# 数学

## 组合

### 卡特兰数

长度为 $2n$ 的合法括号序列个数。

$$C_{2n}^{n} - C_{2n}^{n-1}$$

`1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700 ...`

### 斯特林数

- 第一类斯特林数 $s(n,k)$：长度为 $n$ 的排列中出现 $k$ 个环的方案数。

• $s(n,k)=s(n-1,k-1)+(n-1)s(n-1,k)$

• 第二类斯特林数 $S(n,k)$：长度为 $n$ 的排列中出现 $k$ 个集合的方案数。

• $S(n,k)=S(n-1,k-1)+kS(n-1,k)$

## Burnside 引理 & Pólya 定理

给定集合 $X$ 和**置换群** $G$。

> 如果感觉到十分抽象的话，不妨代入下面的情境（意义）对下文的内容进行理解：
>
> $A$ 表示待染色物品**集合**。
>
> $B$ 表示支持染的颜色的**集合**。
>
> $X$ 表示染色方案**集合**。
>
> $G$ 表示支持的变换。
>
> $X/G$ 表示**本质不同**的染色方案**集合**。
>
> $X^g$ 表示经过一个变换 $g$ 后保持不变的**染色方案**对应的**集合**。
>
> 比如说：给你一串共 $n$ 个珠子，支持旋转变换 $G$（可以看作是置换的一种）（这意味着如果两种染色方案在旋转后一样视为**本质相同**），每个珠子可以被染成 $m$ 种颜色（也就是说方案集 $X$ 的大小为 $m^n$），求本质不同的染色方案数（也就是 $|X/G|$）

### Burnside 引理：

$$|X/G|=\frac{1}{|G|}\sum_{g\in G}|X^g|$$

这意味着，对于实际的计数问题，在给出变换的种类数 $|G|$ 后，我们只需要求出染色方案在各种变换下保持不动的数量的和（即 $\sum_{g\in G}|X^g|$），那么我们就可以求出本质不同的染色方案数了。

### Pólya 定理

考虑到 Burnside 引理需要求的 $\sum_{g\in G}|X^g|$ 在实际统计中时间复杂度较高，而很多问题都是求解 $A\to B$ **所有可能的映射**（也就是比较特殊的 $X$）所对应的染色方案（共 $|B|^{|A|}$ 种）中**本质不同**的数量。

那么，在这样的问题中，可以使用 **Pólya 定理**：

$$|X/G|=\frac{1}{|G|}\sum_{g\in G}|B|^{c(g)}$$

> $c(g)$ 表示置换 $g$ 能拆分出的循环置换数。

### 例题

给定一个 $n$ 个点，$n$ 条边的环，有 $n$ 种颜色，给每个顶点染色，问有多少种**本质不同**的染色方案，答案对 $10^9+7$ 取模

注意本题的本质不同，定义为：**只需要不能通过旋转与别的染色方案相同。**

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

```cpp
using namespace std;

#define int long long

const int mod=1e9+7;

int fpow(int x, int p){
    int res=1;
    for(; p; p>>=1, x=x*x%mod) if(p&1) res=res*x%mod;
    return res;
}

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

int inv(int x){
    return fpow(x, mod-2);
}

int phi(int x){
    int res=x;
    for(int i=2;i<=x/i;i++)
        if(x%i==0){
            res=res/i*(i-1);
            while(x%i==0) x/=i;
        }
    if(x>1) res=res/x*(x-1);

    return res;
}

signed main(){
    int T; cin>>T;
    while(T--){
        int n; cin>>n;
        int res=0;
        for(int i=1; i<=n/i; i++){
            if(n%i==0){
                res=(res+fpow(n, i)*phi(n/i)%mod)%mod;
                if(n/i!=i) res=(res+fpow(n, n/i)*phi(i)%mod)%mod;
            }
        }
        cout<<res*inv(n)%mod<<endl;
    }
    return 0;
}
```

# 数论

## 线性筛

每个数只会被最小的质因数计算一次。

```
int cnt, p[N];
bool vis[N];

int n,q;

void sieve(){
    for(int i=2; i<N; i++){
        if(!vis[i]) p[cnt++]=i;
        for(int j=0; i*p[j]<N; j++){
            vis[i*p[j]]=true;
            if(i%p[j]==0) break;
        }
    }
}
```

## 线性求逆元

`inv[i] = (ll) (p - p / i) * inv[p%i] % p;`

## Lucas

$x, y \leq 10^{18}$。$p$ 是质数。$p \leq 10^5$。

```
ll C(ll a, ll b){
    if(b>a) return 0;
    return fac[a]*inv(fac[b])%mod*inv(fac[a-b])%mod;
}

ll lucas(ll a, ll b){
    if(!b) return 1;
    return C(a%mod, b%mod)*lucas(a/mod, b/mod)%mod;
}
```

## exgcd

```
int exgcd(int a, int b, int &x, int &y){
    if(!b) return x=1, y=0, a;
    int d=exgcd(b, a%b, y, x);
    y-=a/b*x;
    return d;
}
```

## exEuler

介绍：若 $b \geq \phi(m)$，那么 $a^b \equiv a^{b\%\phi(m)+\phi(m)} \pmod m$。

https://www.luogu.com.cn/problem/P5091

给你三个正整数，$a, m, b$，你需要求：$a^b \bmod m$

【数据范围】

对于 $100\%$ 的数据，$1 \leq a \leq 10^9$，$1 \leq b \leq 10^{20000000}$，$1 \leq m \leq 10^8$。

```
#define int long long

int get_phi(int m){
    int res=m;
    for(int i=2; i<=m/i; i++){
        if(m%i==0){
            res=res/i*(i-1);
            while(m%i==0) m/=i;
        }
    }
    if(m>1) res=res/m*(m-1);
    return res;
}

int fpow(int x, int p, int mod){
    int res=1;
    for(; p; p>>=1, x=x*x%mod) if(p&1) res=res*x%mod;
    return res;
}

int Val(string B, int mod){
    int res=0;
    for(auto i: B) (res=res*10+i-'0')%=mod;
    return res;
}

int get(int b, int phi){
    return b%phi+phi;
}

int to_int(string B){
    int res=0;
    for(auto i: B) res=res*10+i-'0';
    return res;
}

signed main(){
    int a, m; cin>>a>>m;
    string B; cin>>B;
    int phi;
    phi=get_phi(m);

    int pw;
    if(B.size()>8) pw=get(Val(B, phi), phi), cout<<fpow(a, pw, m);
    else{
        int b=to_int(B);
        if(b<phi) cout<<fpow(a, b, m);
        else cout<<fpow(a, b%phi+phi, m);
```

```
    }

    return 0;
}
```

