_BFS(int s,int n)
    {
        memset(vis,0,sizeof(vis));
        d[s] = 0;
        vis[s] = 1;
        queue<int> Q;
        Q.push(s);
        while(!Q.empty())
        {
            int u = Q.front();
            Q.pop();

            for(int i=0;i<(int)G[u].size();i++)
            {
                Edge & e = edge[G[u][i]];
                if(!vis[e.to]&&e.cap>e.flow)
                {
                    vis[e.to] = 1;
                    d[e.to] = d[u] + 1;
                    Q.push(e.to);
                }
```

```
            }
        }

        int cnt = 0;
        for(int i=1;i<=n;i++)
        {
            if(vis[i]) cnt++;
        }
        printf("%d\n",cnt);
        for(int i=1;i<=n;i++)
            if(vis[i]) printf("%d ",i);
        puts("");
    }

}sol;
int main()
{
    int n,m;scanf("%d%d",&n,&m);
    int x,y;
    map<int,int>id,id1;
    int tot,tot1;tot=tot1=0;
    int vis[105][105];
    memset(vis,0,sizeof(vis));
    while(~scanf("%d%d",&x,&y))
    {
        if(!id[x])id[x]=++tot;
        if(!id1[y])id1[y]=++tot1;
        vis[id[x]][id1[y]]=1;
    }
    sol.init();
    int S,T;S=0;T=N-1;
    for(int i=1;i<=tot;i++)sol.addEdge(S,i,1);
    for(int i=1;i<=tot;i++){
        for(int j=1;j<=tot1;j++){
            if(vis[i][j])
                sol.addEdge(i,tot+j,1);
        }
    }
    for(int i=1;i<=tot1;i++)sol.addEdge(i+tot,T,1);
    int ans=sol.Maxflow(S,T);
    printf("%d\n",ans);
    return 0;
}
```

最小费用最大流

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

#define MAXN 200+10
```

```
#define MAXM 80000+100
#define INF 0x3f3f3f3f

struct Edge
{
    int from, to, cap, flow, cost, next;
};
Edge edge[MAXM];
int head[MAXN], edgenum;
int pre[MAXN];//记录增广路径上 到达点i的边的编号
int dist[MAXN];
bool vis[MAXN];

void init()
{
    edgenum = 0;
    memset(head, -1, sizeof(head));
}
void addEdge(int u, int v, int w, int c)
{
    Edge E1 = {u, v, w, 0, c, head[u]};
    edge[edgenum] = E1;
    head[u] = edgenum++;
    Edge E2 = {v, u, 0, 0, -c, head[v]};
    edge[edgenum] = E2;
    head[v] = edgenum++;
}
bool SPFA(int s, int t)//寻找花销最少的路径
{
    //跑一遍SPFA 找s—t的最少花销路径 且该路径上每一条边不能满流
    //若存在 说明可以继续增广，反之不能
    queue<int> Q;
    memset(dist, INF, sizeof(dist));
    memset(vis, false, sizeof(vis));
    memset(pre, -1, sizeof(pre));
    dist[s] = 0;
    vis[s] = true;
    Q.push(s);
    while(!Q.empty())
    {
        int u = Q.front();
        Q.pop();
        vis[u] = false;
        for(int i = head[u]; i != -1; i = edge[i].next)
        {
            Edge E = edge[i];
            if(dist[E.to] > dist[u] + E.cost && E.cap > E.flow)//可以松弛 且 没有满流
            {
                dist[E.to] = dist[u] + E.cost;
                pre[E.to] = i;//记录前驱边 的编号
                if(!vis[E.to])
                {
                    vis[E.to] = true;
```

```
                    Q.push(E.to);
                }
            }
        }
    }
    return pre[t] != -1;//可达返回true
}
void MCMF(int s, int t, int &cost, int &flow)
{
    flow = 0;//总流量
    cost = 0;//总费用
    while(SPFA(s, t))//每次寻找花销最小的路径
    {
        int Min = INF;
        //通过反向弧 在源点到汇点的最少花费路径 找最小增广流
        for(int i = pre[t]; i != -1; i = pre[edge[i^1].to])
        {
            Edge E = edge[i];
            Min = min(Min, E.cap - E.flow);
        }
        //增广
        for(int i = pre[t]; i != -1; i = pre[edge[i^1].to])
        {
            edge[i].flow += Min;
            edge[i^1].flow -= Min;
            cost += edge[i].cost * Min;//增广流的花销
        }
        flow += Min;//总流量累加
    }
}
int main()
{
    /*while(scanf("%d%d", &N, &M), N||M)
    {
        init();
        getMap();//建图
        int cost, flow;//最小费用 最大流
        MCMF(source, sink, cost, flow);
        printf("%d %d\n", cost, flow);//最小费用 最大流
    }*/
    return 0;
}

