ACM/ICPC Template Manaual

QUST

hxk

September 4, 2018

Contents

0 Include

```
//#include <bits/stdc++.h>
#include <algorithm>
#include  <iostream>
#include   <cstring>
#include    <string>
#include    <cstdio>
#include    <vector>
#include     <stack>
#include     <queue>
#include     <cmath>
#include       <set>
#include       <map>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define per(i,a,b) for(int i=a;i>=b;i--)
#define clr(a,x) memset(a,x,sizeof(a))
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef unsigned long long ull;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
/************head******************/
int work(){

    return 0;
}
int main(){
#ifdef superkunn
    freopen("input.txt","rt",stdin);
#endif
    work();
    return 0;
}
```

1 Math

1.1 Prime

1.1.1 Eratosthenes Sieve

```
const int MAXN=1e5+5;
int prime[MAXN];//1 base
bool is_prime[MAXN];
int sieve(int n){
    int cnt=0;
    rep(i,0,n)is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    rep(i,2,n){
        if(is_prime[i]){
            prime[++cnt]=i;
            for(int j=i*2;j<=n;j+=i)is_prime[j]=false;
        }
    }
    return cnt;
}
```

1.1.2 Eular Sieve

$O(n)$ $phi[]$ $prime[]$ tot

n

```
const int maxn = "Edit";
bool vis[maxn];
int tot, phi[maxn], prime[maxn];
void CalPhi(int n)
{
    clr(vis, 0);
    phi[1] = 1;
    tot = 0;
    for (int i = 2; i < n; i++)
    {
        if (!vis[i])
            prime[tot++] = i, phi[i] = i - 1;
        for (int j = 0; j < tot; j++)
        {
            if (i * prime[j] > n) break;
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0)
            {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
            else
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
}
```

1.1.3 Prime Factorization

$fact[i][0]^{fact[i][1]}$ i

```
ll fact[100][2];
int getFactors(ll x)
{
    int cnt = 0;
    for (int i = 0; prime[i] <= x / prime[i]; i++)
    {
        fact[cnt][1] = 0;
        if (x % prime[i] == 0)
        {
            fact[cnt][0] = prime[i];
            while (x % prime[i] == 0) fact[cnt][1]++, x /= prime[i];
            cnt++;
        }
    }
    if (x != 1) fact[cnt][0] = x, fact[cnt++][1] = 1;
    return cnt;
}
```

1.1.4 Miller Rabin

```
//using Fast Power
bool Miller_Rabin(ll n, int s){//s is testing frequency . true -> n is prime
    if (n == 2) return 1;
    if (n < 2 || !(n & 1)) return 0;
    int t = 0;
    ll  x, y, u = n - 1;
    while ((u & 1) == 0) t++, u >>= 1;
    for (int i = 0; i < s; i++){
        ll a = rand() % (n - 1) + 1;
        ll x = pow_mod(a, u, n);
        for (int j = 0; j < t; j++){
            ll y = mul_mod(x, x, n);
            if (y == 1 && x != 1 && x != n - 1) return 0;
            x = y;
        }
        if (x != 1) return 0;
    }
    return 1;
}
```

1.1.5 Segment Sieve

```
const int MAXN=1e6+5;
//[a,b)
bool is_prime[MAXN];
bool is_prime_small[MAXN];
ll prime[MAXN];//1 base
int segment_sieve(ll a,ll b){
    int cnt=0;
    for(int i=0;1LL*i*i<b;i++)is_prime_small[i]=true;
    is_prime_small[0]=is_prime_small[1]=false;
    for(int i=0;i<b-a;i++)is_prime[i]=true;
    if(a==1)is_prime[0]=false;

    for(int i=2;1LL*i*i<b;i++){
        if(is_prime_small[i]){
            for(int j=2*i;1LL*j*j<b;j+=i)is_prime_small[j]=false;//[2,sqrt(b))
            for(ll j=max(2LL,(a+i-1)/i)*i;j<b;j+=i)is_prime[j-a]=false;
        }
    }
    //[a,b) [0,b-a)
    for(ll i=0;i<b-a;i++){
        if(is_prime[i])prime[++cnt]=i+a;
    }
    return cnt;
}
```

1.2 Eular phi

1.2.1 Eular

```
ll Euler(ll n)
{
    ll rt = n;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0)
        {
            rt -= rt / i;
            while (n % i == 0) n /= i;
        }
    if (n > 1) rt -= rt / n;
    return rt;
}
```

1.2.2 Sieve

```
const int N = "Edit";
int phi[N] = {0, 1};
void CalEuler()
{
    for (int i = 2; i < N; i++)
        if (!phi[i])
            for (int j = i; j < N; j += i)
            {
                if (!phi[j]) phi[j] = j;
                phi[j] = phi[j] / i * (i - 1);
            }
}
```

1.3 Basic Number Theory

1.3.1 Extended Euclidean

```
__gcd(a,b);

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

//ax+by=1 gcd(a,b)=1
// ll x,y;
//exgcd(3,5,x,y);
//x=2,y=-1;
ll exgcd(ll a, ll b, ll &x, ll &y){
    ll d = a;
    if (b){
        d = exgcd(b, a % b, y, x);
        y -= x * (a / b);
    } else{
        x=1;y=0;
    }
    return d;
}
```

1.3.2 ax+by=c

: $X = x + k * dx, Y = y – k * dy$

x , 0

```
#define Mod(a, b) (((a) % (b) + (b)) % (b))
bool solve(ll a, ll b, ll c, ll& x, ll& y, ll& dx, ll& dy)
{
    if (a == 0 && b == 0) return 0;
    ll x0, y0;
    ll d = exgcd(a, b, x0, y0);
    if (c % d != 0) return 0;
    dx = b / d, dy = a / d;
    x = Mod(x0 * c / d, dx);
    y = (c - a * x) / b;
    //  y = Mod(y0 * c / d, dy); x = (c - b * y) / a;
    return 1;
}
```

1.3.3 Multiplicative Inverse Modulo

$gcd(a, m) == 1.$

```
ll inv(ll a, ll m){
    ll x, y;
    ll d = exgcd(a, m, x, y);
    return d == 1 ? (x + m) % m : -1;
}
```

$a < p$ and p is prime

```
ll inv(ll a, ll p) { return Pow(a, p - 2, p); }
```

```
for (int i = 2; i < n; i++) inv[i] = inv[p % i] * (p - p / i) % p;
```

1.4 Modulo Linear Equation

1.4.1 Chinese Remainder Theory

$X = r_i(modm_i);$ m_i

$X = re + k * mo$

```
void crt(ll r[], ll m[], ll n, ll &re, ll &mo)
{
    mo = 1, re = 0;
    for (int i = 0; i < n; i++) mo *= m[i];
    for (int i = 0; i < n; i++)
    {
        ll x, y,  tm = mo / m[i];
        ll d = exgcd(tm, m[i], x, y);
        re = (re + tm * x * r[i]) % mo;
    }
    re = (re + mo) % mo;
}
```

1.4.2 ExCRT

$X = r_i(mod m_i); m_i$

$X = re + k * mo;$

```
bool excrt(ll r[], ll m[], ll n, ll &re, ll &mo)
{
    ll x, y;
    mo = m[0], re = r[0];
    for (int i = 1; i < n; i++)
    {
        ll d = exgcd(mo, m[i],  x, y);
        if ((r[i] - re) % d != 0) return 0;
        x = (r[i] - re) / d * x % (m[i] / d);
        re += x * mo;
        mo = mo / d * m[i];
        re %= mo;
    }
    re = (re + mo) % mo;
    return 1;
}
```

1.5 Combinatorics

1.5.1 Combination

$0 \leq m \leq n \leq 1000$

```
const int maxn = 1010;
ll C[maxn][maxn];
void CalComb()
{
    C[0][0] = 1;
    for (int i = 1; i < maxn; i++)
    {
        C[i][0] = 1;
        for (int j = 1; j <= i; j++) C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % mod;
    }
}
```

$0 \leq m \leq n \leq 10^5$, p

```
const int maxn = 100010;
ll f[maxn];
ll inv[maxn]; //
void CalFact()
{
    f[0] = 1;
    for (int i = 1; i < maxn; i++) f[i] = (f[i - 1] * i) % p;
    inv[maxn - 1] = Pow(f[maxn - 1], p - 2, p);
    for (int i = maxn - 2; ~i; i--) inv[i] = inv[i + 1] * (i + 1) % p;
}
ll C(int n, int m) { return f[n] * inv[m] % p * inv[n - m] % p; }
```

1.5.2 Lucas

$1 \leq n, m \leq 1000000000, 1 < p < 100000$, p

```
const int maxp = 100010;
ll f[maxn];
ll inv[maxn]; //
void CalFact()
{
    f[0] = 1;
    for (int i = 1; i < maxn; i++) f[i] = (f[i - 1] * i) % p;
    inv[maxn - 1] = Pow(f[maxn - 1], p - 2, p);
    for (int i = maxn - 2; ~i; i--) inv[i] = inv[i + 1] * (i + 1) % p;
}
ll Lucas(ll n, ll m, ll p)
{
    ll ret = 1;
    while (n && m)
    {
        ll a = n % p, b = m % p;
        if (a < b) return 0;
        ret = ret * f[a] % p * inv[b] % p * inv[a - b] % p;
        n /= p, m /= p;
    }
    return ret;
}
```

1.5.3 Big Combination

$0 \leq n \leq 10^9, 0 \leq m \leq 10^4, 1 \leq k \leq 10^9 + 7$

```
vector<int> v;
int dp[110];
ll Cal(int l, int r, int k, int dis)
{
    ll res = 1;
    for (int i = l; i <= r; i++)
    {
        int t = i;
        for (int j = 0; j < v.size(); j++)
        {
            int y = v[j];
            while (t % y == 0) dp[j] += dis, t /= y;
        }
        res = res * (ll)t % k;
    }
    return res;
}
ll Comb(int n, int m, int k)
{
    clr(dp, 0);
    v.clear();
    int tmp = k;
    for (int i = 2; i * i <= tmp; i++)
        if (tmp % i == 0)
        {
            int num = 0;
            while (tmp % i == 0) tmp /= i, num++;
            v.pb(i);
        }
    if (tmp != 1) v.pb(tmp);
    ll ans = Cal(n - m + 1, n, k, 1);
    for (int j = 0; j < v.size(); j++) ans = ans * Pow(v[j], dp[j], k) % k;
    ans = ans * inv(Cal(2, m, k, -1), k) % k;
    return ans;
}
```

1.5.4 Polya

: n , i $gcd(i, n)$

$N * N$,$c^{n^2} + 2c^{\frac{n^2+3}{4}} + c^{\frac{n^2+1}{2}} + 2c^{n\frac{n+1}{2}} + 2c^{\frac{n(n+1)}{2}}$

,$\frac{m^8+17m^4+6m^2}{24}$,$\frac{m^4+11m^2}{12}$

```
//  n  c
ll solve(int c, int n)
{
    if (n == 0) return 0;
    ll ans = 0;
    for (int i = 1; i <= n; i++) ans += Pow(c, __gcd(i, n));
    if (n & 1) ans += n * Pow(c, n + 1 >> 1);
    else ans += n / 2 * (1 + c) * Pow(c, n >> 1);
    return ans / n / 2;
}
```

1.6 Fast Power

```
//hdu 5187
ll mul_mod(ll a,ll b,ll mod){
    ll res=0;
    for(;b;b>>=1){
        if(b&1)res=(res+a)%mod;
        a=(a<<1)%mod;
    }
    return res;
}
ll pow_mod(ll a,ll b,ll mod) {
    ll res=1;
    for(;b;b>>=1){
        if(b&1)res=mul_mod(res,a,mod);
        a=mul_mod(a,a,mod);
    }
    return res;
}
/*****************/
ll pow_mod(ll a,ll b,ll mod) {
    ll res=1;
    for(;b;b>>=1){
        if(b&1)res=res*a%mod;
        a=a*a%mod;
    }
    return res;
}
```