## bsgs

https://www.luogu.com.cn/problem/P3846

给定一个质数 $p$，以及一个整数 $b$，一个整数 $n$，现在要求你计算一个最小的非负整数 $l$，满足 $b^l \equiv n \pmod p$。

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

#define int long long

int bsgs(int a, int p, int b){
    if(1%p==b%p) return 0;
    int k=sqrt(p)+1;
    unordered_map<int, int> hash;

    for(int i=0, j=b%p; i<k; i++){
        hash[j]=i;
        j=j*a%p;
    }
    int ak=1;
    for(int i=0; i<k; i++) ak=ak*a%p; // get a^k
    for(int i=1, j=ak; i<=k; i++){
        if(hash.count(j)) return i*k-hash[j];
        j=j*ak%p;
    }
    return -1;
}

signed main(){
    int a, p, b;
    while(cin>>p>>a>>b){
        int res=bsgs(a, p, b);
        if(res==-1) puts("no solution");
        else cout<<res<<endl;
    }
    return 0;
}
```

- exbsgs
- 给定 $a, p, b$，求满足 $a^x \equiv b \pmod p$ 的最小自然数 $x$ 。
- 保证 $\sum \sqrt{p} \leq 5 \times 10^6$。
- ```
  #define int long long

  const int INF=1e8;
  ```
  ```

```cpp
int exgcd(int a, int b, int &x, int &y){
    if(!b){
        x=1, y=0;
        return a;
    }
    int d=exgcd(b, a%b, y, x);
    y-=a/b*x;
    return d;
}

int bsgs(int a, int b, int p){
    if(1%p==b%p) return 0;
    int k=sqrt(p)+1;
    unordered_map<int, int> hash;

    for(int i=0, j=b%p; i<k; i++){
        hash[j]=i;
        j=j*a%p;
    }
    int ak=1;
    for(int i=0; i<k; i++) ak=ak*a%p; // get a^k
    for(int i=1, j=ak; i<=k; i++){
        if(hash.count(j)) return i*k-hash[j];
        j=j*ak%p;
    }
    return -INF;
}

int exbsgs(int a, int b, int p){
    b=(b%p+p)%p;
    if(1%p==b%p) return 0;
    int x, y;
    int d=exgcd(a, p, x, y);
    if(d>1){
        if(b%d) return -INF;
        exgcd(a/d, p/d, x, y);
        return exbsgs(a, b/d*x%(p/d), p/d)+1;
    }
    return bsgs(a, b, p);
}

signed main(){
    int a, b, p;
    while(cin>>a>>p>>b, a || b || p){
        int res=exbsgs(a, b, p);
        if(res<0) puts("No Solution");
        else cout<<res<<endl;
    }
    return 0;
}
```

## exCRT

https://www.luogu.com.cn/problem/P4777

给定 $n$ 组非负整数 $a_i, b_i$，求解关于 $x$ 的方程组的最小非负整数解。

$$\begin{cases} x \equiv b_1 \pmod{a_1} \\ x \equiv b_2 \pmod{a_2} \\ \cdots \\ x \equiv b_n \pmod{a_n} \end{cases}$$

```
const int N=1e5+50;

int n, M[N], A[N];

int mul(int x, int p, int mod){
    int res=0;
    for(; p; p>>=1, x=(x+x)%mod) if(p&1) res=(res+x)%mod;
    return res;
}

int exgcd(int a, int b, int &x, int &y){
    if(!b) return x=1, y=0, a;
    int d=exgcd(b, a%b, y, x);
    y-=a/b*x;
    return d;
}

int exCRT(int n, int *A, int *M){
    int x, y;
    int m=M[1], res=(A[1]%m+m)%m;
    rep(i,2,n){
        int a=m, b=M[i], d=((A[i]-res)%b+b)%b;
        int g=exgcd(a, b, x, y);
        if(d%g) return -1;

        int P=b/g;
        x=mul(x, d/g, P);
        res+=x*m, m*=P;
        (res=res%m+m)%=m;
    }
    return res;
}

signed main(){
    cin>>n;
    rep(i,1,n) read(M[i]), read(A[i]);

    cout<<exCRT(n, A, M)<<endl;

    return 0;
}
```

## miller-rabin

用费马小定理和二次剩余来判断质数。

```
inline int mul(int a, int b, int mod){
    return (__int128)a*b%mod;
}

inline int fpow(int x, int p, int mod){
    int res=1;
    for(; p; p>>=1, x=mul(x, x, mod)) if(p&1) res=mul(res, x, mod);
    return res;
}

inline bool rabin(int n, int a){
    int d=n-1, ct=0;
    while(~d&1) d>>=1, ct++;
    int x=fpow(a, d, n);
    if(x==1) return true;
    rep(i,1,ct){
        if(x==n-1 || x==1) return true;
        x=mul(x, x, n);
    }
    return false;
}


inline bool isp(int n){
    if(n<2) return false;
    static const int pri[] = {2, 3, 5, 7, 11, 13, 31, 61, 24251};
    for(auto x: pri) {
        if(n==x) return true;
        if(!rabin(n, x)) return false;
    }
    return true;
}
```