```

一般图最大匹配（带花树）

```
int head[T],lst[T*T],nxt[T*T];
int tot,n,m;
void insert(int x,int y)
{
    lst[++tot]=y;
    nxt[tot]=head[x];
```

```
    head[x]=tot;
}
struct Work
{
    int ma[T],st[T],pr[T],fa[T],q[T],v[T];
    int ans,TI,u,t;
    int lca(int x,int y)
    {
        for(TI++;;swap(x,y))if(x)
        {
            if(v[x]==TI)return x;
            v[x]=TI;
            x=fa[pr[ma[x]]];
        }
    }
    void up(int x,int y,int f)
    {
        while(fa[x]!=f)
        {
            pr[x]=y;
            if(st[ma[x]]>0)st[q[++t]=ma[x]]=0;
            if(fa[x]==x)fa[x]=f;
            if(fa[ma[x]]==ma[x])fa[ma[x]]=f;
            x=pr[y=ma[x]];
        }
    }
    int match(int x)
    {
        for(int i=1;i<=n;i++)fa[i]=i,st[i]=-1;
        st[q[t=1]=x]=0;
        for(int l=1;l<=t;l++)for(int i=head[q[l]];i;i=nxt[i])if(st[lst[i]]<0)
        {
            st[lst[i]]=1;
            pr[lst[i]]=q[l];
            if(!ma[lst[i]])
            {
                for(int j=q[l],k=lst[i];j;j=pr[k=u])
                {
                    u=ma[j];
                    ma[j]=k;
                    ma[k]=j;
                }
                return 1;
            }
            st[q[++t]=ma[lst[i]]]=0;
        }
        else if(fa[lst[i]]!=fa[q[l]]&&!st[lst[i]])
        {
            int f=lca(lst[i],q[l]);
            up(q[l],lst[i],f);
            up(lst[i],q[l],f);
            for(int j=1;j<=n;j++)fa[j]=fa[fa[j]];
        }
```

```
        return 0;
    }
    void solve()
    {
        for(int i=1;i<=n;i++)ans+=!ma[i]&&match(i);
        cout<<ans<<endl;
        for(int i=1;i<=n;i++)printf("%d ",ma[i]);
    }
}mp;
```

一般图最大权匹配

```
//from facelessman&vfleaking
#include<bits/stdc++.h>
#define cin kin
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
using namespace std;
typedef long long ll;
const int N=1023,INF=1e9;
struct Edge
{
    int u,v,w;
} g[N][N];
int n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N][N],S[N],vis[N];
vector<int> flower[N];
deque<int> q;
void update_slack(int u,int x)
{
    if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))slack[x]=u;
}
void set_slack(int x)
{
    slack[x]=0;
    for(int u=1; u<=n; ++u)
        if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x)
{
    if(x<=n)return q.push_back(x);
    for(int i=0; i<flower[x].size(); i++)q_push(flower[x][i]);
}
void set_st(int x,int b)
{
    st[x]=b;
    if(x<=n)return;
    for(int i=0; i<flower[x].size(); ++i)set_st(flower[x][i],b);
}
int get_pr(int b,int xr)
{
    int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
    if(pr%2==1) //檢查他在前一層圖是奇點還是偶點
    {
```

```
        reverse(flower[b].begin()+1,flower[b].end());
        return (int)flower[b].size()-pr;
    }
    else return pr;
}
void set_match(int u,int v)
{
    match[u]=g[u][v].v;
    if(u<=n)return;
    Edge e=g[u][v];
    int xr=flower_from[u][e.u],pr=get_pr(u,xr);
    for(int i=0; i<pr; ++i)set_match(flower[u][i],flower[u][i^1]);
    set_match(xr,v);
    rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
}
void augment(int u,int v)
{
    int xnv=st[match[u]];
    set_match(u,v);
    if(!xnv)return;
    set_match(xnv,st[pa[xnv]]);
    augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v)
{
    static int t=0;
    for(++t; u||v; swap(u,v))
    {
        if(u==0)continue;
        if(vis[u]==t)return u;
        vis[u]=t;//這種方法可以不用清空v陣列
        u=st[match[u]];
        if(u)u=st[pa[u]];
    }
    return 0;
}
void add_blossom(int u,int lca,int v)
{
    int b=n+1;
    while(b<=n_x&&st[b])++b;
    if(b>n_x)++n_x;
    lab[b]=0,S[b]=0;
    match[b]=match[lca];
    flower[b].clear();
    flower[b].push_back(lca);
    for(int x=u,y; x!=lca; x=st[pa[y]])
        flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
    reverse(flower[b].begin()+1,flower[b].end());
    for(int x=v,y; x!=lca; x=st[pa[y]])
        flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
    set_st(b,b);
    for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
    for(int x=1; x<=n; ++x)flower_from[b][x]=0;
```

```
 94      for(int i=0; i<flower[b].size(); ++i)
 95      {
 96          int xs=flower[b][i];
 97          for(int x=1; x<=n_x; ++x)
 98              if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
 99                  g[b][x]=g[xs][x],g[x][b]=g[x][xs];
100          for(int x=1; x<=n; ++x)
101              if(flower_from[xs][x])flower_from[b][x]=xs;
102      }
103      set_slack(b);
104  }
105  void expand_blossom(int b)  // S[b] == 1
106  {
107      for(int i=0; i<flower[b].size(); ++i)
108          set_st(flower[b][i],flower[b][i]);
109      int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
110      for(int i=0; i<pr; i+=2)
111      {
112          int xs=flower[b][i],xns=flower[b][i+1];
113          pa[xs]=g[xns][xs].u;
114          S[xs]=1,S[xns]=0;
115          slack[xs]=0,set_slack(xns);
116          q_push(xns);
117      }
118      S[xr]=1,pa[xr]=pa[b];
119      for(int i=pr+1; i<flower[b].size(); ++i)
120      {
121          int xs=flower[b][i];
122          S[xs]=-1,set_slack(xs);
123      }
124      st[b]=0;
125  }
126  bool on_found_Edge(const Edge &e)
127  {
128      int u=st[e.u],v=st[e.v];
129      if(S[v]==-1)
130      {
131          pa[v]=e.u,S[v]=1;
132          int nu=st[match[v]];
133          slack[v]=slack[nu]=0;
134          S[nu]=0,q_push(nu);
135      }
136      else if(S[v]==0)
137      {
138          int lca=get_lca(u,v);
139          if(!lca)return augment(u,v),augment(v,u),1;
140          else add_blossom(u,lca,v);
141      }
142      return 0;
143  }
144  bool matching()
145  {
146      fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
```

```
147     q.clear();
148     for(int x=1; x<=n_x; ++x)
149         if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
150     if(q.empty())return 0;
151     for(;;)
152     {
153         while(q.size())
154         {
155             int u=q.front();
156             q.pop_front();
157             if(S[st[u]]==1)continue;
158             for(int v=1; v<=n; ++v)
159                 if(g[u][v].w>0&&st[u]!=st[v])
160                 {
161                     if(DIST(g[u][v])==0)
162                     {
163                         if(on_found_Edge(g[u][v]))return 1;
164                     }
165                     else update_slack(u,st[v]);
166                 }
167         }
168         int d=INF;
169         for(int b=n+1; b<=n_x; ++b)
170             if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
171         for(int x=1; x<=n_x; ++x)
172             if(st[x]==x&&slack[x])
173             {
174                 if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
175                 else if(S[x]==0)d=min(d,DIST(g[slack[x]][x])/2);
176             }
177         for(int u=1; u<=n; ++u)
178         {
179             if(S[st[u]]==0)
180             {
181                 if(lab[u]<=d)return 0;
182                 lab[u]-=d;
183             }
184             else if(S[st[u]]==1)lab[u]+=d;
185         }
186         for(int b=n+1; b<=n_x; ++b)
187             if(st[b]==b)
188             {
189                 if(S[st[b]]==0)lab[b]+=d*2;
190                 else if(S[st[b]]==1)lab[b]-=d*2;
191             }
192         q.clear();
193         for(int x=1; x<=n_x; ++x)
194             if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(g[slack[x]][x])==0)
195                 if(on_found_Edge(g[slack[x]][x]))return 1;
196         for(int b=n+1; b<=n_x; ++b)
197             if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
198     }
199     return 0;
```

```
}
pair<ll,int> weight_blossom()
{
    fill(match,match+n+1,0);
    n_x=n;
    int n_matches=0;
    ll tot_weight=0;
    for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
    int w_max=0;
    for(int u=1; u<=n; ++u)
        for(int v=1; v<=n; ++v)
        {
            flower_from[u][v]=(u==v?u:0);
            w_max=max(w_max,g[u][v].w);
        }
    for(int u=1; u<=n; ++u)lab[u]=w_max;
    while(matching())++n_matches;
    for(int u=1; u<=n; ++u)
        if(match[u]&&match[u]<u)
            tot_weight+=g[u][match[u]].w;
    return make_pair(tot_weight,n_matches);
}
struct Istream
{
    char b[20<<20],*i,*e;
    Istream(FILE* in):i(b),e(b+fread(b,sizeof(*b),sizeof(b)-1,in)) {}
    Istream& operator>>(int &val)
    {
        while(*i<'0')++i;
        for(val=0; *i>='0'; ++i)val=(val<<3)+(val<<1)+*i-'0';
        return *this;
    }
} kin(stdin);
int main()
{
    cin>>n>>m;
    for(int u=1; u<=n; ++u)
        for(int v=1; v<=n; ++v)
            g[u][v]=Edge {u,v,0};
    for(int i=0,u,v,w; i<m; ++i)
    {
        cin>>u>>v>>w;
        g[u][v].w=g[v][u].w=w;
    }
    cout<<weight_blossom().first<<'\n';
    for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
}
```