1.7 Mobius Inversion

1.7.1 Mobius

$F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) F(\frac{n}{d})$
$F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu(\frac{d}{n}) F(d)$

```
ll ans;
const int maxn = "Edit";
int n, x, prime[maxn], tot, mu[maxn];
bool check[maxn];
void calmu()
{
    mu[1] = 1;
    for (int i = 2; i < maxn; i++)
    {
        if (!check[i]) prime[tot++] = i, mu[i] = -1;
        for (int j = 0; j < tot; j++)
        {
            if (i * prime[j] >= maxn) break;
            check[i * prime[j]] = true;
            if (i % prime[j] == 0)
            {
                mu[i * prime[j]] = 0;
                break;
            }
            else mu[i * prime[j]] = -mu[i];
        }
    }
}
```

1.7.2 Number of Coprime-pair

n $(n \leq 100000)$, n

```
ll solve()
{
    int b[100005];
    ll _max, ans = 0;
    clr(b, 0);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &x);
        if (x > _max) _max = x;
        b[x]++;
    }
    for (int i = 1; i <= _max; i++)
    {
        int cnt = 0;
        for (ll j = i; j <= _max; j += i) cnt += b[j];
        ans += 1LL * mu[i] * cnt * cnt;
    }
    return (ans - b[1]) / 2;
}
```

1.7.3 VisibleTrees

$gcd(x,y)=1$, $x \le n, y \le m$

```cpp
ll solve(int n, int m)
{
    if (n < m) swap(n, m);
    ll ans = 0;
    for (int i = 1; i <= m; ++i) ans += (ll)mu[i] * (n / i) * (m / i);
    return ans;
}
```

1.8 Fast Transformation

1.8.1 FFT

```cpp
const double PI = acos(-1.0);
//
struct Complex
{
    double x, y; //    x+yi
    Complex(double _x = 0.0, double _y = 0.0) { x = _x, y = _y; }
    Complex operator-(const Complex& b) const { return Complex(x - b.x, y - b.y); }
    Complex operator+(const Complex& b) const { return Complex(x + b.x, y + b.y); }
    Complex operator*(const Complex& b) const { return Complex(x * b.x - y * b.y, x * b.y + y * b.x); }
};
/*
*  FFT IFFT
*  i  (i     )
* len  2
*/
void change(Complex y[], int len)
{
    for (int i = 1, j = len / 2; i < len - 1; i++)
    {
        if (i < j) swap(y[i], y[j]);
        //      ,i<j
        //i  +1,j   +1,  i j
        int k = len / 2;
        while (j >= k) j -= k, k /= 2;
        if (j < k) j += k;
    }
}
/*
*  FFT
* len  2^k ,
* on==1 DFT,on==-1 IDFT
*/
void fft(Complex y[], int len, int on)
{
    change(y, len);
    for (int h = 2; h <= len; h <<= 1)
    {
        Complex wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h));
        for (int j = 0; j < len; j += h)
        {
            Complex w(1, 0);
            for (int k = j; k < j + h / 2; k++)
            {
                Complex u = y[k];
                Complex t = w * y[k + h / 2];
                y[k] = u + t, y[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
    if (on == -1)
        for (int i = 0; i < len; i++) y[i].x /= len;
}
```

1.8.2 NTT

P ,G P $G^{\frac{P-1}{n}}$ $w_n = e^{\frac{2i\pi}{n}}$ P G 1.11

```
const int mod = 119 << 23 | 1;
const int G = 3;
int wn[20];
void getwn()
{ //
    for (int i = 0; i < 20; i++) wn[i] = Pow(G, (mod - 1) / (1 << i), mod);
}
void change(int y[], int len)
{
    for (int i = 1, j = len / 2; i < len - 1; i++)
    {
        if (i < j) swap(y[i], y[j]);
        int k = len / 2;
        while (j >= k) j -= k, k /= 2;
        if (j < k) j += k;
    }
}
void ntt(int y[], int len, int on)
{
    change(y, len);
    for (int h = 2, id = 1; h <= len; h <<= 1, id++)
    {
        for (int j = 0; j < len; j += h)
        {
            int w = 1;
            for (int k = j; k < j + h / 2; k++)
            {
                int u = y[k] % mod;
                int t = 1LL * w * (y[k + h / 2] % mod) % mod;
                y[k] = (u + t) % mod, y[k + h / 2] = ((u - t) % mod + mod) % mod;
                w = 1LL * w * wn[id] % mod;
            }
        }
    }
    if (on == -1)
    {
        //
        int inv = Pow(len, mod - 2, mod);
        for (int i = 1; i < len / 2; i++) swap(y[i], y[len - i]);
        for (int i = 0; i < len; i++) y[i] = 1LL * y[i] * inv % mod;
    }
}
```

1.8.3 FWT

```
void fwt(int f[], int m)
{
    int n = __builtin_ctz(m);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            if (j & (1 << i))
            {
                int l = f[j ^ (1 << i)], r = f[j];
                f[j ^ (1 << i)] = l + r, f[j] = l - r;
                // or: f[j] += f[j ^ (1 << i)];
                // and: f[j ^ (1 << i)] += f[j];
            }
}
void ifwt(int f[], int m)
{
    int n = __builtin_ctz(m);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            if (j & (1 << i))
            {
                int l = f[j ^ (1 << i)], r = f[j];
                f[j ^ (1 << i)] = (l + r) / 2, f[j] = (l - r) / 2;
                //
                // or: f[j] -= f[j ^ (1 << i)];
                // and: f[j ^ (1 << i)] -= f[j];
            }
}
```

1.9 Numerical Integration

1.9.1 Adaptive Simpson's Rule

$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a)+4f(\frac{a+b}{2})+f(b)]$
$|S(a,c)+S(c,b)-S(a,b)|/15<\epsilon$

```
double F(double x) {}
double simpson(double a, double b)
{ //  Simpson
    double c = a + (b - a) / 2;
    return (F(a) + 4 * F(c) + F(b)) * (b - a) / 6;
}
double asr(double a, double b, double eps, double A)
{ // Simpson (  )    [a,b]  Simpson A
    double c = a + (b - a) / 2;
    double L = simpson(a, c), R = simpson(c, b);
    if (fabs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15.0;
    return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
}
double asr(double a, double b, double eps) { return asr(a, b, eps, simpson(a, b)); }
```

1.9.2 Berlekamp-Massey

```
const int N = 1 << 14;
ll res[N], base[N], _c[N], _md[N];
vector<int> Md;
void mul(ll* a, ll* b, int k)
```

```
{
    for (int i = 0; i < k + k; i++) _c[i] = 0;
    for (int i = 0; i < k; i++)
        if (a[i])
            for (int j = 0; j < k; j++) _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
    for (int i = k + k - 1; i >= k; i--)
        if (_c[i])
            for (int j = 0; j < Md.size(); j++) _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] * _md[Md[j]]) % mod;
    for (int i = 0; i < k; i++) a[i] = _c[i];
}
int solve(ll n, VI a, VI b)
{
    ll ans = 0, pnt = 0;
    int k = a.size();
    assert(a.size() == b.size());
    for (int i = 0; i < k; i++) _md[k - 1 - i] = -a[i];
    _md[k] = 1;
    Md.clear();
    for (int i = 0; i < k; i++)
        if (_md[i] != 0) Md.push_back(i);
    for (int i = 0; i < k; i++) res[i] = base[i] = 0;
    res[0] = 1;
    while ((1LL << pnt) <= n) pnt++;
    for (int p = pnt; p >= 0; p--)
    {
        mul(res, res, k);
        if ((n >> p) & 1)
        {
            for (int i = k - 1; i >= 0; i--) res[i + 1] = res[i];
            res[0] = 0;
            for (int j = 0; j < Md.size(); j++) res[Md[j]] = (res[Md[j]] - res[k] * _md[Md[j]]) % mod;
        }
    }
    for (int i = 0; i < k; i++) ans = (ans + res[i] * b[i]) % mod;
    if (ans < 0) ans += mod;
    return ans;
}
VI BM(VI s)
{
    VI C(1, 1), B(1, 1);
    int L = 0, m = 1, b = 1;
    for (int n = 0; n < s.size(); n++)
    {
        ll d = 0;
        for (int i = 0; i <= L; i++) d = (d + (ll)C[i] * s[n - i]) % mod;
        if (d == 0)
            ++m;
        else if (2 * L <= n)
        {
            VI T = C;
            ll c = mod - d * Pow(b, mod - 2) % mod;
            while (C.size() < B.size() + m) C.pb(0);
            for (int i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % mod;
            L = n + 1 - L, B = T, b = d, m = 1;
        }
        else
        {
            ll c = mod - d * Pow(b, mod - 2) % mod;
            while (C.size() < B.size() + m) C.pb(0);
            for (int i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % mod;
            ++m;
        }
    }
    return C;
}
int gao(VI a, ll n)
{
    VI c = BM(a);
    c.erase(c.begin());
    for (int i = 0; i < c.size(); i++) c[i] = (mod - c[i]) % mod;
    return solve(n, c, VI(a.begin(), a.begin() + c.size()));
}
```

1.10 Others

n , , m

```
int josephus(int n, int m)
{
    int r = 0;
    for (int k = 1; k <= n; ++k) r = (r + m) % k;
    return r + 1;
}
```

n^n

```
int leftmost(int n)
{
    double m = n * log10((double)n);
    double g = m - (ll)m;
    return (int)pow(10.0, g);
}
```

$n!$

```
int count(ll n)
{
    if (n == 1) return 1;
    return (int)ceil(0.5 * log10(2 * M_PI * n) + n * log10(n) - n * log10(M_E));
}
```

1.11 Formula

1. : $n = \prod_{i=1}^{k} p_i^{a_i}$,
   - (a) $f(n) = \prod_{i=1}^{k} (a_i + 1)$
   - (b) $g(n) = \prod_{i=1}^{k} (\sum_{j=0}^{a_i} p_i^j)$
2. n $n\varphi(n)/2$
3. $gcd(n,i) = 1,\ gcd(n, n-i) = 1 (1 \le i \le n)$
4. :$D(n) = (n-1)(D(n-2) + D(n-1)) = \sum_{i=2}^{n} \frac{(-1)^k n!}{k!} = [\frac{n!}{e} + 0.5]$
5. :$p\ is\ prime\ \Rightarrow (p-1)! \equiv -1(mod\ p)$
6. :$gcd(a,n) = 1 \Rightarrow a^{\varphi(n)} \equiv 1(mod\ n)$
7. :$gcd(n,p) = 1 \Rightarrow a^n \equiv a^{n\%\varphi(p)}(mod\ p)$

8. : n $\pi(n), \lim\limits_{n\to\infty} \pi(n) = \frac{n}{\ln n}$

9. : x $N = log10(n) + 1$

10. $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$

11. $a > 1, m, n > 0, \quad gcd(a^m - 1, a^n - 1) = a^{gcd(m,n)} - 1$

12. $a > b, gcd(a, b) = 1, \quad gcd(a^m - b^m, a^n - b^n) = a^{gcd(m,n)} - b^{gcd(m,n)}$

$$G = gcd(C_n^1, C_n^2, ..., C_n^{n-1}) = \begin{cases} n, & n \text{ is prime} \\ 1, & n \text{ has multy prime factors} \\ p, & n \text{ has single prime factor } p \end{cases}$$

$gcd(Fib(m), Fib(n)) = Fib(gcd(m, n))$

13. $gcd(m, n) = 1,$:

    (a) $m * n - m - n$

    (b) $N = \frac{(m-1)(n-1)}{2}$

14. $(n + 1)lcm(C_n^0, C_n^1, ..., C_n^{n-1}, C_n^n) = lcm(1, 2, ..., n + 1)$

15. p , $(x + y + ... + w)^p \equiv x^p + y^p + ... + w^p (mod\ p)$

16. :1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012

    $h(0) = h(1) = 1, h(n) = \frac{(4n-2)h(n-1)}{n+1} = \frac{C_{2n}^n}{n+1} = C_{2n}^n - C_{2n}^{n-1}$

17. :$B_n = -\frac{1}{n+1}\sum_{i=0}^{n-1} C_{n+1}^i B_i$

$$\sum_{i=1}^{n} i^k = \frac{1}{k+1}\sum_{i=1}^{k+1} C_{k+1}^i B_{k+1-i}(n+1)^i$$