## Pollard-Rho 算法

https://www.luogu.com.cn/problem/P4718

用于在 $O(n^{1/4})$ 的期望时间复杂度内计算合数 $n$ 的某个非平凡因子。

对于每个数字检验是否是质数，是质数就输出 `Prime`；如果不是质数，输出它最大的质因子是哪个。

给出完整代码：

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

#define debug(x) cerr << #x << ": " << (x) << endl
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define dwn(i,a,b) for(int i=(a);i>=(b);i--)
#define pb push_back
#define all(x) (x).begin(), (x).end()

#define x first
#define y second
using pii = pair<int, int>;
```

```cpp
using ll = long long;

#define int long long

inline void read(int &x){
    int s=0; x=1;
    char ch=getchar();
    while(ch<'0' || ch>'9') {if(ch=='-')x=-1;ch=getchar();}
    while(ch>='0' && ch<='9') s=(s<<3)+(s<<1)+ch-'0',ch=getchar();
    x*=s;
}

const int N=1e5+5, s=20;

mt19937_64 rd(time(NULL));
int fac[N], cnt;

inline int mul(int a, int b, int mod){
    return (__int128)a*b%mod;
}

inline int fpow(int x, int p, int mod){
    int res=1;
    for(; p; p>>=1, x=mul(x, x, mod)) if(p&1) res=mul(res, x, mod);
    return res;
}

inline bool rabin(int n, int a){
    int d=n-1, ct=0;
    while(~d&1) d>>=1, ct++;
    int x=fpow(a, d, n);
    if(x==1) return true;
    rep(i,1,ct){
        if(x==n-1 || x==1) return true;
        x=mul(x, x, n);
    }
    return false;
}


inline bool isp(int n){
    if(n<2) return false;
    static const int pri[] = {2, 3, 5, 7, 11, 13, 31, 61, 24251};
    for(auto x: pri) {
        if(n==x) return true;
        if(!rabin(n, x)) return false;
    }
    return true;
}

inline int f(int x, int c, int mod){
    return (mul(x, x, mod)+c)%mod;
}

inline int pollard_rho(int n){
    int c=rd()%(n-1)+1; int v1=0;
    for(int s=1, t=2; ; s<<=1, t<<=1){
        int pro=1, v2=v1, cnt=0;
```

```
        rep(i,s+1,t){
            v2=f(v2, c, n), pro=mul(pro, abs(v1-v2), n), cnt++;
            if(cnt%127==0){
                cnt=0; int d=__gcd(pro, n);
                if(d>1) return d;
            }
        }
        int d=__gcd(pro, n);
        if(d>1) return d;
        v1=v2;
    }
}

inline void find(int n){
    if(isp(n)) return fac[++cnt]=n, void();
    int p;
    while((p=pollard_rho(n))==n) ;
    find(p), find(n/p);
}

signed main(){
    int T; cin>>T;
    while(T--){
        cnt=0;
        int n; read(n);
        if(isp(n)){
            puts("Prime");
            continue;
        }
        find(n);
        int res=0;
        rep(i,1,cnt) if(fac[i]!=n) res=max(res, fac[i]);
        cout<<res<<endl;
    }
    return 0;
}
```

# 线性代数

## 高斯消元

https://www.luogu.com.cn/problem/P3389

即求解方程 $AX = Y$ 的 $X$。

```
const int N=110;
const double eps=1e-8;

int n;
```

```
double b[N][N];

void guass(){
    int rk=0;
    for(int r=1, c=1; r<=n; c++, r++){
        int t=r;
        for(int i=r+1; i<=n; i++) if(fabs(b[i][c])>fabs(b[t][c])) t=i;
        if(fabs(b[t][c])<eps) continue;
        ++rk;
        for(int i=c; i<=n+1; i++) swap(b[r][i], b[t][i]);
        for(int i=n+1; i>=c; i--) b[r][i]/=b[r][c];
        for(int i=r+1; i<=n; i++) for(int j=n+1; j>=c; j--) b[i][j]-=b[i]
[c]*b[r][j];
    }
    for(int i=n; i>1; i--) for(int j=i-1; j; j--) b[j][n+1]-=b[i][n+1]*b[j][i],
b[j][i]=0;
    if(rk!=n){
        puts("No Solution");
        return;
    }
    for(int i=1; i<=n; i++) printf("%.2lf\n", b[i][n+1]);
}

signed main(){
    cin>>n;
    rep(i,1,n) rep(j,1,n+1) cin>>b[i][j];
    guass();
    return 0;
}
```

## 矩阵求逆

求一个 $N \times N$ 的矩阵的逆矩阵。答案对 $10^9 + 7$ 取模。

```
#define int long long

inline void read(int &x){
    int s=0; x=1;
    char ch=getchar();
    while(ch<'0' || ch>'9') {if(ch=='-')x=-1;ch=getchar();}
    while(ch>='0' && ch<='9') s=(s<<3)+(s<<1)+ch-'0',ch=getchar();
    x*=s;
}

const int N=405, mod=1e9+7;

int n, a[N][N<<1];

int fpow(int x, int p){
    int res=1;
    for(; p; p>>=1, x=1LL*x*x%mod) if(p&1) res=1LL*res*x%mod;
    return res%mod;
}

int inv(int x){
```

```
    return fpow(x, mod-2);
}

void mat_inv(){
    for(int i=1; i<=n; i++){
        int r=i;
        for(int j=i+1; j<=n; j++) if(a[j][i]>a[r][i]) r=j;
        if(r!=i) swap(a[i], a[r]);
        if(!a[i][i]) return puts("No Solution"), void();

        int Inv=inv(a[i][i]);
        for(int k=1; k<=n; k++){
            if(k==i) continue;
            int p=a[k][i]*Inv%mod;
            for(int j=i; j<=(n<<1); ++j) a[k][j]=((a[k][j]-p*a[i]
[j])%mod+mod)%mod;
        }
        for(int j=1; j<=(n<<1); j++) a[i][j]=(a[i][j]*Inv%mod);
    }
    for(int i=1; i<=n; i++){
        for(int j=n+1; j<=(n<<1); j++) cout<<a[i][j]<<" ";
        cout<<endl;
    }
}

signed main(){
    cin>>n;
    rep(i,1,n) rep(j,1,n) read(a[i][j]);
    rep(i,1,n) a[i][i+n]=1;
    mat_inv();
    return 0;
}
```

## 行列式求值（取模）

https://www.luogu.com.cn/problem/P7112

求取模意义下 $w[][]$ 的行列式的值，也就是计算 $|A|$。