KM算法-二分图最大权完美匹配

```
```

```
//o(n^3)
struct KM
{
    #define N 505
    #define inf 0x3fffffff
    bool sx[N],sy[N];
    int match[N],w[N][N],n,m,d,lx[N],ly[N];
    // n: 左集元素个数, m: 右集元素个数 w[][]: 权值
    void init(){
        memset(w,0,sizeof(w));//求最小值初始化为负无穷
    }
    bool dfs(int u){
        sx[u]=true;
        for(int v=0;v<m;v++){
            if(!sy[v] && lx[u]+ly[v] == w[u][v]){
                sy[v]=true;
                if(match[v] == -1 || dfs(match[v])){
                    match[v]=u;
                    return true;
                }
            }
        }
        return false;
    }
    int km(){
        int sum=0;
        memset(ly,0,sizeof(ly));
        for(int i=0;i<n;i++){
            lx[i]=-inf;
            for(int j=0;j<m;j++)if(lx[i]<w[i][j])lx[i]=w[i][j];
        }
        memset(match,-1,sizeof(match));
        for(int i=0;i<n;i++){
            while(true){
                memset(sx,false,sizeof(sx));
                memset(sy,false,sizeof(sy));
                if(dfs(i))break;
                d=inf;
                for(int j=0;j<n;j++){
                    if(sx[j])
                        for(int k=0;k<m;k++)
                            if(!sy[k])d=min(d,lx[j]+ly[k]-w[j][k]);
                }
                if(d==inf)return -1;
                for(int j=0;j<n;j++)if(sx[j])lx[j]-=d;
                for(int j=0;j<m;j++)if(sy[j])ly[j]+=d;
            }
        }
        for(int i=0;i<m;i++)if(match[i]>-1)sum+=w[match[i]][i];
        return sum;
    }
}KM;
```

二分图相关定理

最大匹配数：最大匹配的匹配边的数目

最小点覆盖数：选取最少的点，使任意一条边至少有一个端点被选择

最大独立集：选取最多的点，使任意所选两点均不相连

最小路径覆盖数：选取最少条路径，使得每个顶点属于且仅属于一条路径。路径长可以为 0（即单个点）。

最小点覆盖数 = 最大匹配数

最小路径覆盖 = 顶点数 - 最大匹配数

二分图最大独立集 = 顶点数 - 最大匹配数

二分图匈牙利算法

```
const int maxn = 105;
int p, n;
int ans;
bool visit[3 * maxn];
int Match[3 * maxn];
vector<int> g[maxn];
void init()
{
    ans = 0;
    for(int i = 1; i <= p; i++) g[i].clear();
    memset(Match, -1, sizeof(Match));
}
bool dfs(int u)
{
    for(int i = 0; i < g[u].size(); i++){
        int v = g[u][i];
        if(visit[v]) continue;
        if(Match[v] == -1){Match[v] = u;return true;}
        visit[v] = true;
        if(dfs(Match[v])){Match[v] = u;return true;}
    }
    return false;
}
int work()
{
    for(int i = 1; i <= p; i++){
        memset(visit, false, sizeof(visit));
        if(dfs(i)) ans++;
    }
    return ans;
}
```