18. FFT

$r\ 2^k+1$	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
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
998244353	119	23	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1231453023109121	35	45	3
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

2 String Processing

```
//hihocoder 1014
const int maxnode=2600000+10;
const int sigma_size=26;
struct Trie{
    int ch[maxnode][sigma_size];
    int val[maxnode];
    int sz;
    void init(){sz=0;clr(ch[0],0);}
    int idx(char c){return c-'a';}
    void insert(char *s){
        int u=0,n=strlen(s);
        rep(i,0,n-1){
            int x=idx(s[i]);
            if(!ch[u][x]){
                ++sz;
                clr(ch[sz],0);
                val[sz]=0;
                ch[u][x]=sz;
            }
            u=ch[u][x];
            val[u]++;
        }
    }
    int query(char *s){
        int u=0,n=strlen(s),res=0;
        rep(i,0,n-1){
            int x=idx(s[i]);
            if(!ch[u][x])break;
            u=ch[u][x];
            if(i==n-1)res=val[u];
        }
        return res;
    }
}trie;
char s[30];
int work(){
    trie.init();
    int n,m;
    scanf("%d",&n);
    while(n--){
        scanf("%s",s);
        trie.insert(s);
    }
    scanf("%d",&m);
    while(m--){
        scanf("%s",s);
        printf("%d\n",trie.query(s));
    }
    return 0;
}
```

2.1 KMP

```
//MAXN
int nxt[MAXN];
void initkmp(char x[],int m){
    int i=0,j=nxt[0]=-1;
    while(i<m){
        while(j!=-1&&x[i]!=x[j])j=nxt[j];
        nxt[++i]=++j;
    }
}
//x:pa y:tx
int kmp(char x[],int m,char y[],int n){
    int i,j,ans;
    i=j=ans=0;
    initkmp(x,m);
    while(i<n){
        while(j!=-1&&y[i]!=x[j])j=nxt[j];
        i++,j++;
        if(j>=m){
            ans++;
            j=nxt[j];
            //pos:i-m
        }
    }
    return ans;
}
```

2.2 ExtendKMP

```
//next[i]:x[i...m-1] x[0...m-1]
//extend[i]:y[i...n-1] x[0...m-1]
const int N = "Edit";
int next[N], extend[N];
void pre_ekmp(char x[], int m)
{
    next[0] = m;
    int j = 0;
    while (j + 1 < m && x[j] == x[j + 1]) j++;
    next[1] = j;
    int k = 1;
    for (int i = 2; i < m; i++)
    {
        int p = next[k] + k - 1;
        int L = next[i - k];
        if (i + L < p + 1)
            next[i] = L;
        else
        {
            j = max(0, p - i + 1);
            while (i + j < m && x[i + j] == x[j]) j++;
            next[i] = j;
            k = i;
        }
    }
}
void ekmp(char x[], int m, char y[], int n)
{
    pre_ekmp(x, m, next);
    int j = 0;
    while (j < n && j < m && x[j] == y[j]) j++;
    extend[0] = j;
    int k = 0;
    for (int i = 1; i < n; i++)
    {
        int p = extend[k] + k - 1;
        int L = next[i - k];
        if (i + L < p + 1)
            extend[i] = L;
        else
        {
            j = max(0, p - i + 1);
            while (i + j < n && j < m && y[i + j] == x[j]) j++;
            extend[i] = j, k = i;
        }
    }
}
```

2.3 Manacher

```
//hihocoder 1032
const int MAXN=2e6+10;//more than 2 times !
char s[MAXN],str[MAXN];
int len1,len2,p[MAXN];
void init(){
    str[0]='$';
    str[1]='#';
    rep(i,0,len1){
        str[i*2+2]=s[i];
        str[i*2+3]='#';
    }
    len2=len1*2+2;
    str[len2]='*';
}
int manacher(){
    int id=0,mx=0,ans=0;
    rep(i,1,len2-1){
        if(mx>i)p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while(str[i+p[i]]==str[i-p[i]])p[i]++;
        if(i+p[i]>mx){
            mx=i+p[i];
            id=i;
        }
        ans=max(ans,p[i]);
    }
    return ans-1;
}
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%s",s);
        len1=strlen(s);
        init();
        printf("%d\n",manacher());
    }
    return 0;
}
```

2.4 Aho-Corasick Automaton

```
const int maxn = "Edit";
struct Trie
{
    int ch[maxn][26], f[maxn], val[maxn];
    int sz, rt;
    int newnode() { clr(ch[sz], -1), val[sz] = 0; return sz++; }
    void init() { sz = 0, rt = newnode(); }
    inline int idx(char c) { return c - 'A'; };
    void insert(const char* s)
    {
        int u = 0, n = strlen(s);
        for (int i = 0; i < n; i++)
        {
            int c = idx(s[i]);
            if (ch[u][c] == -1) ch[u][c] = newnode();
            u = ch[u][c];
        }
        val[u]++;
    }
    void build()
    {
        queue<int> q;
        f[rt] = rt;
        for (int c = 0; c < 26; c++)
        {
            if (~ch[rt][c])
                f[ch[rt][c]] = rt, q.push(ch[rt][c]);
            else
                ch[rt][c] = rt;
        }
        while (!q.empty())
        {
            int u = q.front();
            q.pop();
            // val[u] |= val[f[u]];
            for (int c = 0; c < 26; c++)
            {
                if (~ch[u][c])
                    f[ch[u][c]] = ch[f[u]][c], q.push(ch[u][c]);
                else
                    ch[u][c] = ch[f[u]][c];
            }
        }
    }
    //
    int query(const char* s)
    {
        int u = rt, n = strlen(s);
        int res = 0;
        for (int i = 0; i < n; i++)
        {
            int c = idx(s[i]);
            u = ch[u][c];
            int tmp = u;
            while (tmp != rt)
            {
                res += val[tmp];
                val[tmp] = 0;
                tmp = f[tmp];
            }
        }
        return res;
    }
};
```

2.5 Suffix Array

```
//     , O(nlogn)
const int maxn = "Edit";
char s[maxn];
int sa[maxn], t[maxn], t2[maxn], c[maxn], rank[maxn], height[maxn];
//n    ,  0~m-1
void build_sa(int m, int n)
{
    n++;
    int *x = t, *y = t2;
    //
    for (int i = 0; i < m; i++) c[i] = 0;
    for (int i = 0; i < n; i++) c[x[i] = s[i]]++;
    for (int i = 1; i < m; i++) c[i] += c[i - 1];
    for (int i = n - 1; ~i; i--) sa[--c[x[i]]] = i;
    for (int k = 1; k <= n; k <<= 1)
    {
        //  sa
        int p = 0;
        for (int i = n - k; i < n; i++) y[p++] = i;
        for (int i = 0; i < n; i++)
            if (sa[i] >= k) y[p++] = sa[i] - k;
        //
        for (int i = 0; i < m; i++) c[i] = 0;
        for (int i = 0; i < n; i++) c[x[y[i]]]++;
        for (int i = 0; i < m; i++) c[i] += c[i - 1];
        for (int i = n - 1; ~i; i--) sa[--c[x[y[i]]]] = y[i];
        // sa y   x
        swap(x, y);
        p = 1;
        x[sa[0]] = 0;
        for (int i = 1; i < n; i++)
            x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k] ? p - 1 : p++;
        if (p >= n) break; //    ,sa   ,
        m = p;             //
    }
    n--;
    int k = 0;
    for (int i = 0; i <= n; i++) rank[sa[i]] = i;
    for (int i = 0; i < n; i++)
    {
        if (k) k--;
        int j = sa[rank[i] - 1];
        while (s[i + k] == s[j + k]) k++;
        height[rank[i]] = k;
    }
}

```

```
int dp[maxn][30];
void initrmq(int n)
{
    for (int i = 1; i <= n; i++)
        dp[i][0] = height[i];
    for (int j = 1; (1 << j) <= n; j++)
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
            dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
}
int rmq(int l, int r)
{
    int k = 31 - __builtin_clz(r - l + 1);
    return min(dp[l][k], dp[r - (1 << k) + 1][k]);
}
int lcp(int a, int b)
{ //
    a = rank[a], b = rank[b];
    if (a > b) swap(a, b);
    return rmq(a + 1, b);
}
```

2.6 Suffix Automation

```
const int maxn = "Edit";
struct SAM
{
    int len[maxn << 1], link[maxn << 1], ch[maxn << 1][26];
    int sz, rt, last;
    int newnode(int x = 0)
    {
        len[sz] = x;
        link[sz] = -1;
        clr(ch[sz], -1);
        return sz++;
    }
    void init() { sz = last = 0, rt = newnode(); }
    void extend(int c)
    {
        int np = newnode(len[last] + 1);
        int p;
        for (p = last; ~p && ch[p][c] == -1; p = link[p]) ch[p][c] = np;
        if (p == -1)
            link[np] = rt;
        else
        {
            int q = ch[p][c];
            if (len[p] + 1 == len[q])
                link[np] = q;
            else
            {
                int nq = newnode(len[p] + 1);
                memcpy(ch[nq], ch[q], sizeof(ch[q]));
                link[nq] = link[q], link[q] = link[np] = nq;
                for (; ~p && ch[p][c] == q; p = link[p]) ch[p][c] = nq;
            }
        }
        last = np;
    }
    int topcnt[maxn], topsam[maxn << 1];
    void sort()
    { //
        clr(topcnt, 0);
        for (int i = 0; i < sz; i++) topcnt[len[i]]++;
        for (int i = 0; i < maxn - 1; i++) topcnt[i + 1] += topcnt[i];
        for (int i = 0; i < sz; i++) topsam[--topcnt[len[i]]] = i;
    }
};
```

2.7 HashString

```
const ll B1=1e7+7;
const ll B2=1e9+7;
char pa[10004];
char tx[1000006];
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%s%s",pa,tx);
        int pl=strlen(pa);
        int tl=strlen(tx);
        ll w=1;
        rep(i,1,pl)w=(w*B1)%B2;
        ll ph=0,th=0;
        rep(i,0,pl-1){
            ph=(ph*B1+pa[i])%B2;
            th=(th*B1+tx[i])%B2;
        }
        int ans=0;
        for(int i=0;i+pl<=tl;i++){
            if(ph==th)ans++;
            if(i+pl<tl)th=(th*B1+tx[i+pl]-tx[i]*w)%B2;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

3 Data Structure

3.1 other

```
//hdu 1394
const int MAXN=5005;
int n;
vi A;
int x[MAXN];
int merging(vi &a){
    int n=SZ(a);
    if(n<=1)return 0;
    int cnt=0;
    vi b(a.begin(),a.begin()+n/2);
    vi c(a.begin()+n/2,a.end());
    cnt+=merging(b);
    cnt+=merging(c);
    int ai=0,bi=0,ci=0;
    while(ai<n){
        if(bi<SZ(b)&&(ci==SZ(c)||b[bi]<=c[ci])){
            a[ai++]=b[bi++];
        }else{
            cnt+=n/2-bi;
            a[ai++]=c[ci++];
        }
    }
    return cnt;
}
int work(){
    while(~scanf("%d",&n)){
        A.clear();
        rep(i,1,n)scanf("%d",&x[i]),A.pb(x[i]);
        int sum=merging(A);
        int res=sum;
        rep(i,1,n){
            sum=sum-x[i]+(n-1-x[i]);
            res=min(res,sum);
        }
        printf("%d\n",res);
    }
    return 0;
}
```

3.1.1 QuickSelect

```
anytype QuickSelect(anytype arr[],int l,int r,int k){
    int i=l,j=r,mid=arr[(i+j)>>1];
    while(i<=j){
        while(arr[i]<mid)i++;
        while(arr[j]>mid)j--;
        if(i<=j){
            swap(arr[i],arr[j]);
            i++;
            j--;
        }
    }
    if(l<j&&k<=j)return QuickSelect(arr,l,j,k);
    if(i<r&&k>=i)return QuickSelect(arr,i,r,k);
    return arr[k];
}
```

3.2 Binary Indexed Tree