```
const int N=610;

int n, mod;
int w[N][N];

int getDet(){
    int res=1;
    rep(i,1,n){
        rep(j,i+1,n){
            while(w[i][i]){
                int div=w[j][i]/w[i][i];
                rep(k,i,n) w[j][k]=(w[j][k]-1LL*div*w[i][k]%mod+mod)%mod;
                res=-res, swap(w[i], w[j]);
            }
            res=-res, swap(w[i], w[j]);
```

```
        }
    }
    rep(i,1,n) res=1LL*res*w[i][i]%mod;
    return (res%mod+mod)%mod;
}

signed main(){
    while(cin>>n>>mod){
        rep(i,1,n) rep(j,1,n) read(w[i][j]);
        cout<<getDet()<<endl;
    }
    return 0;
}
```

## 线性基

给定 $n$ 个整数（数字可能重复），求在这些数中选取任意个，使得他们的异或和最大。

- 传统写法：

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

typedef long long ll;
const int N=55;

ll base[N];

int main(){
    int n; cin>>n;
    while(n--){
        ll k; cin>>k;
        for(int j=N-1;~j;j--){
            if(k&(1LL<<j)){
                if(!base[j]) base[j]=k;
                k^=base[j];
            }
        }
    }

    ll ans=0;
    for(int j=N-1;~j;j--){
        if((ans^base[j])>ans) ans=ans^base[j];
    }

    cout<<ans<<endl;

    return 0;
}
```

- 我之前在 Atcoder 学到的简单写法：

```
signed main(){
    int n; cin>>n;
    vector<int> w(n), d;
```

```
    rep(i,0,n-1) read(w[i]);

    for(auto i: w){
        int val=i;
        for(auto j: d) if((val^j)<val) val^=j;
        if(val) d.push_back(val);
    }

    int res=0;
    for(auto i: d) res=max(res, res^i);
    cout<<res<<endl;

    return 0;
}
```

## LGV 引理

```
// Problem: P6657 【模板】LGV 引理
// Contest: Luogu
// URL: https://www.luogu.com.cn/problem/P6657
// Memory Limit: 128 MB
// Time Limit: 1000 ms
//
// Powered by CP Editor (https://cpeditor.org)

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

#define debug(x) cerr << #x << ": " << (x) << endl
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define dwn(i,a,b) for(int i=(a);i>=(b);i--)
#define pb push_back
#define all(x) (x).begin(), (x).end()

#define x first
#define y second
using pii = pair<int, int>;
using ll = long long;

#define int long long

inline void read(int &x){
    int s=0; x=1;
    char ch=getchar();
    while(ch<'0' || ch>'9') {if(ch=='-')x=-1;ch=getchar();}
    while(ch>='0' && ch<='9') s=(s<<3)+(s<<1)+ch-'0',ch=getchar();
    x*=s;
}

const int N=2e6+5, M=110, mod=998244353;

int fpow(int x, int p){
    int res=1;
    for(; p; p>>=1, x=1LL*x*x%mod) if(p&1) res=1LL*res*x%mod;
    return res%mod;
```

```
}

int inv(int x){
    return fpow(x, mod-2);
}

int fac[N];
int invfac[N];

void getFac(int n){
    fac[0]=invfac[0]=1;
    for(int i=1; i<=n; i++) fac[i]=1LL*fac[i-1]*i%mod, invfac[i]=1LL*invfac[i-
1]*inv(i)%mod;
}

int C(int a, int b){
    if(b>a) return 0;
    return 1LL*fac[a]*invfac[b]%mod*invfac[a-b]%mod;
}

int n, m;
int a[M], b[M];

int f[M][M];

int det(){
    int res=1, sig=1;
    rep(i,1,m){
        int t=0;
        rep(j,i,m) if(f[j][i]) t=j;
        if(!t) return 0;
        if(i!=t){
            sig*=-1;
            rep(j,i,m) swap(f[i][j], f[t][j]);
        }
        rep(j,i+1,m){
            int rate=f[j][i]*inv(f[i][i])%mod;
            rep(k,i,m) f[j][k]-=f[i][k]*rate%mod, f[j][k]=(f[j][k]%mod+mod)%mod;
        }
        (res*=f[i][i])%=mod;
    }
    res*=sig;
    return (res%mod+mod)%mod;
}

signed main(){
    getFac(N-1);
    int cs; cin>>cs;
    while(cs--){
        read(n), read(m);
        rep(i,1,m) read(a[i]), read(b[i]);
        rep(i,1,m) rep(j,1,m) f[i][j]=C(n-1+b[j]-a[i], n-1);
        cout<<det()<<endl;
    }
    return 0;
}
```