二分图hopcroft_karp算法（匈牙利算法的优化）

```
//可能不会快多少。。。
struct Edge
{
    int v,next;
}edge[N*N];

int cnt,head[N];
int xline[N],yline[N],dy[N],dx[N];///xline表示与x配对的y编号,yline表示与y配对的x编号,dy,dx表示在各自集合里的编号
int vis[N],dis;

void addedge(int u,int v)
{
    edge[cnt].v=v;
    edge[cnt].next=head[u];
    head[u]=cnt++;
}
void init()
{
    cnt=0;
    memset(head,-1,sizeof(head));
    memset(xline,-1,sizeof(xline));
    memset(yline,-1,sizeof(yline));
}
int bfs()
{
    queue<int>que;
    dis=INF;
    memset(dx,-1,sizeof(dx));
    memset(dy,-1,sizeof(dy));
    for(int i=1;i<=m;i++)
    {
        if(xline[i]==-1)
        {
            que.push(i);
            dx[i]=0;
        }
    }
    while(!que.empty())
    {
        int u=que.front();que.pop();
        if(dx[u]>dis) break;
        for(int i=head[u];i!=-1;i=edge[i].next)
        {
            int v = edge[i].v;
            if(dy[v] == -1)
            {
                dy[v] = dx[u] + 1;
                if(yline[v] == -1) dis = dy[v];
                else
                {
                    dx[yline[v]] = dy[v]+1;
                    que.push(yline[v]);
```

```
                }
            }
        }
    }
    return dis!=INF;
}
int can(int t)
{
    for(int i=head[t];i!=-1;i=edge[i].next)
    {
        int v=edge[i].v;
        if(!vis[v]&&dy[v]==dx[t]+1)
        {
            vis[v]=1;
            if(yline[v]!=-1&&dy[v]==dis) continue;
            if(yline[v]==-1||can(yline[v]))
            {
                yline[v]=t,xline[t]=v;
                return 1;
            }
        }
    }
    return 0;
}
int Maxmatch()///最大匹配
{
    int ans=0;
    while(bfs())
    {
        memset(vis,0,sizeof(vis));
        for(int i=1;i<=m;i++)
            if(xline[i]==-1&&can(i))
                ans++;
    }
    return ans;
}

```

无向图最大完全子团大小（NP-hard）

```
const int maxn = 55;
bool mp[maxn][maxn];
int best, n, num[maxn];
bool dfs(int *adj, int total, int cnt)
{
    int t[maxn], k;
    if(total == 0)
    {
        if(cnt > best)
        {
```

```
            best = cnt;
            return true;    //剪枝4
        }
        return false;
    }
    for(int i = 0; i < total; ++i)
    {
        if(cnt+total-i <= best) return false;   //剪枝1
        if(cnt+num[adj[i]] <= best) return false;   //剪枝3
        k = 0;
        for(int j = i+1; j < total; ++j)
        if(mp[adj[i]][adj[j]]) t[k++] = adj[j];
        //扫描与u相连的顶点中与当前要选中的adj[i]相连的顶点adj[j]并存储到数组t[]中，数量为k
        if(dfs(t, k, cnt+1)) return true;
    }
    return false;
}
int MaximumClique()
{
    int adj[maxn], k;
    best = 0;
    for(int i = n; i >= 1; --i)
    {
        k = 0;
        for(int j = i+1; j <= n; ++j)
        if(mp[i][j]) adj[k++] = j;
        //得到当前点i的所有相邻点存入adj
        dfs(adj, k, 1); //每次dfs相当于必选当前i点看是否能更新best
        num[i] = best;
    }
    return best;
}
```

无向图最大团个数（NP-hard）

最大团的个数 = 补图的最大独立数

```
//一个点集S被称为极大团，当且仅当S中的所有点均互为朋友，且所有不在S中的人，均与S中的某些人不是朋友。
const int maxn = 130;
bool mp[maxn][maxn];
int some[maxn][maxn], none[maxn][maxn], all[maxn][maxn];
int n, m, ans;
void dfs(int d, int an, int sn, int nn)
{
    if(!sn && !nn) ++ans;
    int u = some[d][0];
    for(int i = 0; i < sn; ++i)
    {
        int v = some[d][i];
        if(mp[u][v]) continue;
        for(int j = 0; j < an; ++j)
        all[d+1][j] = all[d][j];
```

```
        all[d+1][an] = v;
        int tsn = 0, tnn = 0;
        for(int j = 0; j < sn; ++j)
        if(mp[v][some[d][j]])
        some[d+1][tsn++] = some[d][j];
        for(int j = 0; j < nn; ++j)
        if(mp[v][none[d][j]])
        none[d+1][tnn++] = none[d][j];
        dfs(d+1, an+1, tsn, tnn);
        some[d][i] = 0, none[d][nn++] = v;
        if(ans > 1000) return;
    }
}
int work()
{
    ans = 0;
    for(int i = 0; i < n; ++i) some[1][i] = i+1;
    dfs(1, 0, n, 0);
    return ans;
}
```