```
//add(pos,a) sum(r)-sum(l-1)
//add(l,a) add(r+1,-a) sum(pos)
const int MAXN=100000;
struct BIT{
    int n,c[MAXN<<1];
    void init(int _n){
        n=_n;
        rep(i,0,n)c[i]=0;
    }
    void update(int i,int v){
        for(;i<=n;i+=i&-i)c[i]+=v;
    }
    int query(int i){
        int s=0;
        for(;i;i-=i&-i)s+=c[i];
        return s;
    }
    int findpos(int v){
        int sum=0;
        int pos=0;
        int i=1;
        for(;i<n;i<<=1);
        for(;i;i>>=1){
            if(pos+i<=n&&sum+c[pos+i]<v){
                sum+=c[pos+i];
                pos+=i;
            }
        }
        return pos+1;
    }
}bit;
```

3.2.1 poj3468

$a_i = \sum_{i=1}^{x} d_i$

$\sum_{i=1}^{x} a_i = \sum_{i=1}^{x} \sum_{j=1}^{i} d_j = \sum_{i=1}^{x} (x - i + 1) d_i$

$\sum_{i=1}^{x} a_i = (x + 1) \sum_{i=1}^{x} d_i - \sum_{i=1}^{x} d_i \times i$

```
const int MAXN=1e5+5;
int n,q,x,y,z;
long long c1[MAXN],c2[MAXN];
void add(int x,int y){
    for(int i=x;i<=n;i+=i&(-i))c1[i]+=y,c2[i]+=1LL*x*y;
}
ll sum(int x){
    ll ans(0);
    for(int i=x;i;i-=i&(-i))ans+=1LL*(x+1)*c1[i]-c2[i];
    return ans;
}
char op[5];
int work(){
    scanf("%d%d",&n,&q);
    int a1,a2;
    a1=0;
    rep(i,1,n){
        scanf("%d",&a2);
        add(i,a2-a1);
        a1=a2;
    }
    while(q--){
        scanf("%s",op);
        if(op[0]=='Q'){
            scanf("%d%d%d",&x,&y,&z);
            printf("%lld\n",sum(y)-sum(x-1));
        }else{
            scanf("%d%d%d",&x,&y,&z);
            add(x,z);
            add(y+1,-z);
        }
    }
    return 0;
}
```

3.3 Segment Tree

```
#define lson rt<<1
#define rson rt<<1|1
#define le l,m,lson
#define ri m+1,r,rson
#define mid m=(l+r)>>1
```

3.3.1 Single-point Update

```
const int MAXN=5e4+5;
int sum[MAXN<<2];
void push_up(int rt){
    sum[rt]=sum[lson]+sum[rson];
}
void build(int l,int r,int rt){
    if(l==r){
        scanf("%d",&sum[rt]);
        return;
    }
    int mid;
    build(le);
    build(ri);
    push_up(rt);
}
void update(int p,int v,int l,int r,int rt){
    if(l==r){
        sum[rt]+=v;
        return;
    }
    int mid;
    if(p<=m)update(p,v,le);
    else update(p,v,ri);
    push_up(rt);
}
```

```
int query(int L,int R,int l,int r,int rt){
    if(L<=l&&r<=R){
        return sum[rt];
    }
    int mid;
    int ret=0;
    if(L<=m)ret+=query(L,R,le);
    if(R>m)ret+=query(L,R,ri);
    return ret;
}
```

3.3.2 Interval Update

```
const int MAXN=1e5+5;
ll lazy[MAXN<<2];
ll tree[MAXN<<2];
void push_up(int rt){
    tree[rt]=tree[lson]+tree[rson];
}
void push_down(int rt,int m){
    ll w=lazy[rt];
    if(w){
        lazy[lson]+=w;
        lazy[rson]+=w;
        tree[lson]+=w*(m-(m>>1));
        tree[rson]+=w*(m>>1);
        lazy[rt]=0;
    }
}
void build(int l,int r,int rt){
    lazy[rt]=0;
    if(l==r){
        scanf("%lld",&tree[rt]);
        return;
    }
    int mid;
    build(le);
    build(ri);
    push_up(rt);
}
void update(int L,int R,int v,int l,int r,int rt){
    if(L<=l&&r<=R){
        lazy[rt]+=v;
        tree[rt]+=1ll*v*(r-l+1);
        return;
    }
    push_down(rt,r-l+1);
    int mid;
    if(L<=m)update(L,R,v,le);
    if(R>m)update(L,R,v,ri);
    push_up(rt);
}
ll query(int L,int R,int l,int r,int rt){
    if(L<=l&&r<=R){
        return tree[rt];
    }
    push_down(rt,r-l+1);
    int mid;
    ll ret=0;
    if(L<=m)ret+=query(L,R,le);
    if(R>m)ret+=query(L,R,ri);
    return ret;
}
```

3.4 Splay Tree

```
#define key_value ch[ch[rt][1]][0]
const int MAXN=1e5;
struct Splay{
    int a[MAXN];//0 base
    int sz[MAXN],ch[MAXN][2],fa[MAXN];
    int key[MAXN],rev[MAXN];
    int rt,tot;
    int stk[MAXN],top;
    void push_up(int x){
        sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+1;
    }
    void push_down(int x){
        if(rev[x]){
            swap(ch[x][0],ch[x][1]);
            if(ch[x][0])rev[ch[x][0]]^=1;
            if(ch[x][1])rev[ch[x][1]]^=1;
            rev[x]=0;
        }
    }
    int newnode(int p=0,int k=0){
        int x=top?stk[top--]:++tot;
        fa[x]=p;
        sz[x]=1;
        ch[x][0]=ch[x][1]=0;
        key[x]=k;
        rev[x]=0;
        return x;
    }
    int build(int l,int r,int p){
        if(l>r)return 0;
        int mid=(l+r)>>1;
        int x=newnode(p,a[mid]);
        ch[x][0]=build(l,mid-1,x);
        ch[x][1]=build(mid+1,r,x);
        push_up(x);
        return x;
    }
    void init(int n){
        tot=0,top=0;
        rt=newnode(0,-1);
        ch[rt][1]=newnode(rt,-1);
        rep(i,0,n-1)a[i]=i+1;
        key_value=build(0,n-1,ch[rt][1]);
        push_up(ch[rt][1]);
        push_up(rt);
    }
    void rotate(int x,int d){
        int y=fa[x];
        push_down(y);
        push_down(x);
        ch[y][d^1]=ch[x][d];
        fa[ch[x][d]]=y;
        if(fa[y])ch[fa[y]][ch[fa[y]][1]==y]=x;
        fa[x]=fa[y];
        ch[x][d]=y;
        fa[y]=x;
        push_up(y);
    }
    void splay(int x,int goal=0){
        push_down(x);
        while(fa[x]!=goal){
            if(fa[fa[x]]==goal){
                rotate(x,ch[fa[x]][0]==x);
            }else{
                int y=fa[x];
                int d=ch[fa[y]][0]==y;
                ch[y][d]==x?rotate(x,d^1):rotate(y,d);
                rotate(x,d);
            }
        }
        push_up(x);
        if(goal==0)rt=x;
    }
    int kth(int r,int k){
        push_down(r);
        int t=sz[ch[r][0]]+1;
        if(t==k)return r;
        return t>k?kth(ch[r][0],k):kth(ch[r][1],k-t);
    }
    void select(int l,int r){
        splay(kth(rt,1),0);
        splay(kth(ch[rt][1],r-l+2),rt);
    }
};
```

3.5 Functional Segment Tree

```
//poj 2104
const int MAXN=1e5+6;
int n,m,cnt,x,y,k,root[MAXN],a[MAXN];
struct node{int l,r,sum;}T[MAXN*40];
vi v;
int getid(int x){return lower_bound(all(v),x)-v.begin()+1;}
void update(int l,int r,int &x,int y,int pos){
    x=++cnt;
    T[x]=T[y];
    T[x].sum++;
    if(l==r)return;
    int mid=(l+r)>>1;
    if(mid>=pos)update(l,mid,T[x].l,T[y].l,pos);
    else update(mid+1,r,T[x].r,T[y].r,pos);
}
int query(int l,int r,int x,int y,int k){
    if(l==r)return l;
    int sum=T[T[y].l].sum-T[T[x].l].sum;
    int mid=(l+r)>>1;
    if(sum>=k)return query(l,mid,T[x].l,T[y].l,k);
    else return query(mid+1,r,T[x].r,T[y].r,k-sum);
```

```cpp
}
int work(){
    scanf("%d%d",&n,&m);
    v.clear();
    rep(i,1,n)scanf("%d",&a[i]),v.pb(a[i]);
    sort(all(v)),v.erase(unique(all(v)),v.end());
    cnt=0;
    rep(i,1,n)update(1,n,root[i],root[i-1],getid(a[i]));
    rep(i,1,m)scanf("%d%d%d",&x,&y,&k),printf("%d\n",v[query(1,n,root[x-1],root[y],k)-1]);
    return 0;
}
```

3.6 Sparse Table

```cpp
const int MAXN = "Edit";
int mmax[MAXN][30], mmin[MAXN][30];
int a[MAXN], n, k;
void init(){
    for (int i = 1; i <= n; i++) mmax[i][0] = mmin[i][0] = a[i];
    for (int j = 1; (1 << j) <= n; j++)
        for (int i = 1; i + (1 << j) - 1 <= n; i++){
            mmax[i][j] = max(mmax[i][j - 1], mmax[i + (1 << (j - 1))][j - 1]);
            mmin[i][j] = min(mmin[i][j - 1], mmin[i + (1 << (j - 1))][j - 1]);
        }
}
// op=0/1 return [l,r] max/min
int rmq(int l, int r, int op){
    int k = 31 - __builtin_clz(r - l + 1);
    if (op == 0)
        return max(mmax[l][k], mmax[r - (1 << k) + 1][k]);
    return min(mmin[l][k], mmin[r - (1 << k) + 1][k]);
}
```

2D

```cpp
void init(){
    for (int i = 0; (1 << i) <= n; i++){
        for (int j = 0; (1 << j) <= m; j++){
            if (i == 0 && j == 0) continue;
            for (int row = 1; row + (1 << i) - 1 <= n; row++)
                for (int col = 1; col + (1 << j) - 1 <= m; col++)
                    if (i){
                        dp[row][col][i][j] = max(dp[row][col][i - 1][j];
                        dp[row + (1 << (i - 1))][col][i - 1][j]);
                    }else{
                        dp[row][col][i][j] = max(dp[row][col][i][j - 1];
                        dp[row][col + (1 << (j - 1))][i][j - 1]);
                    }
        }
    }
}
int rmq(int x1, int y1, int x2, int y2){
    int kx = 31 - __builtin_clz(x2 - x1 + 1);
    int ky = 31 - __builtin_clz(y2 - y1 + 1);
    int m1 = dp[x1][y1][kx][ky];
    int m2 = dp[x2 - (1 << kx) + 1][y1][kx][ky];
    int m3 = dp[x1][y2 - (1 << ky) + 1][kx][ky];
    int m4 = dp[x2 - (1 << kx) + 1][y2 - (1 << ky) + 1][kx][ky];
    return max(max(m1, m2), max(m3, m4));
}
```

3.7 Heavy-Light Decomposition

```
const int maxn = "Edit";
struct HLD
{
    int n, dfs_clock;
    int sz[maxn], top[maxn], son[maxn], dep[maxn], fa[maxn], id[maxn];
    vector<int> G[maxn];
    void init(int n)
    {
        this->n = n, clr(son, -1), dfs_clock = 0;
        for (int i = 0; i < n; i++) G[i].clear();
    }
    void add_edge(int u, int v) { G[u].pb(v), G[v].pb(u); }
    void dfs(int u, int p, int d)
    {
        dep[u] = d, fa[u] = p, sz[u] = 1;
        for (auto& v : G[u])
        {
            if (v == p) continue;
            dfs(v, u, d + 1);
            sz[u] += sz[v];
            if (son[u] == -1 || sz[v] > sz[son[u]]) son[u] = v;
        }
    }
    void link(int u, int t)
    {
        top[u] = t, id[u] = ++dfs_clock;
        if (son[u] == -1) return;
        link(son[u], t);
        for (auto& v : G[u])
            if (v != son[u] && v != fa[u]) link(v, v);
    }
    //      ,
    int query_path(int u, int v)
    {
        int ret = 0;
        while (top[u] != top[v])
        {
            if (dep[top[u]] < dep[top[v]]) swap(u, v);
            ret += query(id[top[u]], id[u]);
            u = fa[top[u]];
        }
        if (dep[u] > dep[v]) swap(u, v);
        ret += query(id[u], id[v]);
    }
};
```

3.8 Link-Cut Tree