# 多项式及计数技巧

**NTT 相关**是同学分享的板子。

## NTT 相关

```
#include <bits/stdc++.h>
#define fp(i, a, b) for (int i = (a), i##_ = (b) + 1; i < i##_; ++i)
#define fd(i, a, b) for (int i = (a), i##_ = (b) - 1; i > i##_; --i)
#define MUL(a, b) ((ll)(a) * (b) % P) // 大模数时这里ll要改成__int128
#define ADD(a, b) (((a) += (b)) >= P ? (a) -= P : 0) // (a += b) %= P
#define SUB(a, b) (((a) -= (b)) < 0 ? (a) += P: 0) // ((a -= b) += P) %= P
#define int long long
using namespace std;
const int N = 3e5 + 5, P = 998244353; //模数
using ll = int64_t;
using Poly = vector<int>;
/*------------------------------------------------------------------------------*/
class Cipolla {
    int P, I2 {};
    using pll = pair<ll, ll>;
#define X first
#define Y second
    ll mul(ll a, ll b) const {
        return a * b % P;
    }
    pll mul(pll a, pll b) const {
        return {(a.X * b.X + I2 * a.Y % P * b.Y) % P,
                (a.X * b.Y + a.Y * b.X) % P
               };
    }
    template<class T> T POW(T a, int b, T x) {
        for (; b; b >>= 1, a = mul(a, a))
            if (b & 1) x = mul(x, a);
        return x;
    }
public:
    Cipolla(int p = 0) : P(p) {}
    pair<int, int> sqrt(int n) {
        int a = rand(), x;
        if (!(n %= P)) return {0, 0};
        if (POW(n, (P - 1) >> 1, 1ll) == P - 1) return {-1, -1};
        while (POW(I2 = ((ll) a * a - n + P) % P, (P - 1) >> 1, 1ll) == 1) a =
                rand();
        x = (int) POW(pll {a, 1}, (P + 1) >> 1, {1, 0}).X;
        if (2 * x > P) x = P - x;
        return {x, P - x};
    }
#undef X
#undef Y
};
/*------------------------------------------------------------------------------*/
Poly getInv(int L) {
```

```
    Poly inv(L);
    inv[1] = 1;
    fp(i, 2, L - 1)
    inv[i] = MUL((P - P / i), inv[P % i]);
    return inv;
}
ll POW(ll a, ll b = P - 2, ll x = 1) {
    for (; b; b >>= 1, a = MUL(a, a))
        if (b & 1) x = MUL(x, a);
    return x;
}
auto inv = getInv(N); // NOLINT
/*------------------------------------------------------------------------*/
namespace NTT {
    const int g = 3; //原根
    Poly Omega(int L) {
        int wn = POW(g, P / L);
        Poly w(L);
        w[L >> 1] = 1;
        fp(i, L / 2 + 1, L - 1) w[i] = MUL(w[i - 1], wn);
        fd(i, L / 2 - 1, 1) w[i] = w[i << 1];
        return w;
    }
    auto W = Omega(1 << 21); // NOLINT 初始化 2的次幂大于等于NTT长度
    void DIF(int *a, int n) {
        for (int k = n >> 1; k; k >>= 1)
            for (int i = 0, y; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    y = a[i + j + k], a[i + j + k] = MUL(a[i + j] - y + P, W[k +
                                                                             j]), ADD(a[i + j], y);
    }
    void IDIT(int *a, int n) {
        for (int k = 1; k < n; k <<= 1)
            for (int i = 0, x, y; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    x = a[i + j], y = MUL(a[i + j + k], W[k + j]),
                    a[i + j + k] = x - y < 0 ? x - y + P : x - y, ADD(a[i + j],
                                                                     y);
        int Inv = P - (P - 1) / n;
        fp(i, 0, n - 1) a[i] = MUL(a[i], Inv);
        reverse(a + 1, a + n);
    }
}
/*------------------------------------------------------------------------*/
namespace Polynomial {
    // basic operator
    int norm(int n) {
        return 1 << (32 - __builtin_clz(n - 1));
    }
    void norm(Poly &a) {
        if (!a.empty()) a.resize(norm(a.size()), 0);
        else a =
        {0};
    }
    void DFT(Poly &a) {
        NTT::DIF(a.data(), a.size());
    }
    void IDFT(Poly &a) {
```

```
        NTT::IDIT(a.data(), a.size());
    }
    Poly &dot(Poly &a, Poly &b) {
        fp(i, 0, a.size() - 1) a[i] = MUL(a[i], b[i]);
        return a;
    }
    // mul / div int
    Poly &operator*=(Poly &a, int b) {
        for (auto &x : a) x = MUL(x, b);
        return
            a;
    }
    Poly operator*(Poly a, int b) {
        return a *= b;
    }
    Poly operator*(int a, Poly b) {
        return b * a;
    }
    Poly &operator/=(Poly &a, int b) {
        return a *= POW(b);
    }
    Poly operator/(Poly a, int b) {
        return a /= b;
    }
    // Poly add / sub
    Poly &operator+=(Poly &a, Poly b) {
        a.resize(max(a.size(), b.size()));
        fp(i, 0, b.size() - 1) ADD(a[i], b[i]);
        return a;
    }
    Poly operator+(Poly a, Poly b) {
        return a += b;
    }
    Poly &operator-=(Poly &a, Poly b) {
        a.resize(max(a.size(), b.size()));
        fp(i, 0, b.size() - 1) SUB(a[i], b[i]);
        return a;
    }
    Poly operator-(Poly a, Poly b) {
        return a -= b;
    }
    // Poly mul
    Poly operator*(Poly a, Poly b) {
        int n = a.size() + b.size() - 1, L = norm(n);
        if (a.size() <= 8 || b.size() <= 8) {
            Poly c(n);
            fp(i, 0, a.size() - 1) fp(j, 0, b.size() - 1)
            c[i + j] = (c[i + j] + (ll) a[i] * b[j]) % P;