字符串相关

最长公共子序列（LCS)

```
随机数据可以建图dp
```

```
for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            if(a[i]==b[j])dp[i][j]=dp[i-1][j-1]+1;
            else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
        }
    }
    stack<char>stk;int i=n,j=m;
    while(i&&j){
        if(a[i]==b[j]){stk.push(a[i]);i--;j--;
        }else if(dp[i][j]==dp[i-1][j])i--;
        else j--;
    }
```

KMP算法

```
KMP最小循环节、循环周期

    对于next数组中的i，符合i%(i-next[i])==0&&next[i]!=0，则字符串[0,...,i-1]循环

    循环节长度：i-next[i]              循环次数：i/(i-next[i])

    补全成循环串最少需要字符数：i-next[i]-i%(i-next[i])
```

```
void getfill(char *s,int len)
{
    memset(Next,0,sizeof(Next));
    for(int i=1;i<len;i++){
        int j=Next[i];
        while(j&&s[i]!=s[j])j=Next[j];
        Next[i+1]=(s[i]==s[j])?j+1:0;
    }
}
int sum(char *s,int lens,char *t,int lent)
{
    int j=0,cnt=0;
    for(int i=0;i<lent;i++){
        while(j&&t[i]!=s[j])j=Next[j];
        if(t[i]==s[j])j++;
        if(j==lens)cnt++;
    }return cnt;
}
```

后缀数组

后缀数组sa[i]表示排名为i的后缀的起始位置的下标

映射数组rk[i]表示起始位置的下标为i的后缀的排名

辅助工具：最长公共前缀LCP

定义LCP(i,j)为suff(sa[i])与suff(sa[j])的最长公共前缀

定理：

$LCP(i,j) = LCP(j,i)$

$LCP(i,i) = len(sa[i]) = n - sa[i] + 1$

$LCP(i,k) = min(LCP(i,j), LCP(j,k))1 <= i <= j <= k <= n$

$LCP(i,k) = min(LCP(j,j-1))1 <= i < j <= k <= n$

设height[i]为LCP(i,i-1) 1<i<=n height[1]=0

则$LCP(i,k) = min(height[j])i < j <= k$

应用：

最长公共子串：两个字符串拼起来找符合条件的最大的height[i]

```
int s[N],sa[N],sa2[N],rk[N],cnt[N],height[N];
bool same(int *rank, int idx1, int idx2, int len)
{
    return rank[idx1]==rank[idx2] && rank[idx1+len]==rank[idx2+len];
}
// s:输入字符串的末尾要补一个 '0' , n是字符串的实际长度+1.
//m: 单字符rank的范围，辅助变量[0,m)
void SA(int *s, int *sa, int *sa2, int *rk, int *cnt, int *hgt, int n, int m)
{
    for(int i=0; i<m; i++) cnt[i]=0;
    for(int i=0; i<n; i++) cnt[rk[i]=s[i]]++;
    for(int i=1; i<m; i++) cnt[i]+=cnt[i-1];
    for(int i=n-1; i>=0; i--) sa[--cnt[rk[i]]]=i;
    for(int len=1; len<n; len*=2){
       int p=0;
        for(int i=n-len; i<n; i++) sa2[p++]=i;
        for(int i=0; i<n; i++)
            if(sa[i]>=len)
                sa2[p++]=sa[i]-len;
        for(int i=0; i<m; i++) cnt[i]=0;
        for(int i=0; i<n; i++) cnt[rk[i]]++;
        for(int i=1; i<m; i++) cnt[i]+=cnt[i-1];
        for(int i=n-1; i>=0; i--)
            sa[--cnt[rk[sa2[i]]]]=sa2[i];
        swap(rk, sa2);
        rk[sa[0]]=0;
        for(int i=1; i<n; i++)
            rk[sa[i]]=rk[sa[i-1]]+!same(sa2, sa[i-1], sa[i], len);
        m=rk[sa[n-1]]+1;
        if(m==n) break;
    }
    for(int i=0, j, lcp=0; i<n-1; i++){
        lcp?--lcp:0;
        j=sa[rk[i]-1];
        for(; s[j+lcp]==s[i+lcp]; lcp++);
        hgt[rk[i]]=lcp;
    }
}
```

后缀自动机（待补）

```
后缀自动机DAWG(s)中后缀链接组成的树就是后缀树ST(rev(s))

Next形成的DAG图中，每条从st[0]开始的路径形成的子串都是原串本质不同的子串的集合元素之一，且两者相等
```

```
struct state{
    int len,link,Next[30];
};
const int MAXLEN = 1e5+5;
state st[MAXLEN*2];
```

```
int sz,last;
void sa_init()
{
    sz=last=0;
    st[0].len=0;
    st[0].link=-1;
    ++sz;
    for(int i=0;i<MAXLEN*2;i++) memset(st[i].Next,0,sizeof(st[i].Next));
}
void sa_extend(char c)
{
    int cur=sz++;
    st[cur].len=st[last].len+1;
    int p;
    for(p=last;p!=-1&&!st[p].Next[c-'a'];p=st[p].link) st[p].Next[c-'a']=cur;
    if(p==-1)st[cur].link=0;
    else{
        int q=st[p].Next[c-'a'];
        if(st[p].len+1==st[q].len) st[cur].link=q;
        else{
            int clone=sz++;
            st[clone].len=st[p].len+1;
            memcpy(st[clone].Next,st[q].Next,sizeof(st[q].Next));
            st[clone].link=st[q].link;
            for(;p!=-1&&st[p].Next[c-'a']==q;p=st[p].link) st[p].Next[c-'a']=clone;
            st[q].link=st[cur].link=clone;
        }
    }
    last=cur;
}
```

