ACM/ICPC Template Manaual

QUST

hxk

August 5, 2018

Contents

0 Include

```
#include <bits/stdc++.h>
using namespace std;
#define clr(a, x) memset(a, x, sizeof(a))
#define mp(x, y) make_pair(x, y)
#define pb(x) push_back(x)
#define X first
#define Y second
#define fastin                    \
    ios_base::sync_with_stdio(0); \
    cin.tie(0);
typedef long long ll;
typedef long double ld;
typedef pair<int, int> PII;
typedef vector<int> VI;
const int INF = 0x3f3f3f3f;
const int mod = 1e9 + 7;
const double eps = 1e-6;

int main()
{
#ifndef ONLINE_JUDGE
    freopen("test.in", "r", stdin);
    freopen("test.out", "w", stdout);
#endif

    return 0;
}
```

1 Math

1.1 Prime

1.1.1 Eratosthenes Sieve

$O(n \log\log n)$ maxn
$notprime[i] = 0/1$ 0 1

```
const int maxn = "Edit";
bool notprime[maxn] = {1, 1};   // 0 && 1
void GetPrime()
{
    for (int i = 2; i < maxn; i++)
        if (!notprime[i] && i <= maxn / i)  //  √n
            for (int j = i * i; j < maxn; j += i)
                notprime[j] = 1;
}
```

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

```cpp
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

$O(s \log n)$ 2^{63} ,s

```cpp
bool Miller_Rabin(ll n, int s)
{
    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(a, u, n);
        for (int j = 0; j < t; j++)
        {
            ll y = Mul(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

$[a, b)$

is_prime[i-a]=true i
$a < b \leq 10^{12}, b - a \leq 10^6$

```cpp
const int maxn = "Edit";
bool is_prime_small[maxn], is_prime[maxn];
int prime[maxn];
int segment_sieve(ll a, ll b)
{
    int tot = 0;
    for (ll i = 0; i * i < b; ++i)
        is_prime_small[i] = true;
    for (ll i = 0; i < b - a; ++i)
        is_prime[i] = true;
    for (ll i = 2; i * i < b; ++i)
        if (is_prime_small[i])
        {
            for (ll j = 2 * i; j * j < b; j += i)
                is_prime_small[j] = false;
            for (ll j = max(2LL, (a + i - 1) / i) * i; j < b; j += i)
                is_prime[j - a] = false;
        }
    for (ll i = 0; i < b - a; ++i)
        if (is_prime[i]) prime[tot++] = i + a;
    return tot;
}
```

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

```
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

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

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

$a < p$ p ,

```
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

$X = r_i(mod m_i);$ 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

$0 \le m \le n \le 1000$

```
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;
    }
}
```

$0 \le m \le n \le 10^5$, 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

```
ll Mul(ll a, ll b, ll mod)
{
    ll t = 0;
    for (; b; b >>= 1, a = (a << 1) % mod)
        if (b & 1) t = (t + a) % mod;
    return t;
}
ll Pow(ll a, ll n, ll mod)
{
    ll t = 1;
    for (; n; n >>= 1, a = (a * a % mod))
        if (n & 1) t = (t * a % mod);
    return t;
}
```

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

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

```
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

```
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);
```

```cpp
    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

```cpp
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

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

$n!$

```cpp
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 $\pi(n), \lim_{n\to\infty} \pi(n) = \frac{n}{\ln n}$
9. : x $N = log10(n) + 1$
10. $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$
11. $a > 1, m, n > 0,\ \ gcd(a^m - 1, a^n - 1) = a^{gcd(m,n)} - 1$
12. $a > b, gcd(a, b) = 1,\ \ gcd(a^m - b^m, a^n - b^n) = a^{gcd(m,n)} - b^{gcd(m,n)}$