            return c;
        }
        a.resize(L), b.resize(L);
        DFT(a), DFT(b), dot(a, b), IDFT(a);
        return a.resize(n), a;
    }
    // Poly inv
    Poly Inv2k(Poly a) { // |a| = 2 ^ k
        int n = a.size(), m = n >> 1;
        if (n == 1) return {POW(a[0])};
```

```
    Poly b = Inv2k(Poly(a.begin(), a.begin() + m)), c = b;
    b.resize(n), DFT(a), DFT(b), dot(a, b), IDFT(a);
    fp(i, 0, n - 1) a[i] = i < m ? 0 : P - a[i];
    DFT(a), dot(a, b), IDFT(a);
    return move(c.begin(), c.end(), a.begin()), a;
}
Poly Inv(Poly a) {
    int n = a.size();
    norm(a), a = Inv2k(a);
    return a.resize(n), a;
}
// Poly div / mod
Poly operator/(Poly a,Poly b) {
    int k = a.size() - b.size() + 1;
    if (k < 0) return {0};
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    b.resize(k), a = a * Inv(b);
    a.resize(k), reverse(a.begin(), a.end());
    return a;
}
pair<Poly, Poly> operator%(Poly a, const Poly& b) {
    Poly c = a / b;
    a -= b * c, a.resize(b.size() - 1);
    return {c, a};
}
// Poly calculus
Poly deriv(Poly a) {
    fp(i, 1, a.size() - 1) a[i - 1] = MUL(i, a[i]);
    return a.pop_back(), a;
}
Poly integ(Poly a) {
    a.push_back(0);
    fd(i, a.size() - 1, 1) a[i] = MUL(inv[i], a[i - 1]);
    return a[0] = 0, a;
}
// Poly ln
Poly Ln(Poly a) {
    int n = a.size();
    a = deriv(a) * Inv(a);
    return a.resize(n - 1), integ(a);
}
// Poly exp
Poly Exp(Poly a) {
    int n = a.size(), k = norm(n);
    Poly b = {1}, c, d;
    a.resize(k);
    for (int L = 2; L <= k; L <<= 1) {
        d = b, b.resize(L), c = Ln(b), c.resize(L);
        fp(i, 0, L - 1) c[i] = a[i] - c[i] + (a[i] < c[i] ? P : 0);
        ADD(c[0], 1), DFT(b), DFT(c), dot(b, c), IDFT(b);
        move(d.begin(), d.end(), b.begin());
    }
    return b.resize(n), b;
}
// Poly sqrt
Poly Sqrt(Poly a) {
    int n = a.size(), k = norm(n);
```

```
        a.resize(k);
        Poly b = {(new Cipolla(P))->sqrt(a[0]).first, 0}, c;
        for (int L = 2; L <= k; L <<= 1) {
            b.resize(L), c = Poly(a.begin(), a.begin() + L) * Inv2k(b);
            fp(i, L / 2, L - 1) b[i] = MUL(c[i], (P + 1) / 2);
        }
        return b.resize(n), b;
    }
    // Poly pow
    Poly Pow(Poly &a, int b) {
        return Exp(Ln(a) * b);    // a[0] = 1
    }
    Poly Pow(Poly a, int b1, int b2) { // b1 = b % P, b2 = b % phi(P) and b >= n
iff a[0] > 0
        int n = a.size(), d = 0, k;
        while (d < n && !a[d]) ++d;
        if ((ll) d * b1 >= n) return Poly(n);
        a.erase(a.begin(), a.begin() + d);
        k = POW(a[0]), norm(a *= k);
        a = Pow(a, b1) * POW(k, P - 1 - b2);
        a.resize(n), d *= b1;
        fd(i, n - 1, 0) a[i] = i >= d ? a[i - d] : 0;
        return a;
    }
    // Get [x ^ k](f / g)
    int divAt(Poly f, Poly g, ll k) {
        int n = max(f.size(), g.size()), m = norm(n);
        for (; k; k >>= 1) {
            f.resize(m * 2, 0), DFT(f);
            g.resize(m * 2, 0), DFT(g);
            fp(i, 0, 2 * m - 1) f[i] = MUL(f[i], g[i ^ 1]);
            fp(i, 0, m - 1) g[i] = MUL(g[2 * i], g[2 * i + 1]);
            g.resize(m), IDFT(f), IDFT(g);
            for (int i = 0, j = k & 1; i < n; i++, j += 2) f[i] = f[j];
            f.resize(n), g.resize(n);
        }
        return f[0];
    }
    // Get a[k] by a[n] = sum c[i] * a[n - i]
    int LinearRecur(Poly a, Poly c, ll k) {
        c[0] = P - 1, a = a * c, a.resize(c.size() - 1);
        return divAt(a, c, k);
    }
    //Poly cyc循环卷积
    Poly cyc(Poly a, Poly b) {
        int n = a.size();
        reverse(a.begin(), a.end());
        b.insert(b.end(), b.begin(), b.end());
        a = a * b;
        fp(i, 0, n - 1) a[i] = a[i + n - 1];
        return a.resize(n), a;
    }

    //Poly subpoly差卷积
    Poly subpoly(Poly a, Poly b) {
        int n = a.size();
        int m = b.size();
        reverse(b.begin(), b.end());
```

```
        Poly c = a * b;
        fp(i, 0, n - 1) c[i] = c[i + m - 1];
        return c.resize(n), c;
    }
}
/*-------------------------------------------------------------------------*/
Poly BerlekampMassey(Poly &a) {
    Poly c, las, cur;
    int k, delta, d, w;
    fp(i, 0, a.size() - 1) {
        d = P - a[i];
        fp(j, 0, c.size() - 1) d = (d + (ll) a[i - j - 1] * c[j]) % P;
        if (!d) continue;
        if (c.empty()) {
            k = i, delta = d, c.resize(i + 1);
            continue;
        }
        cur = c, w = POW(delta, P - 2, P - d);
        if (c.size() < las.size() + i - k) c.resize(las.size() + i - k);
        SUB(c[i - k - 1], w);
        fp(j, 0, las.size() - 1) c[i - k + j] = (c[i - k + j] + (ll) w * las[j])
                                                % P;
        if (cur.size() <= las.size() + i - k) las = cur, k = i, delta = d;
    }
    return c;
}
/*-------------------------------------------------------------------------*/
using namespace Polynomial;
```