两个字符串的最长子串

```

pair<int,int> lcs(char *s,int lens,char *t,int lent)//return {len,pos(t)}
{
    sa_init();
    for(int i=0;i<lens;i++) sa_extend(s[i]);
    int v=0,l=0,best=0,bestpos=0;
    for(int i=0;i<lent;i++){
        while(v&&!st[v].Next[t[i]-'a']){v=st[v].link;l=st[v].len;}
        if(st[v].Next[t[i]-'a']){v=st[v].Next[t[i]-'a'];l++;}
        if(l>best)best=l,bestpos=i;
    }
    return {best,bestpos-best+1};
}
```

本质不同的字符串个数

```
每次增加一个字符所产生的贡献cnt是节点的len减去失配节点的len,每次有新节点或节点都要相应增减贡献
```

```
void sa_extend(char c)
{
    ...
    if(p==-1)st[cur].link=0,cnt+=st[cur].len;
    else{
        ...
        if(st[p].len+1==st[q].len) st[cur].link=q,cnt+=st[cur].len-st[q].len;
        else{
            ...
            st[clone].link=st[q].link;
            cnt+=st[clone].len-st[st[clone].link].len;
            for(;p!=-1&&st[p].Next[c-'a']==q;p=st[p].link) st[p].Next[c-'a']=clone;
            cnt-=st[q].len-st[st[q].link].len;
            st[q].link=st[cur].link=clone;
            cnt+=st[q].len+st[cur].len-2*st[clone].len;
        }
    }
    last=cur;
}
```

本质不同的字符串长度之和

```
对DAG图dp即可
```

字典序第k小本质不同子串

```
就是求字典序第k小路径，对DAG图dp后即可逐位确定答案
```

```
//预处理每个节点的cnt代表该节点下有多少个子节点也就是多少个子串
int id[N],num[N];
long long cnt[N];
void diff(int u)
{
    memset(num,0,sizeof(num));
    for(int i=1;i<sz;i++)num[st[i].len]++;
    for(int i=1;i<sz;i++)num[i]+=num[i-1];
    for(int i=1;i<sz;i++)id[num[st[i].len]--]=i;
    for(int i=sz-1;i>=0;i--){
        int now=id[i];
        cnt[now]=1;
        for(int j=0;j<30;j++){
            if(st[now].Next[j])
                cnt[now]+=cnt[st[now].Next[j]];
        }
    }
}
```

一个串出现几次

1	**每次新增一个点就在后缀树上将从根到该节点的路径上加一，节点值就是从st[0]到该点所形成子串的出现次数，用LCT维护可做到在线**

最小循环移位

1	**给定字符串S，找到和它循环同构的字典序最小字符串**
2	
3	**解法：对S+S建立自动机后找字典序最小长度为len(S)的路径，贪心即可**

序列自动机

$next[i][j]$表示在原串i之后第一个j出现的位置

```
for(int i=n;i;i--)
{
    for(int j=1;j<=a;j++) next[i-1][j]=next[i][j];
    next[i-1][s[i]]=i;
}
```

本质不同子序列个数

dp[i]表示前i个数的不同子序列个数，对于a[i]

1.在之前没出现过，dp[i]=dp[i-1]+dp[i-1]+1

2.在之前出现过，最近的位置为x，dp[i]=dp[i-1]+dp[i-1]-dp[x-1]

两个串的公共子序列个数

```
int dfs(int x,int y)//表示目前字符是串1的第x位，串2的第y位
{
    if(f[x][y]) return f[x][y];
    for(r int i=1;i<=a;i++)
        if(next1[x][i]&&next2[y][i]) f[x][y]+=dfs(next1[x][i],next2[y][i]);
    return ++f[x][y];
}
//调用：dfs(0,0);
```

一个串的回文子序列个数

对原串和反串分别构造next数组

```
int dfs(int x,int y)
{
    if(f[x][y]) return f[x][y];//记忆化
    for(int i=1;i<=26;i++)
        if(next1[x][i]&&next2[y][i])
        {
            if(next1[x][i]+next2[y][i]>n+1) continue;//x+y>n+1,状态不合法,不进行统计
            if(next1[x][i]+next2[y][i]<n+1) f[x][y]++;
            //满足x+y=n+1的奇串不会被漏掉,其他奇串需要特别统计
            f[x][y]=(f[x][y]+dfs(next1[x][i],next2[y][i]))%mod;
        }
    return ++f[x][y];
}
```

求A、B最长子序列，满足是C的子序列

改变C的next数组然后dfs(0,0,0)即可

```
for(int i=1;i<=26;i++) nextc[n][i]=n;//字符集为26个字母,C长度为n
for(int i=0;i<n;i++)
{
    for(r int j=1;j<=26;j++) nextc[i][j]=i;
    nextc[i][c[i+1]]=i+1;
}
```