```
const int maxn = "Edit";
struct LCT
```

```
{
    int val[maxn], sum[maxn]; //
    int rev[maxn], ch[maxn][2], fa[maxn];
    int stk[maxn];
    inline void init(int n)
    { //
        for (int i = 1; i <= n; i++) scanf("%d", val + i);
    }
    inline bool isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }
    inline bool get(int x) { return ch[fa[x]][1] == x; }
    void pushdown(int x)
    {
        if (!rev[x]) return;
        swap(ch[x][0], ch[x][1]);
        if (ch[x][0]) rev[ch[x][0]] ^= 1;
        if (ch[x][1]) rev[ch[x][1]] ^= 1;
        rev[x] ^= 1;
    }
    void pushup(int x) { sum[x] = val[x] + sum[ch[x][0]] + sum[ch[x][1]]; }
    void rotate(int x)
    {
        int y = fa[x], z = fa[fa[x]], d = get(x);
        if (!isroot(y)) ch[z][get(y)] = x;
        fa[x] = z;
        ch[y][d] = ch[x][d ^ 1], fa[ch[y][d]] = y;
        ch[x][d ^ 1] = y, fa[y] = x;
        pushup(y), pushup(x);
    }
    void splay(int x)
    {
        int top = 0;
        stk[++top] = x;
        for (int i = x; !isroot(i); i = fa[i]) stk[++top] = fa[i];
        for (int i = top; i; i--) pushdown(stk[i]);
        for (int f; !isroot(x); rotate(x))
            if (!isroot(f = fa[x])) rotate(get(x) == get(f) ? f : x);
    }
    void access(int x)
    {
        for (int y = 0; x; y = x, x = fa[x]) splay(x), ch[x][1] = y, pushup(x);
    }
    int find(int x) { access(x), splay(x); while (ch[x][0]) x = ch[x][0]; return x; }
    void makeroot(int x) { access(x), splay(x), rev[x] ^= 1; }
    void link(int x, int y) { makeroot(x), fa[x] = y, splay(x); }
    void cut(int x, int y) { makeroot(x), access(y), splay(y), fa[x] = ch[y][0] = 0; }
    void update(int x, int v) { val[x] = v, access(x), splay(x); }
    int query(int x, int y) { makeroot(y), access(x), splay(x); return sum[x]; }
};
```

4 Graph Theory

4.1 Union-Find Set

```
const int MAXN=1e6+5;
struct DSU{
    int p[MAXN];
    void init(int n){rep(i,0,n)p[i]=i;}
    int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
    void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
    bool same(int x,int y){return findp(x)==findp(y);}
}dsu;
```

4.2 Minimal Spanning Tree

4.2.1 Kruskal

```
//poj 1258
#include<cstdio>
#include<algorithm>
using namespace std;
const int MAXE=1e5+5;
const int MAXN=1e5+5;
struct DSU{
    int p[MAXN];
    void init(int n){for(int i=0;i<=n;i++)p[i]=i;}
    int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
    void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
    bool same(int x,int y){return findp(x)==findp(y);}
}dsu;
struct edge{int u,v,cost;}es[MAXE];
bool cmp(const edge &x,const edge &y){return x.cost<y.cost;}
int V,E;
int kruskal(){
    sort(es,es+E,cmp);
    dsu.init(V);
    int res=0;
    for(int i=0;i<E;i++){
        if(!dsu.same(es[i].u,es[i].v)){
            dsu.unite(es[i].u,es[i].v);
            res+=es[i].cost;
        }
    }
    return res;
}
int main(){
    while(~scanf("%d",&V)){
        E=0;
        for(int i=1;i<=V;i++){
            for(int j=1;j<=V;j++){
                int w;
                scanf("%d",&w);
                if(i==j)continue;
                es[E].u=i;
                es[E].v=j;
                es[E].cost=w;
                E++;
            }
        }
        printf("%d\n",kruskal());
    }
    return 0;
}
```

4.3 Shortest Path

4.3.1 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define clr(a,x) memset(a,x,sizeof(a))
#define mp make_pair
const int MAXV=2e6;
const int MAXE=5e6+10;
typedef long long anytype;
typedef pair<anytype,int> P;
int tot=0;
int head[MAXV];
struct Edge{
    int v,c,nxt;
    Edge(){}
    Edge(int v,int c,int nxt):v(v),c(c),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int c){
    edge[tot]=Edge(v,c,head[u]);
    head[u]=tot++;
}
anytype d[MAXV];
void dij(int s){
    priority_queue<P,vector<P>,greater<P> > que;
    clr(d,-1);
    d[s]=0;
    que.push(P(0,s));
    while(!que.empty()){
        P t=que.top();
        que.pop();
        int v=t.second;
        if(d[v]!=-1&&d[v]<t.first)continue;
        for(int i=head[v];~i;i=edge[i].nxt){
            Edge e=edge[i];
            if(d[e.v]==-1||d[e.v]>d[v]+e.c){
                d[e.v]=d[v]+e.c;
                que.push(mp(d[e.v],e.v));
            }
        }
    }
}
int main(){
    int T;
    scanf("%d",&T);
    while(T--){
        int n,m,k;
        scanf("%d%d%d",&n,&m,&k);
        init();
        rep(i,1,m){
            int u,v,c;
            scanf("%d%d%d",&u,&v,&c);
            rep(j,0,k){
                add_edge(u+j*n,v+j*n,c);
                if(j!=k)add_edge(u+j*n,v+(j+1)*n,0);
            }
        }
        dij(1);
        printf("%lld\n",d[n+k*n]);
    }
    return 0;
}
```

4.3.2 Spfa

```
//hdu3592
const int MAXN=1e3+5;
const int MAXE=3e4+5;
const int INF=0x3f3f3f3f;
int N,X,Y;
int tot;
int head[MAXN];
struct Edge{
    int v,w,nxt;
    Edge(){}
    Edge(int v,int w,int nxt):v(v),w(w),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int w){
    edge[tot]=Edge(v,w,head[u]);
    head[u]=tot++;
}
queue<int> que;
bool inq[MAXN];
int qtime[MAXN];
int d[MAXN];
int spfa(){
    while(!que.empty())que.pop();
    clr(qtime,0);
    clr(inq,0);
    rep(i,1,N)d[i]=INF;
    d[1]=0;
    que.push(1);
    inq[1]=1;
    qtime[1]++;
    while(!que.empty()){
        int u=que.front();
        que.pop();
        inq[u]=0;
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            int w=edge[i].w;
            if(d[v]>d[u]+w){
                d[v]=d[u]+w;
                if(!inq[v]){
                    que.push(v);
                    inq[v]=1;
                    qtime[v]++;
                    if(qtime[v]>N)return -1;
                }
            }
        }
    }
    if(d[N]==INF)return -2;
    else return d[N];
}
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d%d%d",&N,&X,&Y);
        init();
        rep(i,1,N-1){
            add_edge(i+1,i,0);
        }
        while(X--){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            add_edge(x,y,z);
        }
        while(Y--){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            add_edge(y,x,-z);
        }
        printf("%d\n",spfa());
    }
    return 0;
}
```

4.4 Topo Sort

```
//cf 915D
const int MAXN=505;
const int MAXM=1e5+5;
int n,m;
int tot;
int head[MAXN],cur[MAXN],idec[MAXN];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXM];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int que[MAXN];
int st,ed;
bool topsort(int x){
    int nst=1,ned=0;
    rep(i,1,n)cur[i]=idec[i];
    cur[x]--;
    que[++ned]=x;
    while(nst<=ned){
        int u=que[nst++];
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(--cur[v]==0)que[++ned]=v;
        }
    }
    if(ned+ed==n)return true;
    else return false;
}
int work(){
    scanf("%d%d",&n,&m);
    init();
    while(m--){
        int u,v;
        scanf("%d%d",&u,&v);
        add_edge(u,v);
        idec[v]++;
    }
    st=1,ed=0;
    rep(i,1,n){
        if(idec[i]==0)que[++ed]=i;
    }
    while(st<=ed){
        int u=que[st++];
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(--idec[v]==0)que[++ed]=v;
        }
    }
    if(ed==n){
        puts("YES");
        return 0;
    }
    rep(i,1,n){
        if(idec[i]==1){
            if(topsort(i)){
                puts("YES");
                return 0;
            }
        }
    }
    puts("NO");
    return 0;
}
```

4.5 LCA

4.5.1 Tarjan

Tarjan

$O(n+q)$

```
const int maxn = "Edit";
int par[maxn];            //
int ans[maxn];            //
vector<int> G[maxn];      //
vector<PII> query[maxn]; //
bool vis[maxn];           //
inline void init(int n)
{
    for (int i = 1; i <= n; i++)
    {
        G[i].clear(), query[i].clear();
        par[i] = i, vis[i] = 0;
    }
}
inline void add_edge(int u, int v) { G[u].pb(v); }
inline void add_query(int id, int u, int v)
{
    query[u].pb(mp(v, id));
    query[v].pb(mp(u, id));
}
void tarjan(int u)
{
    vis[u] = 1;
    for (auto& v : G[u])
    {
        if (vis[v]) continue;
        tarjan(v);
        unite(u, v);
    }
    for (auto& q : query[u])
    {
        int &v = q.X, &id = q.Y;
        if (!vis[v]) continue;
        ans[id] = find(v);
    }
}
```

4.5.2 LCArmq

```
#include<bits/stdc++.h>
#define MAXV 100005
#define MAXLOGV 32
using namespace std;
int N,M,Q;
int st[MAXLOGV][MAXV];
vector<int> G[MAXV];
int root;
int vs[MAXV*2];
int depth[MAXV*2];
int id[MAXV];
void dfs(int v,int p,int d,int &k){
    id[v]=k;
    vs[k]=v;
    depth[k++]=d;
    for(int i=0;i<G[v].size();i++){
        if(G[v][i]!=p){
            dfs(G[v][i],v,d+1,k);
            vs[k]=v;
            depth[k++]=d;
        }
    }
}
int getMin(int x, int y){
    return depth[x]<depth[y]?x:y;
}

void rmq_init(int n){
    for(int i=0;i<n;++i) st[0][i]=i;
    for(int i=1;1<<i<n;++i)
        for(int j=0;j+(1<<i)-1<n;++j)
            st[i][j]=getMin(st[i-1][j],st[i-1][j+(1<<(i-1))]);
}
void init(int V){
    int k=0;
    dfs(root,-1,0,k);
    rmq_init(V*2-1);
}
int query(int l, int r){
    int k=31-__builtin_clz(r-l+1);
    return getMin(st[k][l],st[k][r-(1<<k)+1]);
}
int lca(int u,int v){
    if(u==v) return u;
    return vs[query(min(id[u],id[v]),max(id[u],id[v]))];
}
int dis(int u,int v){
    return depth[id[u]]+depth[id[v]]-2*depth[id[lca(u,v)]];
}
int main()
{
    scanf("%d%d",&N,&M);
    for(int i=0;i<M;i++){
        int x,y;
        scanf("%d%d",&x,&y);
        G[x].push_back(y);
        G[y].push_back(x);
    }
    root=0;
    init(N);
    scanf("%d",&Q);
    while(Q--){
        int x,y;
        scanf("%d%d",&x,&y);
        printf("%d\n",lca(x,y));
    }
    return 0;
}
```

4.6 Depth-First Traversal

```
vector<int> G[MAXN];
int vis[MAXN];
void dfs(int u){
    vis[u]=1;
    PREVISIT(u);
    for(auto v:G[u]){
        if(!vis[v])dfs(v);
    }
    POSTVISIT(u);
}
```