## 二项式反演

$g_n$ 表示钦定 $n$ 个位置满足要求的方案数；$f_n$ 表示恰好 $n$ 个位置满足要求的方案数。

因为 $f$ 一般难求，所以考虑利用 $g$ 求出 $f$。

我们有公式：

$$g_n = \sum_{i=n}^{m} C_i^n f_i$$

$$f_n = \sum_{i=n}^{m} (-1)^{i-n} C_i^n g_i$$

使用多项式科技可以在 $O(n\log n)$ 复杂度求出 $f$。（上式可以化为卷积形式）

下示例代码是求字符串刚好有 $k$ 个子段 `bit` 的方案数。

```
using namespace Polynomial;

const int mod=P;

int n;

int fpow(int x, int p){
    if(p<0) return 0;
    int res=1;
```

```
    for(; p; p>>=1, x=1LL*x*x%mod) if(p&1) res=1LL*res*x%mod;
    return res%mod;
}

int Inv(int x){
    return fpow(x, mod-2);
}

int fac[N];
int invfac[N];

void getFac(int n){
    fac[0]=invfac[0]=1;
    for(int i=1; i<=n; i++) fac[i]=1LL*fac[i-1]*i%mod, invfac[i]=1LL*invfac[i-
1]*Inv(i)%mod;
}

int C(int a, int b){
    if(b>a) return 0;
    return 1LL*fac[a]*invfac[b]%mod*invfac[a-b]%mod;
}

Poly binoinv(Poly &G){
    Poly P, Q(n+1), R(n+1);
    int sig=1;
    for(int i=0; i<=n; i++) Q[i]=G[i]*fac[i]%mod, R[i]=(sig*invfac[i]+mod)%mod,
sig*=-1;
    P=subpoly(Q, R);
    for(int i=0; i<=n; i++) P[i]=P[i]*invfac[i]%mod;
    return P;
}

Poly calG(){
    Poly G(n+1);
    for(int k=0; k<=n; k++) G[k]=C(n-2*k, k)*fpow(26, n-3*k)%mod;
    return G;
}

signed main(){
    cin>>n;
    getFac(n);

    auto G=calG();
    auto res=binoinv(G);
    for(auto &i: res) cout<<i<<' ';
    puts("");

    return 0;
}
```

## FFT

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

```
const int N=3e6+5;
const double pi=acos(-1);

int n, m;

struct Complex{
    double x, y;
    Complex operator + (const Complex &o)const { return {x+o.x, y+o.y}; }
    Complex operator - (const Complex &o)const { return {x-o.x, y-o.y}; }
    Complex operator * (const Complex &o)const { return {x*o.x-y*o.y,
x*o.y+y*o.x}; }
};

Complex a[N], b[N];
int res[N];

int rev[N], bit, tot;

void fft(Complex a[], int inv){
    for(int i=0; i<tot; i++) if(i<rev[i]) swap(a[i], a[rev[i]]);

    for(int mid=1; mid<tot; mid<<=1){
        auto w1=Complex({cos(pi/mid), inv*sin(pi/mid)});
        for(int i=0; i<tot; i+=mid*2){
            auto wk=Complex({1, 0});
            for(int j=0; j<mid; j++, wk=wk*w1){
                auto x=a[i+j], y=wk*a[i+j+mid];
                a[i+j]=x+y, a[i+j+mid]=x-y;
            }
        }
    }
}

int main(){
    cin>>n>>m;
    for(int i=0; i<=n; i++) cin>>a[i].x;
    for(int i=0; i<=m; i++) cin>>b[i].x;

    while((1<<bit)<n+m+1) bit++;

    tot=1<<bit;

    for(int i=0; i<tot; i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));

    fft(a, 1), fft(b, 1);

    for(int i=0; i<tot; i++) a[i]=a[i]*b[i];

    fft(a, -1);

    for(int i=0; i<=n+m; i++) res[i]=(int)(a[i].x/tot+0.5), printf("%d ",
res[i]);

    return 0;
}
```