树相关（待补）

树链剖分

```
int sz[N],top[N],son[N],dep[N],fa[N],id[N],rid[N],cnt;//区间[1,]
void init(){
    cnt=1;
    memset(son,0,sizeof(son));
}
void dfs1(int u,int father,int depth){
    dep[u]=depth;fa[u]=father;sz[u]=1;
    for(int i=head[u];i;i=edg[i].next){
        int v=edg[i].to;
        if(v!=fa[u]){
            dfs1(v,u,depth+1);
            sz[u]+=sz[v];
            if(!son[u]||sz[v]>sz[son[u]])son[u]=v;
        }
    }
}
void dfs2(int u,int t){
    top[u]=t;id[u]=cnt;rid[cnt]=u;cnt++;
    if(!son[u])return;
    dfs2(son[u],t);
    for(int i=head[u];i;i=edg[i].next){
```

```
        int v=edg[i].to;
        if(v!=son[u]&&v!=fa[u])dfs2(v,v);
    }

}
ll query_path(int x,int y){
    ll ans=0;
    int fx=top[x],fy=top[y];
    while(fx!=fy){
        if(dep[fx]>=dep[fy]){
            ans+=query(1,1,n,id[fx],id[x]);
            x=fa[fx];
        }else{
            ans+=query(1,1,n,id[fy],id[y]);
            y=fa[fy];
        }
        fx=top[x],fy=top[y];
    }
    if(x!=y){
        if(id[x]<id[y]) ans+=query(1,1,n,id[x],id[y]);
        else ans+=query(1,1,n,id[y],id[x]);
    }else  ans+=query(1,1,n,id[x],id[y]);
    return ans;
}
void update_path(int x,int y,int value){
    int fx=top[x],fy=top[y];
    while(fx!=fy){
        if(dep[fx]>dep[fy]){
            update(1,1,n,id[fx],id[x],value);
            x=fa[fx];
        }else{
            update(1,1,n,id[fy],id[y],value);
            y=fa[fy];
        }
        fx=top[x],fy=top[y];
    }
    if(x!=y){
        if(id[x]<id[y])update(1,1,n,id[x],id[y],value);
        else update(1,1,n,id[y],id[x],value);
    }else update(1,1,n,id[x],id[y],value);
}
```

左偏树

splay

```
const int N=200005, inf=0x3f3f3f3f;

typedef struct splaynode* node;
struct splaynode {
    node pre, ch[2];
    ll value, lazy, min, sum;
```

```
    int size, rev;
    void init(int _value) {
        pre=ch[0]=ch[1]=NULL;
        min=value=sum=_value;
        lazy=rev=0;
        size=1;
    }
}mem[N];
int memtop;
stack<node> S;
node root;
inline int getsize(node &x) {
    return x ? x->size : 0;
}
void pushdown(node &x) {
    if (!x) return;
    if (x->lazy) {
        ll w = x->lazy;
        x->value += w;
        if (x->ch[0]) {
            x->ch[0]->lazy += w;
            x->ch[0]->min += w;
            x->ch[0]->sum += w*getsize(x->ch[0]);
        }
        if (x->ch[1]) {
            x->ch[1]->lazy += w;
            x->ch[1]->min += w;
            x->ch[1]->sum += w*getsize(x->ch[1]);
        }
        x->lazy = 0;
    }
    if (x->rev) {
        node t = x->ch[0];
        x->ch[0] = x->ch[1];
        x->ch[1] = t;
        x->rev = 0;
        if (x->ch[0]) x->ch[0]->rev ^= 1;
        if (x->ch[1]) x->ch[1]->rev ^= 1;
    }
}
void update(node &x) {
    if (!x) return;
    x->size = 1;
    x->min = x->value;
    x->sum = x->value;
    if (x->ch[0]) {
        x->sum += x->ch[0]->sum;
        x->min = min(x->min, x->ch[0]->min);
        x->size += x->ch[0]->size;
    }
    if (x->ch[1]) {
        x->sum += x->ch[1]->sum;
        x->min = min(x->min, x->ch[1]->min);
```

```
        x->size += x->ch[1]->size;
    }
}
void rotate(node &x, int d) {
    node y = x->pre;
    pushdown(y);
    pushdown(x);
    pushdown(x->ch[d]);
    y->ch[!d] = x->ch[d];
    if (x->ch[d] != NULL) x->ch[d]->pre = y;
    x->pre = y->pre;
    if (y->pre != NULL){
        if (y->pre->ch[0] == y) y->pre->ch[0] = x; else y->pre->ch[1] = x;
    }
    x->ch[d] = y;
    y->pre = x;
    update(y);
    if (y == root) root = x;
}
void splay(node &src, node &dst) {
    pushdown(src);
    while (src!=dst) {
        if (src->pre==dst) {
            if (dst->ch[0]==src) rotate(src, 1); else rotate(src, 0);
            break;
        }
        else {
            node y=src->pre, z=y->pre;
            if (z->ch[0]==y) {
                if (y->ch[0]==src) {
                    rotate(y, 1);
                    rotate(src, 1);
                }else {
                    rotate(src, 0);
                    rotate(src, 1);
                }
            }
            else {
                if (y->ch[1]==src) {
                    rotate(y, 0);
                    rotate(src, 0);
                }else {
                    rotate(src, 1);
                    rotate(src, 0);
                }
            }
            if (z==dst) break;
        }
        update(src);
    }
    update(src);
}
void select(int k, node &f) {
```

```
    int tmp;
    node t = root;
    while (1) {
        pushdown(t);
        tmp = getsize(t->ch[0]);
        if (k == tmp + 1) break;
        if (k <= tmp) t = t->ch[0];
        else {
            k -= tmp + 1;
            t = t->ch[1];
        }
    }
    pushdown(t);
    splay(t, f);
}
inline void selectsegment(int l,int r) {
    select(l, root);
    select(r + 2, root->ch[1]);
}

void insert(int pos, int value) {  //在pos位置后面插入一个新值value
    selectsegment(pos + 1, pos);
    node t;
    node x = root->ch[1];
    pushdown(root);
    pushdown(x);
    if (!S.empty()) {
        t = S.top();
        S.pop();
    } else {
        t = &mem[memtop++];
    }
    t->init(value);
    t->ch[1] = x;
    x->pre = t;
    root->ch[1] = t;
    t->pre = root;
    splay(x, root);
}
void add(int a,int b, int value) {  //区间[a,b]中的数都加上value
    selectsegment(a, b);
    node x = root->ch[1]->ch[0];
    pushdown(x);
    update(x);
    x->min += value;
    x->lazy += value;
    splay(x, root);
}
void reverse(int a, int b) {   //区间[a,b]中的数翻转
    selectsegment(a, b);
    root->ch[1]->ch[0]->rev ^= 1;
    node x = root->ch[1]->ch[0];
    splay(x, root);
```

```
}
void revolve(int a, int b, int t) { //区间[a,b]中的数向后循环移t位
    int tn=b-a+1;
    t=(t%tn+tn)%tn;
    if(a==b||t==0)return;
    node p1, p2;
    selectsegment(a, b);
    select(b + 1 - t, root->ch[1]->ch[0]);
    p1 = root->ch[1]->ch[0];
    pushdown(p1);
    p2 = p1->ch[1];
    p1->ch[1] = NULL;

    select(a + 1, root->ch[1]->ch[0]);
    p1 = root->ch[1]->ch[0];
    pushdown(p1);
    p1->ch[0] = p2;
    p2->pre = p1;

    splay(p2, root);
}

ll getmin(int a, int b) {   //取[a,b]中最大的值
    selectsegment(a, b);
    node x = root->ch[1];
    pushdown(x);
    x = x->ch[0];
    pushdown(x);
    update(x);
    return x->min;
}
ll getsum(int a, int b) { //[a,b]区间和
    selectsegment(a, b);
    node x = root->ch[1];
    pushdown(x);
    x = x->ch[0];
    pushdown(x);
    update(x);
    return x->sum;
}
void erase(int pos) {               //抹去第pos个元素
    selectsegment(pos, pos);
    pushdown(root->ch[1]);
    S.push(root->ch[1]->ch[0]);        //回收内存
    root->ch[1]->ch[0] = NULL;
    node x = root->ch[1];
    splay(x, root);
}

void cutandmove(int a,int b,int c)//移动区间到位置c后
{
    selectsegment(a,b);
    node CutRoot=root->ch[1]->ch[0];
```

```
    CutRoot->pre=NULL;
    root->ch[1]->size-=CutRoot->size;
    root->ch[1]->ch[0]=NULL;

    selectsegment(c+1,c);

    CutRoot->pre=root->ch[1];
    root->ch[1]->ch[0]=CutRoot;
    root->ch[1]->size+=CutRoot->size;
}

void cut(int a,int b)//删除区间
{
    selectsegment(a,b);
    node CutRoot=root->ch[1]->ch[0];
    CutRoot->pre=NULL;
    root->size-=CutRoot->size;
    root->ch[1]->size-=CutRoot->size;
    root->ch[1]->ch[0]=NULL;
}

vector<int> ans;
void inorder(node x)
{
    if (!x) return;
    pushdown(x);
    inorder(x->ch[0]);
    if (x->value!=inf) ans.push_back(x->value);
    inorder(x->ch[1]);
}

void initsplaytree(ll *a, int n) {
    memtop = 0;
    root = &mem[memtop++];
    root->init(inf);
    root->ch[1] = &mem[memtop++];
    root->ch[1]->init(inf);
    while (!S.empty()) S.pop();
    rep(i) insert(i, a[i]);
}
```