4.6.1 Biconnected-Component

```
//UVALive - 3523
#include<bits/stdc++.h>
using namespace std;
#define clr(a,x) memset(a,x,sizeof(a))
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef pair<int,int> pii;
typedef vector<int> vi;
const int MAXV=1e3+10;
const int MAXE=1e6+10;
int tot;
int head[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int pre[MAXV],is_cut[MAXV],bccno[MAXV],dfs_clock,bcc_cnt;
vi bcc[MAXV];
stack<pii > st;
int dfs(int u,int fa){
    int lowu=pre[u]=++dfs_clock;
    int child=0;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        pii e=mp(u,v);
        if(!pre[v]){
            st.push(e);
            child++;
            int lowv=dfs(v,u);
            lowu=min(lowu,lowv);
            if(lowv>=pre[u]){
                is_cut[u]=1;
                bcc_cnt++;
                bcc[bcc_cnt].clear();
                for(;;){
                    pii x=st.top();
                    st.pop();
                    if(bccno[x.fi]!=bcc_cnt){
                        bcc[bcc_cnt].pb(x.fi);
                        bccno[x.fi]=bcc_cnt;
                    }
                    if(bccno[x.se]!=bcc_cnt){
                        bcc[bcc_cnt].pb(x.se);
                        bccno[x.se]=bcc_cnt;
                    }
                    if(x.fi==u&&x.se==v)break;
                }
            }
        }else if(pre[v]<pre[u]&&v!=fa){
            st.push(e);
            lowu=min(lowu,pre[v]);
        }
    }
    if(fa<0&&child==1)is_cut[u]=0;
    return lowu;
}
void find_bcc(int n){
    clr(pre,0);
    clr(is_cut,0);
    clr(bccno,0);
    dfs_clock=bcc_cnt=0;
    rep(i,1,n){
        if(!pre[i])dfs(i,-1);
    }
}
int odd[MAXV],color[MAXV];
bool bipartite(int u,int b){
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(bccno[v]!=b)continue;
        if(color[v]==color[u])return false;
        if(!color[v]){
            color[v]=3-color[u];
            if(!bipartite(v,b))return false;
        }
    }
    return true;
}
bool mmp[MAXV][MAXV];
int main(){
    int n,m;
    while(scanf("%d%d",&n,&m),n+m){
        clr(mmp,0);
        rep(i,1,m){
            int x,y;
            scanf("%d%d",&x,&y);
            mmp[x][y]=1;
            mmp[y][x]=1;
        }
        init();
        rep(i,1,n){
            rep(j,i+1,n){
                if(!mmp[i][j]){
                    add_edge(i,j);
                    add_edge(j,i);
                }
            }
        }
        find_bcc(n);
        clr(odd,0);
        for(int i=1;i<=bcc_cnt;i++){
            clr(color,0);
            for(int j=0;j<bcc[i].size();j++){
                bccno[bcc[i][j]]=i;
            }
            int u=bcc[i][0];
            color[u]=1;
            if(!bipartite(u,i)){
                for(int j=0;j<bcc[i].size();j++){
                    odd[bcc[i][j]]=1;
                }
            }
        }
        int ans=n;
        rep(i,1,n)if(odd[i])ans--;
        printf("%d\n",ans);
    }
    return 0;
}
```

4.6.2 Strongly Connected Component

```
const int MAXV=1e4+10;
const int MAXE=1e5+10;
int tot,head[MAXV];
int low[MAXV],dfn[MAXV],stk[MAXV],Belong[MAXV];
int idx,top,scc;
bool instk[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
void Tarjan(int u){
    int v;
    low[u]=dfn[u]=++idx;
    stk[top++]=u;
    instk[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        v=edge[i].v;
        if(!dfn[v]){
            Tarjan(v);
            if(low[u]>low[v])low[u]=low[v];
        }else if(instk[v]&&low[u]>dfn[v])low[u]=dfn[v];
    }
    if(low[u]==dfn[u]){
        scc++;
        do{
            v=stk[--top];
            instk[v]=false;
            Belong[v]=scc;
        }while(v!=u);
    }
}
void tscc(int N){
    clr(dfn,0);
    clr(instk,0);
    idx=scc=top=0;
    rep(i,1,N)if(!dfn[i])Tarjan(i);
}
```

4.6.3 Kosaraju

```
const int MAXV=2e4+10;
const int MAXE=5e4+10;
int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
bool vis[MAXV];
int stk[MAXV],top;
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE],redge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
    clr(rhead,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    redge[tot]=Edge(u,rhead[v]);
    head[u]=rhead[v]=tot++;
}
void dfs(int u){
    vis[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!vis[v])dfs(v);
    }
    stk[++top]=u;
}
void rdfs(int u,int k){
    vis[u]=true;
    Belong[u]=k;
    for(int i=rhead[u];~i;i=redge[i].nxt){
        int v=redge[i].v;
        if(!vis[v])rdfs(v,k);
    }
}
void kscc(int V){
    scc=top=0;
    clr(vis,0);
    rep(i,1,V)if(!vis[i])dfs(i);
    clr(vis,0);
    per(i,top,1){
        int v=stk[i];
        if(!vis[v])rdfs(v,++scc);
    }
}
```

4.6.4 TwoSAT

```
//poj3683
//0 base !
//if (x V (!y))then add_clause(1,x,0,y)
//if  x then add_var(1,x)
const int MAXV=1e5;
const int MAXE=3e6+5;
int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
bool vis[MAXV];
int stk[MAXV],top;
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE],redge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
    clr(rhead,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    redge[tot]=Edge(u,rhead[v]);
    head[u]=rhead[v]=tot++;
}
void dfs(int u){
    vis[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!vis[v])dfs(v);
    }
    stk[++top]=u;
}
void rdfs(int u,int k){
    vis[u]=true;
    Belong[u]=k;
    for(int i=rhead[u];~i;i=redge[i].nxt){
        int v=redge[i].v;
        if(!vis[v])rdfs(v,k);
    }
}
void kscc(int V){
    scc=top=0;
    clr(vis,0);
    rep(i,0,V-1)if(!vis[i])dfs(i);
    clr(vis,0);
    per(i,top,1){
        int v=stk[i];
        if(!vis[v])rdfs(v,++scc);
    }
}
```

```
void add_clause(int xv,int x,int yv,int y){
    x=x<<1|xv;
    y=y<<1|yv;
    add_edge(x^1,y);
    add_edge(y^1,x);
}
void add_var(int xv,int x){
    x=x<<1|xv;
    add_edge(x^1,x);
}
int st[MAXV],ed[MAXV],d[MAXV];
char tm[10];
int fun(){
    int res=0;
    int h=(tm[0]-'0')*10+tm[1]-'0';
    res=h*60;
    res+=(tm[3]-'0')*10+tm[4]-'0';
    return res;
}
int work(){
    int n;
    scanf("%d",&n);
    rep(i,0,n-1){
        scanf("%s",tm);
        st[i]=fun();
        scanf("%s",tm);
        ed[i]=fun();
        scanf("%d",&d[i]);
    }
    init();
    rep(i,0,n-1){
        rep(j,0,i-1){
            if(min(st[i]+d[i],st[j]+d[j])>max(st[i],st[j])){
                add_clause(0,i,0,j);
            }
            if(min(st[i]+d[i],ed[j])>max(st[i],ed[j]-d[j])){
                add_clause(0,i,1,j);
            }
            if(min(ed[i],st[j]+d[j])>max(ed[i]-d[i],st[j])){
                add_clause(1,i,0,j);
            }
            if(min(ed[i],ed[j])>max(ed[i]-d[i],ed[j]-d[j])){
                add_clause(1,i,1,j);
            }
        }
    }
    kscc(2*n);
    rep(i,0,n-1){
        if(Belong[i<<1]==Belong[i<<1|1]){
            puts("NO");
            return 0;
        }
    }
    puts("YES");
    rep(i,0,n-1){
        if(Belong[i<<1|1]>Belong[i<<1]){
            printf("%02d:%02d %02d:%02d\n",st[i]/60,st[i]%60,(st[i]+d[i])/60,(st[i]+d[i
    ])%60);
        }else{
            printf("%02d:%02d %02d:%02d\n",(ed[i]-d[i])/60,(ed[i]-d[i])%60,ed[i]/60,ed[i]%60);
        }
    }
    return 0;
}
```

4.6.5 cut$_vertex$

```
//poj 1144
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define clr(a,x) memset(a,x,sizeof(a))
const int MAXV=105;
const int MAXE=1e5;
int tot;
int head[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int n;
bool is_cut[MAXV];
int low[MAXV],pre[MAXV];
int dfs_clock;
int dfs(int u,int fa){
    int lowu=pre[u]=++dfs_clock;
    int child=0;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!pre[v]){
            child++;
            int lowv=dfs(v,u);
            lowu=min(lowu,lowv);
            if(lowv>=pre[u]){
                is_cut[u]=true;
            }
        }else if(pre[v]<pre[u]&&v!=fa){
            lowu=min(lowu,pre[v]);
        }
    }
    if(fa<0&&child==1)is_cut[u]=false;
    low[u]=lowu;
    return lowu;
}
int main(){
    while(scanf("%d",&n),n){
        init();
        int x;
        while(scanf("%d",&x),x){
            int y;
            while(getchar()!='\n'){
                scanf("%d",&y);
                add_edge(x,y);
                add_edge(y,x);
            }
        }
        clr(is_cut,0);
        clr(low,0);
        clr(pre,0);
        dfs_clock=0;
        int cnt=0;
        dfs(1,-1);
        for(int i=1;i<=n;i++){
            if(is_cut[i])cnt++;
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

4.7 Eular Path

- :
  - : ( )
  - :
  - : (,),
- G
  - G
  - G ( ) 0 2.
- G
  - G
  - G
- G
  - G
  - u 1,v 1, (u ,v)
- G
  - G
  - G

4.7.1 Fleury

,

```
const int maxn = "Edit";
int G[maxn][maxn];
int deg[maxn][maxn];
vector<int> Ans;
inline void init() { clr(G, 0), clr(deg, 0); }
inline void AddEdge(int u, int v) { deg[u]++, deg[v]++, G[u][v]++, G[v][u]++; }
void Fleury(int s)
{
    for (int i = 0; i < n; i++)
        if (G[s][i])
        {
            G[s][i]--, G[i][s]--;
            Fleury(i);
        }
    Ans.pb(s);
}
```

4.8 Bipartite Graph Matching

4.8.1 Hungry

```
//poj3041
const int MAXV=1e3+5;
struct BM{
    int V;
    vi G[MAXV];
    int match[MAXV];
    bool vis[MAXV];
    void init(int x){
        V=x;
        rep(i,1,V)G[i].clear();
    }
    void add_edge(int u,int v){
        G[u].pb(v);
        G[v].pb(u);
    }
    bool dfs(int u){
        vis[u]=true;
        for(int i=0;i<(int)G[u].size();i++){
            int v=G[u][i];
            int w=match[v];
            if(w==-1||(!vis[w]&&dfs(w))){
                match[u]=v;
                match[v]=u;
                return true;
            }
        }
        return false;
    }
    int matching(){
        int ret=0;
        clr(match,-1);
        rep(i,1,V){
            if(match[i]==-1){
                clr(vis,0);
                if(dfs(i))ret++;
            }
        }
        return ret;
    }
}bm;
int work(){
    int n,k;
    scanf("%d%d",&n,&k);
    bm.init(2*n);
    while(k--){
        int u,v;
        scanf("%d%d",&u,&v);
        bm.add_edge(u,n+v);
    }
    printf("%d",bm.matching());
    return 0;
}
```

4.8.2 Hungry(Multiple)