# 任意模数 NTT

```
// Problem: P4245 【模板】任意模数多项式乘法
// Contest: Luogu
// URL: https://www.luogu.com.cn/problem/P4245
// Memory Limit: 500 MB
// Time Limit: 2000 ms
//
// Powered by CP Editor (https://cpeditor.org)

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

#define debug(x) cerr << #x << ": " << (x) << endl
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define dwn(i,a,b) for(int i=(a);i>=(b);i--)
#define pb push_back
#define all(x) (x).begin(), (x).end()

#define x first
#define y second
using pii = pair<int, int>;

inline void read(int &x){
    int s=0; x=1;
    char ch=getchar();
    while(ch<'0' || ch>'9') {if(ch=='-')x=-1;ch=getchar();}
    while(ch>='0' && ch<='9') s=(s<<3)+(s<<1)+ch-'0',ch=getchar();
    x*=s;
}

#define fp(i, a, b) for (int i = (a), i##_ = (b) + 1; i < i##_; ++i)
#define fd(i, a, b) for (int i = (a), i##_ = (b) - 1; i > i##_; --i)
using namespace std;
const int N = 2e5 + 5;
using ll = int64_t;
using db = double;
/*------------------------------------------------------------------------------*/
struct cp {
    db x, y;
    cp(db real = 0, db imag = 0) : x(real), y(imag){};
    cp operator+(cp b) const { return {x + b.x, y + b.y}; }
    cp operator-(cp b) const { return {x - b.x, y - b.y}; }
    cp operator*(cp b) const { return {x * b.x - y * b.y, x * b.y + y * b.x}; }
};
using vcp = vector<cp>;
using Poly = vector<int>;
namespace FFT {
    const db pi = acos(-1);
    vcp Omega(int L) {
        vcp w(L); w[1] = 1;
        for (int i = 2; i < L; i <<= 1) {
            auto w0 = w.begin() + i / 2, w1 = w.begin() + i;
            cp wn(cos(pi / i), sin(pi / i));
            for (int j = 0; j < i; j += 2)
                w1[j] = w0[j >> 1], w1[j + 1] = w1[j] * wn;
        }
```

```
        return w;
    }
    auto W = Omega(1 << 18); // NOLINT
    void DIF(cp *a, int n) {
        cp x, y;
        for (int k = n >> 1; k; k >>= 1)
            for (int i = 0; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    x = a[i + j], y = a[i + j + k],
                    a[i + j + k] = (x - y) *  W[k + j], a[i + j] = x + y;
    }
    void IDIT(cp *a, int n) {
        cp x, y;
        for (int k = 1; k < n; k <<= 1)
            for (int i = 0; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    x = a[i + j], y = a[i + j + k] * W[k + j],
                    a[i + j + k] = x - y, a[i + j] = x + y;
        const db Inv = 1. / n;
        fp(i, 0, n - 1) a[i].x *= Inv, a[i].y *= Inv;
        reverse(a + 1, a + n);
    }
}
/*--------------------------------------------------------------------------*/
namespace MTT{
    Poly conv(const Poly &a, const Poly &b, const int&P) {
        int n = a.size(), m = b.size(), o = n + m - 1, l = 1 << (__lg(o - 1) +
1);
        vcp A(l), B(l), c0(l), c1(l);
        for (int i = 0; i < n; i++) A[i] = cp(a[i] & 0x7fff, a[i] >> 15);
        for (int i = 0; i < m; i++) B[i] = cp(b[i] & 0x7fff, b[i] >> 15);
        FFT::DIF(A.data(), l), FFT::DIF(B.data(), l);
        for (int k = 1, i = 0, j; k < l; k <<= 1)
            for (; i < k * 2; i++) {
                j = i ^ k - 1;
                c0[i] = cp(A[i].x + A[j].x, A[i].y - A[j].y) * B[i] * 0.5;
                c1[i] = cp(A[i].y + A[j].y, -A[i].x + A[j].x) * B[i] * 0.5;
            }
        FFT::IDIT(c0.data(), l), FFT::IDIT(c1.data(), l);
        Poly res(o);
        for (int i = 0; i < o; i++) {
            ll c00 = c0[i].x + 0.5, c01 = c0[i].y + 0.5, c10 = c1[i].x + 0.5,
c11 = c1[i].y + 0.5;
            res[i] = (c00 + ((c01 + c10) % P << 15) + (c11 % P << 30)) % P;
        }
        return res;
    }
}
/*--------------------------------------------------------------------------*/
namespace Polynomial {
    // basic operator
    void DFT(vcp &a) { FFT::DIF(a.data(), a.size()); }
    void IDFT(vcp &a) { FFT::IDIT(a.data(), a.size()); }
    int norm(int n) { return 1 << (__lg(n - 1) + 1); }

    // Poly mul
    vcp &dot(vcp &a, vcp &b) { fp(i, 0, a.size() - 1) a[i] = a[i] * b[i]; return
a; }
```

```
    Poly operator*(Poly &a, Poly &b) {
        int n = a.size() + b.size() - 1;
        vcp c(norm(n));
        fp(i, 0, a.size() - 1) c[i].x = a[i];
        fp(i, 0, b.size() - 1) c[i].y = b[i];
        DFT(c), dot(c, c), IDFT(c), a.resize(n);
        fp(i, 0, n - 1) a[i] = int(c[i].y * .5 + .5);
        return a;
    }
}
/*--------------------------------------------------------------------------*/
using namespace Polynomial;


signed main(){
    int n, m, P; cin>>n>>m>>P;
    Poly A(n+1), B(m+1);
    rep(i,0,n) read(A[i]);
    rep(i,0,m) read(B[i]);
    A=MTT::conv(A, B, P);
    for(auto i: A) cout<<i<<' ';

    return 0;
}
```

## FWT

https://www.luogu.com.cn/problem/P4717

给定长度为 $2^n$ 两个序列 $A, B$，设

$C_i = \sum_{j \oplus k = i} A_j \times B_k$

分别当 $\oplus$ 是 `or`,`and`,`xor` 时求出 $C$

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

#define int long long

const int N=(1<<17)+50, mod=998244353, inv2=499122177;

int A[N], B[N], a[N], b[N];
int n;

int add(int a, int b){
    return ((a+b)%mod+mod)%mod;
}

int mul(int a, int b){
    return a*b%mod;
}

void copy(){
```

```
    memcpy(a, A, sizeof a);
    memcpy(b, B, sizeof b);
}

void OR(int *w, int inv){
    for(int o=2, k=1; o<=n; o<<=1, k<<=1)
        for(int i=0; i<n; i+=o)
            for(int j=0; j<k; j++)
                w[i+j+k]=add(w[i+j+k], mul(w[i+j], inv));
}

void AND(int *w, int inv){
    for(int o=2, k=1; o<=n; o<<=1, k<<=1)
        for(int i=0; i<n; i+=o)
            for(int j=0; j<k; j++)
                w[i+j]=add(w[i+j], mul(w[i+j+k], inv));
}

void XOR(int *w, int inv){
    for(int o=2, k=1; o<=n; o<<=1, k<<=1)
        for(int i=0; i<n; i+=o)
            for(int j=0; j<k; j++)
                w[i+j]=add(w[i+j], w[i+j+k]),
                w[i+j+k]=add(w[i+j]-w[i+j+k], -w[i+j+k]),
                w[i+j]=mul(w[i+j], inv), w[i+j+k]=mul(w[i+j+k], inv);
}

void dot(int *a, int *b){
    for(int i=0; i<n; i++) a[i]=a[i]*b[i]%mod;
}

void out(){
    for(int i=0; i<n; i++) cout<<a[i]<<' ';
    cout<<endl;
}

signed main(){
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    cin>>n;
    n=1<<n;

    for(int i=0; i<n; i++) cin>>A[i];
    for(int i=0; i<n; i++) cin>>B[i];

    copy();
    OR(a, 1);
    OR(b, 1);
    dot(a, b);
    OR(a, mod-1);
    out();

    copy();
    AND(a, 1);
    AND(b, 1);
    dot(a, b);
    AND(a, mod-1);
    out();
```

```
    copy();
    XOR(a, 1);
    XOR(b, 1);
    dot(a, b);
    XOR(a, inv2);
    out();

    return 0;
}
```

## 子集和/超集和

```
void sos(){
    for(int i=0;i<(1<<N);i++)
        f[i]=w[i];
    for(int i=0;i<N;i++)
        for(int st=0;st<(1<<N);st++)
            if(st&(1<<i)) f[st]+=f[st^(1<<i)];
}
```

```
rep(j,0,k-1) dwn(st,U,0) if(st>>j&1) f[st^(1<<j)]=min(f[st]+f[st^(1<<j)], 1LL);
// 超集和
```