link-cut tree

```
int mx[N],stk[N],top;
bool rev[N];
int s[N], val[N], ch[N][2], fa[N],lazy[N];
inline bool wh(int p) {return ch[fa[p]][1] == p;}
inline bool Isroot(int p) {return ch[fa[p]][wh(p)] != p;}
inline void Update(int p) {/*mx[p] = val[p] ^ mx[ch[p][0]] ^ mx[ch[p][1]];*/}//根据题目修改
inline void Update_add(int u,int v){if(!u)return ;s[u]+=v;lazy[u]+=v;}
void Pushdown(int p) {
    if(rev[p]) {
```

```
        rev[p] ^= 1; swap(ch[p][0], ch[p][1]);
        rev[ch[p][1]] ^= 1; rev[ch[p][0]] ^= 1;
    }
    if(lazy[p]){
        Update_add(ch[p][0],lazy[p]);
        Update_add(ch[p][1],lazy[p]);
        lazy[p]=0;
    }
}
inline void Pushup(int p) {
    /*if(!Isroot(p))Pushup(fa[p]);
    Pushdown(p);*/
    top = 0; stk[++top] = p;
    for(int i = p; !Isroot(i); i = fa[i]) stk[++top] = fa[i];
    for(int i = top; i; --i) Pushdown(stk[i]);
}

void Rotate(int p) {
    int f = fa[p], g = fa[f], c = wh(p);
    if(!Isroot(f)) ch[g][wh(f)] = p; fa[p] = g;
    ch[f][c] = ch[p][c ^ 1]; if(ch[f][c]) fa[ch[f][c]] = f;
    ch[p][c ^ 1] = f; fa[f] = p;
    Update(f);
}
void Splay(int p) {
    Pushup(p);
    for(; !Isroot(p); Rotate(p))
        if(!Isroot(fa[p])) Rotate(wh(fa[p]) == wh(p) ? fa[p] : p);
    Update(p);
}
void Access(int p) {
    for(int pre = 0; p; pre = p, p = fa[p]) {
        Splay(p);
        ch[p][1] = pre;
        Update(p);
    }
}
inline void Makeroot(int p) {Access(p); Splay(p); rev[p] ^= 1;}
inline int Find(int p) {for(Access(p), Splay(p); ch[p][0]; p = ch[p][0]) ; return p;}
inline void Change(int u, int v) {Access(u); Splay(u); val[u] = v; Update(u);}//单点更新
inline void Add(int u,int v,int w){Makeroot(u); Access(v); Splay(v);Update_add(v,w);}//区间更新[1,...]
inline void Cut(int u, int v) {Makeroot(u); Access(v); Splay(v); ch[v][0] = fa[u] = 0;Add(v,1,-s[u]);}
inline void Link(int u, int v) {Makeroot(u); fa[u] = v;Add(v,1,s[u]);}
inline int Query(int u, int v) {Makeroot(u); Access(v); Splay(v); return s[v];}
```

Euler-Tour tree