```
const int maxn = "Edit";
const int maxm = "Edit";
int uN, vN;         //u,v  ,
int g[maxn][maxm]; //
int linker[maxm][maxn];
bool used[maxm];
int num[maxm]; //
bool dfs(int u)
{
    for (int v = 0; v < vN; v++)
        if (g[u][v] && !used[v])
        {
            used[v] = true;
            if (linker[v][0] < num[v])
            {
                linker[v][++linker[v][0]] = u;
                return true;
            }
            for (int i = 1; i <= num[0]; i++)
                if (dfs(linker[v][i]))
                {
                    linker[v][i] = u;
                    return true;
                }
        }
    return false;
}
int hungary()
{
    int res = 0;
    for (int i = 0; i < vN; i++) linker[i][0] = 0;
    for (int u = 0; u < uN; u++)
    {
        clr(used, 0);
        if (dfs(u)) res++;
    }
    return res;
}
```

4.8.3 Kuhn-Munkres

```
const int maxn = "Edit";
int nx, ny;                           //
int g[maxn][maxn];                    //
int linker[maxn], lx[maxn], ly[maxn]; //y    ,x,y
int slack[N];
bool visx[N], visy[N];
bool dfs(int x)
{
    visx[x] = true;
    for (int y = 0; y < ny; y++)
    {
        if (visy[y]) continue;
        int tmp = lx[x] + ly[y] - g[x][y];
        if (tmp == 0)
        {
            visy[y] = true;
            if (linker[y] == -1 || dfs(linker[y]))
            {
                linker[y] = x;
                return true;
            }
        }
        else if (slack[y] > tmp)
            slack[y] = tmp;
    }
    return false;
}
int KM()
{
    clr(linker, -1), clr(ly, 0);
    for (int i = 0; i < nx; i++)
    {
        lx[i] = -INF;
        for (int j = 0; j < ny; j++)
            if (g[i][j] > lx[i]) lx[i] = g[i][j];
    }
    for (int x = 0; x < nx; x++)
    {
        clr(slack, 0x3f);
        for (;;)
        {
            clr(visx, 0), clr(visy, 0);
            if (dfs(x)) break;
            int d = INF;
            for (int i = 0; i < ny; i++)
                if (!visy[i] && d > slack[i]) d = slack[i];
            for (int i = 0; i < nx; i++)
                if (visx[i]) lx[i] -= d;
            for (int i = 0; i < ny; i++)
                if (visy[i])
                    ly[i] += d;
                else
                    slack[i] -= d;
        }
    }
    int res = 0;
    for (int i = 0; i < ny; i++)
        if (~linker[i]) res += g[linker[i]][i];
    return res;
}
```

4.9 Network Flow

4.9.1 Dinic

```
//poj 3281
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
#include<queue>
using namespace std;
#define clr(a,x) memset(a,x,sizeof(a))
const int MAXV=400+5;
const int MAXE=1e5+5;
const int INF=0x3f3f3f3f;
int tot;
int head[MAXV],level[MAXV],iter[MAXV];
struct Edge{
    int v,cap,nxt;
    Edge(){}
    Edge(int v,int cap,int nxt):v(v),cap(cap),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int c){
    edge[tot]=Edge(v,c,head[u]);
    head[u]=tot++;
    edge[tot]=Edge(u,0,head[v]);
    head[v]=tot++;
}
void bfs(int s){
    clr(level,-1);
    level[s]=0;
    queue<int> que;
    que.push(s);
    while(!que.empty()){
        int u=que.front();
        que.pop();
        for(int i=head[u];~i;i=edge[i].nxt){
            int v=edge[i].v;
            int c=edge[i].cap;
            if(c>0&&level[v]<0){
                level[v]=level[u]+1;
                que.push(v);
            }
        }
    }
}
int dfs(int u,int t,int f){
    if(u==t)return f;
    for(int &i=iter[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        int c=edge[i].cap;
        if(c>0&&level[u]<level[v]){
            int d=dfs(v,t,min(f,c));
            if(d>0){
                edge[i].cap-=d;
                edge[i^1].cap+=d;
                return d;
            }
        }
    }
    return 0;
}
int max_flow(int s,int t){
    int flow=0;
    while(1){
        bfs(s);
        if(level[t]<0)return flow;
        int f;
        memcpy(iter,head,sizeof(head));
        while(f=dfs(s,t,INF))flow+=f;
    }
}
int main(){
    int n,f,d;
    scanf("%d%d%d",&n,&f,&d);
    int s=0,t=2*n+f+d;
    init();
    for(int i=1;i<=f;i++){
        add_edge(s,2*n+i,1);
    }
    for(int i=1;i<=d;i++){
        add_edge(2*n+f+i,t,1);
    }
    for(int i=1;i<=n;i++){
        add_edge(i,n+i,1);
        int ff,dd;
        scanf("%d%d",&ff,&dd);
        while(ff--){
            int x;
            scanf("%d",&x);
            add_edge(2*n+x,i,1);
        }
        while(dd--){
            int x;
            scanf("%d",&x);
            add_edge(n+i,2*n+f+x,1);
        }
    }
    printf("%d",max_flow(s,t));
    return 0;
}
```

4.9.2 MinCost MaxFlow

```
// poj2135
#include<cstdio>
#include<vector>
#include<algorithm>
#include<queue>
using namespace std;
const int MAXV=1005;
const int MAXE=50000;
const int INF=100000000;
typedef pair<int,int> P;
struct edge{int to,cap,cost,rev;};
int dist[MAXV],h[MAXV],prevv[MAXV],preve[MAXV];
int V;
vector<edge> G[MAXV];
void add_edge(int from,int to,int cap,int cost){
    G[from].push_back((edge){to,cap,cost,G[to].size()});
    G[to].push_back((edge){from,0,-cost,G[from].size()-1});
}
int min_cost_flow(int s,int t,int f){
    int res=0;
    fill(h,h+V,0);
    while(f>0){
        priority_queue<P,vector<P>,greater<P> >que;
        fill(dist,dist+V,INF);
        dist[s]=0;
        que.push(P(0,s));
        while(!que.empty()){
            P p=que.top(); que.pop();
            int v=p.second;
            if(dist[v]<p.first) continue;
            for(int i=0;i<G[v].size();i++){
                edge &e=G[v][i];
                if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]){
                    dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
                    prevv[e.to]=v;
                    preve[e.to]=i;
                    que.push(P(dist[e.to],e.to));
                }
            }
        }
        if(dist[t]==INF){
            return -1;
        }
        for(int v=0;v<V;v++) h[v]+=dist[v];
        int d=f;
        for(int v=t;v!=s;v=prevv[v]){
            d=min(d,G[prevv[v]][preve[v]].cap);
        }
        f-=d;
        res+=d*h[t];
        for(int v=t;v!=s;v=prevv[v]){
            edge &e=G[prevv[v]][preve[v]];
            e.cap-=d;
            G[v][e.rev].cap+=d;
        }
    }
    return res;
}
int main(){
    int N,M;
    scanf("%d%d",&N,&M);
    V=N;
    for(int i=1;i<=M;i++){
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        add_edge(x-1,y-1,1,z);
        add_edge(y-1,x-1,1,z);
    }
    printf("%d",min_cost_flow(0,N-1,2));
    return 0;
}
```

5 Computational Geometry

5.1 Basic Function

```
#define zero(x) ((fabs(x) < eps ? 1 : 0))
#define sgn(x) (fabs(x) < eps ? 0 : ((x) < 0 ? -1 : 1))

struct point
{
    double x, y;
    point(double a = 0, double b = 0) { x = a, y = b; }
    point operator-(const point& b) const { return point(x - b.x, y - b.y); }
    point operator+(const point& b) const { return point(x + b.x, y + b.y); }
    //
    bool operator==(point& b) { return zero(x - b.x) && zero(y - b.y); }
    //  (   )
    double operator*(const point& b) const { return x * b.x + y * b.y; }
    //  (   )
    double operator^(const point& b) const { return x * b.y - y * b.x; }
    // P   a
    point rotate(point b, double a)
    {
        double dx, dy;
        (*this - b).split(dx, dy);
        double tx = dx * cos(a) - dy * sin(a);
        double ty = dx * sin(a) + dy * cos(a);
        return point(tx, ty) + b;
    }
    //     a b
    void split(double& a, double& b) { a = x, b = y; }
};
struct line
{
    point s, e;
    line() {}
    line(point ss, point ee) { s = ss, e = ee; }
};
```

5.2 Position

5.2.1 Point-Point

```
double dist(point a, point b) { return sqrt((a - b) * (a - b)); }
```

5.2.2 Line-Line

```
// <0, *>   ; <1, *>   ; <2, P>   P;
pair<int, point> spoint(line l1, line l2)
{
    point res = l1.s;
    if (sgn((l1.s - l1.e) ^ (l2.s - l2.e)) == 0)
        return mp(sgn((l1.s - l2.e) ^ (l2.s - l2.e)) != 0, res);
    double t = ((l1.s - l2.s) ^ (l2.s - l2.e)) / ((l1.s - l1.e) ^ (l2.s - l2.e));
    res.x += (l1.e.x - l1.s.x) * t;
    res.y += (l1.e.y - l1.s.y) * t;
    return mp(2, res);
}
```

5.2.3 Segment-Segment

```
bool segxseg(line l1, line l2)
{
    return
        max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) &&
        max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&
        max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) &&
        max(l2.s.y, l2.e.y) >= min(l1.s.y, l1.e.y) &&
        sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e-l1.e) ^ (l1.s - l1.e)) <= 0 &&
        sgn((l1.s - l2.e) ^ (l2.s - l2.e)) * sgn((l1.e-l2.e) ^ (l2.s - l2.e)) <= 0;
}
```

5.2.4 Line-Segment

```
//l1  ,l2
bool segxline(line l1, line l2)
{
    return sgn((l2.s - l1.e) ^ (l1.s - l1.e)) * sgn((l2.e - l1.e) ^ (l1.s - l1.e)) <=
    0;
}
```

5.2.5 Point-Line

```
double pointtoline(point p, line l)
{
    point res;
    double t = ((p - l.s) * (l.e - l.s)) / ((l.e - l.s) * (l.e - l.s));
    res.x = l.s.x + (l.e.x - l.s.x) * t, res.y = l.s.y + (l.e.y - l.s.y) * t;
    return dist(p, res);
}
```

5.2.6 Point-Segment

```
double pointtosegment(point p, line l)
{
    point res;
    double t = ((p - l.s) * (l.e - l.s)) / ((l.e - l.s) * (l.e - l.s));
    if (t >= 0 && t <= 1)
        res.x = l.s.x + (l.e.x - l.s.x) * t, res.y = l.s.y + (l.e.y - l.s.y) * t;
    else
        res = dist(p, l.s) < dist(p, l.e) ? l.s : l.e;
    return dist(p, res);
}
```

5.2.7 Point on Segment

```
bool PointOnSeg(point p, line l)
{
    return
        sgn((l.s - p) ^ (l.e-p)) == 0 &&
        sgn((p.x - l.s.x) * (p.x - l.e.x)) <= 0 &&
        sgn((p.y - l.s.y) * (p.y - l.e.y)) <= 0;
}
```

5.3 Polygon

5.3.1 Area

```
double area(point p[], int n)
{
    double res = 0;
    for (int i = 0; i < n; i++) res += (p[i] ^ p[(i + 1) % n]) / 2;
    return fabs(res);
}
```

5.3.2 Point in Convex

```
//    ,     (    <0 >0)
//    : [0,n)
// -1 :
// 0  :
// 1  :
int PointInConvex(point a, point p[], int n)
{
    for (int i = 0; i < n; i++)
        if (sgn((p[i] - a) ^ (p[(i + 1) % n] - a)) < 0)
            return -1;
        else if (PointOnSeg(a, line(p[i], p[(i + 1) % n])))
            return 0;
    return 1;
}
```

5.3.3 Point in Polygon

```
//   ,poly[]    3,  0~n-1
// -1 :
// 0  :
// 1  :
int PointInPoly(point p, point poly[], int n)
{
    int cnt;
    line ray, side;
    cnt = 0;
    ray.s = p;
    ray.e.y = p.y;
    ray.e.x = -100000000000.0; // -INF,
    for (int i = 0; i < n; i++)
    {
        side.s = poly[i], side.e = poly[(i + 1) % n];
        if (PointOnSeg(p, side)) return 0;
        //
        if (sgn(side.s.y - side.e.y) == 0)
            continue;
        if (PointOnSeg(sid e.s, r ay))
            cnt += (sgn(side.s.y - side.e.y) > 0);
        else if (PointOnSeg(side.e, ray))
            cnt += (sgn(side.e.y - side.s.y) > 0);
        else if (segxseg(ray, side))
            cnt++;
    }
    return cnt % 2 == 1 ? 1 : -1;
}
```

5.3.4 Judge Convex

```
//
//  1~n-1
bool isconvex(point poly[], int n)
{
    bool s[3];
    clr(s, 0);
    for (int i = 0; i < n; i++)
    {
        s[sgn((poly[(i + 1) % n] - poly[i]) ^ (poly[(i + 2) % n] - poly[i])) + 1] = 1;
        if (s[0] && s[2]) return 0;
    }
    return 1;
}
```

5.4 Integer Points

5.4.1 On Segment

```
int OnSegment(line l) { return __gcd(fabs(l.s.x - l.e.x), fabs(l.s.y - l.e.y)) + 1; }
```

5.4.2 On Polygon Edge

```
int OnEdge(point p[], int n)
{
    int i, ret = 0;
    for (i = 0; i < n; i++)
        ret += __gcd(fabs(p[i].x - p[(i + 1) % n].x), fabs(p[i].y - p[(i + 1) % n].y));
    return ret;
}
```

5.4.3 Inside Polygon

```
int InSide(point p[], int n)
{
    int i, area = 0;
    for (i = 0; i < n; i++)
        area += p[(i + 1) % n].y * (p[i].x - p[(i + 2) % n].x);
    return (fabs(area) - OnEdge(n, p)) / 2 + 1;
}
```

5.5 Circle

5.5.1 Circumcenter

```
point waixin(point a, point b, point c)
{
    double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1 * a1 + b1 * b1) / 2;
    double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2 * a2 + b2 * b2) / 2;
    double d = a1 * b2 - a2 * b1;
    return point(a.x + (c1 * b2 - c2 * b1) / d, a.y + (a1 * c2 - a2 * c1) / d);
}
```

6 Dynamic Programming

6.1 Subsequence

6.1.1 Max Sum

```
//   a  n,
int MaxSeqSum(int a[], int n)
{
    int rt = 0, cur = 0;
    for (int i = 0; i < n; i++)
        cur += a[i], rt = max(cur, rt), cur = max(0, cur);
    return rt;
}
```

6.1.2 Longest Increase

```
//    1 ,LIS()  ,  lis[]
const int N = "Edit";
int len, a[N], b[N], f[N];
int Find(int p, int l, int r)
{
    while (l <= r)
    {
        int mid = (l + r) >> 1;
        if (a[p] > b[mid])
            l = mid + 1;
        else
            r = mid - 1;
    }
    return f[p] = l;
}
int LIS(int lis[], int n)
{
    int len = 1;
    f[1] = 1, b[1] = a[1];
    for (int i = 2; i <= n; i++)
    {
        if (a[i] > b[len])
            b[++len] = a[i], f[i] = len;
        else
            b[Find(i, 1, len)] = a[i];
    }
    for (int i = n, t = len; i >= 1 && t >= 1; i--)
        if (f[i] == t) lis[--t] = a[i];
    return len;
}

//   ( 0 ,  )
int dp[N];
int LIS(int a[], int n)
{
    clr(dp, 0x3f);
    for (int i = 0; i < n; i++) *lower_bound(dp, dp + n, a[i]) = a[i];
    return lower_bound(dp, dp + n, INF) - dp;
}
```

6.1.3 Longest Common Increase

```
//    1
int LCIS(int a[], int b[], int n, int m)
{
    clr(dp, 0);
    for (int i = 1; i <= n; i++)
    {
        int ma = 0;
        for (int j = 1; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if (a[i] > b[j]) ma = max(ma, dp[i - 1][j]);
            if (a[i] == b[j]) dp[i][j] = ma + 1;
        }
    }
    return *max_element(dp[n] + 1, dp[n] + 1 + m);
}
```

6.2 Digit Statistics

```
int a[20];
ll dp[20][state];
ll dfs(int pos, /*state */, bool lead /*  */, bool limit /*   */)
{
    //  ，  ，0，pos==-1
    if (pos == -1) return 1;
    /*   1，      ，             ，
        pos，         */
    if (!limit && !lead && dp[pos][state] != -1) return dp[pos][state];
    /*          ，      */
    int up = limit ? a[pos] : 9; // limit   up
    ll ans = 0;
    for (int i = 0; i <= up; i++) // ，     ans
    {
        if () ...
        else if () ...
        ans += dfs(pos - 1, /*  */, lead && i == 0, limit && i == a[pos])
        //
        /*     i，
                ，  state   i  */
    }
    //  ，
    if (!limit && !lead) dp[pos][state] = ans;
    /*      ，    ，  ，
          lead，  lead    */
    return ans;
}
ll solve(ll x)
{
    int pos = 0;
    do //
        a[pos++] = x % 10;
    while (x /= 10);
    return dfs(pos - 1 /*    */, /*    */, true, true);
    //         ，      0
}
```

7 Others

7.1 Matrix

7.1.1 Matrix FastPow

```
typedef vector<ll> vec;
typedef vector<vec> mat;
mat mul(mat& A, mat& B)
{
    mat C(A.size(), vec(B[0].size()));
    for (int i = 0; i < A.size(); i++)
        for (int k = 0; k < B.size(); k++)
            if (A[i][k]) //
                for (int j = 0; j < B[0].size(); j++)
                    C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
    return C;
}
mat Pow(mat A, ll n)
{
    mat B(A.size(), vec(A.size()));
    for (int i = 0; i < A.size(); i++) B[i][i] = 1;
    for (; n; n >>= 1, A = mul(A, A))
        if (n & 1) B = mul(B, A);
    return B;
}
```

7.1.2 Gauss Elimination

```
void gauss()
{
    int now = 1, to;
    double t;
    for (int i = 1; i <= n; i++, now++)
    {
        /*for (to = now; !a[to][i] && to <= n; to++);
        //     ,
        if (to != now)
            for (int j = 1; j <= n + 1; j++)
                swap(a[to][j], a[now][j]);*/
        t = a[now][i];
        for (int j = 1; j <= n + 1; j++) a[now][j] /= t;
        for (int j = 1; j <= n; j++)
            if (j != now)
            {
                t = a[j][i];
                for (int k = 1; k <= n + 1; k++) a[j][k] -= t * a[now][k];
            }
    }
}
```

7.2 Tricks

7.2.1 Stack-Overflow

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

7.2.2 Fast-Scanner

```cpp
template <class T>
inline bool scan_d(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}
inline void out(int x){
    if(x<0){
        putchar('-');
        x=-x;
    }
    if (x > 9) out(x / 10);
    putchar(x % 10 + '0');
}
```

7.2.3 Strok-Sscanf

```cpp
// get some integers in a line
gets(buf);
int v;
char *p = strtok(buf, " ");
while (p){
    sscanf(p, "%d", &v);
    p = strtok(NULL," ");
}
```

7.3 Mo Algorithm

```cpp
//cf 671 E
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN=1<<20;
struct node{
    int l,r,id;
}Q[MAXN];
int n,m,k;
int block;
int a[MAXN];
int pre[MAXN];
ll cnt[MAXN];
ll ANS,ans[MAXN];
bool cmp(node x,node y){
    if(x.l/block==y.l/block)return x.r<y.r;
    else return x.l/block<y.l/block;
}
void add(int x){
    ANS+=cnt[pre[x]^k];
    cnt[pre[x]]++;
}
void del(int x){
    cnt[pre[x]]--;
    ANS-=cnt[pre[x]^k];
}
int main(){
    scanf("%d%d%d",&n,&m,&k);
    block=(int)sqrt(n);
    pre[0]=0;
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        pre[i]=a[i]^pre[i-1];
    }
    for(int i=1;i<=m;i++){
        scanf("%d%d",&Q[i].l,&Q[i].r);
        Q[i].id=i;
    }
    sort(Q+1,Q+1+m,cmp);
    ANS=0;
    memset(cnt,0,sizeof(cnt));
    cnt[0]=1;
    int L=1,R=0;
    for(int i=1;i<=m;i++){
        while(L>Q[i].l){L--;add(L-1);};
        while(L<Q[i].l){del(L-1);L++;}
        while(R<Q[i].r){R++;add(R);};
        while(R>Q[i].r){del(R);R--;};
        ans[Q[i].id]=ANS;
    }
    for(int i=1;i<=m;i++){
        printf("%lld\n",ans[i]);
    }
    return 0;
}
```

7.4 BigNum

7.4.1 High-precision

```
java
```

7.5 VIM

```
syntax on
set nu
set tabstop=4
set expandtab
set autoindent
set cin
set mouse=a

map<F2> :call SetTitle()<CR>
func SetTitle()
let l = 0
let l = l + 1 | call setline(l,'#include <algorithm>')
let l = l + 1 | call setline(l,'#include  <iostream>')
let l = l + 1 | call setline(l,'#include   <cstring>')
let l = l + 1 | call setline(l,'#include    <string>')
let l = l + 1 | call setline(l,'#include    <cstdio>')
let l = l + 1 | call setline(l,'#include    <vector>')
let l = l + 1 | call setline(l,'#include     <stack>')
let l = l + 1 | call setline(l,'#include     <queue>')
let l = l + 1 | call setline(l,'#include     <cmath>')
let l = l + 1 | call setline(l,'#include       <set>')
let l = l + 1 | call setline(l,'#include       <map>')
let l = l + 1 | call setline(l,'using namespace std;')
let l = l + 1 | call setline(l,'#define rep(i,a,b) for(int i=a;i<=b;i++)')
let l = l + 1 | call setline(l,'#define per(i,a,b) for(int i=a;i>=b;i--)')
let l = l + 1 | call setline(l,'#define clr(a,x) memset(a,x,sizeof(a))')
let l = l + 1 | call setline(l,'#define pb push_back')
let l = l + 1 | call setline(l,'#define mp make_pair')
let l = l + 1 | call setline(l,'#define all(x) (x).begin(),(x).end()')
let l = l + 1 | call setline(l,'#define fi first')
let l = l + 1 | call setline(l,'#define se second')
let l = l + 1 | call setline(l,'#define SZ(x) ((int)(x).size())')
let l = l + 1 | call setline(l,'typedef unsigned long long ull;')
let l = l + 1 | call setline(l,'typedef long long ll;')
let l = l + 1 | call setline(l,'typedef vector<int> vi;')
let l = l + 1 | call setline(l,'typedef pair<int,int> pii;')
let l = l + 1 | call setline(l,'/************head******************/')
let l = l + 1 | call setline(l,'int work(){')
let l = l + 1 | call setline(l,'')
let l = l + 1 | call setline(l,'    return 0;')
let l = l + 1 | call setline(l,'}')
let l = l + 1 | call setline(l,'int main(){')
let l = l + 1 | call setline(l,'#ifdef superkunn')
let l = l + 1 | call setline(l,'    freopen("input.txt","rt",stdin);')
let l = l + 1 | call setline(l,'#endif')
let l = l + 1 | call setline(l,'    work();')
let l = l + 1 | call setline(l,'    return 0;')
let l = l + 1 | call setline(l,'}')
endfunc
```

7.6 BASH

7.6.1 a.sh

8 Geometry

```
struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};
typedef Point Vector;
Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
Vector operator - (Point A,Point B){return Vector(A.x-B.x,A.y-B.y);}
Vector operator * (Vector A,double p){return Vector(A.x*p,A.y*p);}
Vector operator / (Vector A,double p){return Vector(A.x/p,A.y/p);}
bool operator < (const Point& a,const Point &b){
    return a.x<b.x||(a.x==b.x&&a.y<b.y);
}
const double eps = 1e-10;
int dcmp(double x){
    if(fabs(x)<eps)return 0;else return x<0?-1:1;
}
bool operator == (const Point& a,const Point &b){
    return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
//(x,y)->  atan2(y,x)
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
double Area2(Point A,Point B,Point C){return Cross(B-A,C-A);}
Vector Rotate(Vector A,double rad){
    return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
Vector Normal(Vector A){
    double L=Length(A);
    return Vector(-A.y/L,A.x/L);
}
```