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

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

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

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

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

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;
        }
    }
}
```

```cpp
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

O(n)

```cpp
const int N = "Edit";
char s[N], str[N << 1];
int p[N << 1];
void Manacher(char s[], int& n)
{
    str[0] = '$', str[1] = '#';
    for (int i = 0; i < n; i++) str[(i << 1) + 2] = s[i], str[(i << 1) + 3] = '#';
    n = 2 * n + 2;
    str[n] = 0;
    int mx = 0, id;
    for (int i = 1; i < n; i++)
    {
        p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
        while (str[i - p[i]] == str[i + p[i]]) p[i]++;
        if (p[i] + i > mx) mx = p[i] + i, id = i;
    }
}
int solve(char s[])
{
    int n = strlen(s);
    Manacher(s, n);
    return *max_elememt(p, p + n) - 1;
}
```

2.4 Aho-Corasick Automaton

```cpp
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]);
}
```

```cpp
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

```cpp
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 Binary Indexed Tree

$O(\log n)$

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

3.2 Segment Tree

```
#define lson rt << 1        //
#define rson rt << 1 | 1    //
#define Lson l, m, lson     //
#define Rson m + 1, r, rson //
void PushUp(int rt);        //  lson rson rt
void PushDown(int rt[, int m]);                     // rt   ,m   (    )
void build(int l, int r, int rt);                   //  rt  ,  [l, r]
void update([...,] int l, int r, int rt)            // rt[l, r]
int query(int L, int R, int l, int r, int rt)       // rt[l, r]  [L, R]
```

3.2.1 Single-point Update

```
const int maxn = "Edit";
int sum[maxn << 2]; // sum[rt]
void PushUp(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 m = (l + r) >> 1;
    build(Lson);
    build(Rson);
    PushUp(rt);
}
void update(int p, int add, int l, int r, int rt)
{
    if (l == r)
    {
        sum[rt] += add;
        return;
    }
    int m = (l + r) >> 1;
    if (p <= m)
        update(p, add, Lson);
    else
        update(p, add, Rson);
    PushUp(rt);
}
int query(int L, int R, int l, int r, int rt)
{
    if (L <= l && r <= R) return sum[rt];
    int m = (l + r) >> 1, s = 0;
    if (L <= m) s += query(L, R, Lson);
    if (m < R) s += query(L, R, Rson);
    return s;
}
```

3.2.2 Interval Update

```
const int maxn = "Edit";
int seg[maxn << 2], sum[maxn << 2]; // seg[rt]     , PushDown
void PushUp(int rt) { sum[rt] = sum[lson] + sum[rson]; }
void PushDown(int rt, int m)
{
    if (seg[rt] == 0) return;
    seg[lson] += seg[rt];
    seg[rson] += seg[rt];
    sum[lson] += seg[rt] * (m - (m >> 1));
    sum[rson] += seg[rt] * (m >> 1);
    seg[rt] = 0;
}
void build(int l, int r, int rt)
{
    seg[rt] = 0;
    if (l == r)
    {
        scanf("%lld", &sum[rt]);
        return;
    }
    int m = (l + r) >> 1;
    build(Lson);
    build(Rson);
    PushUp(rt);
}
void update(int L, int R, int add, int l, int r, int rt)
{
    if (L <= l && r <= R)
    {
        seg[rt] += add;
        sum[rt] += add * (r - l + 1);
        return;
    }
    PushDown(rt, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L, R, add, Lson);
    if (m < R) update(L, R, add, Rson);
    PushUp(rt);
}
int query(int L, int R, int l, int r, int rt)
{
    if (L <= l && r <= R) return sum[rt];
    PushDown(rt, r - l + 1);
    int m = (l + r) >> 1, ret = 0;
    if (L <= m) ret += query(L, R, Lson);
    if (m < R) ret += query(L, R, Rson);
    return ret;
}
```

3.3 Splay Tree

```
#define key_value ch[ch[root][1]][0]
const int maxn = "Edit";
struct Splay
{
    int a[maxn];
    int sz[maxn], ch[maxn][2], fa[maxn];
    int key[maxn], rev[maxn];
    int root, tot;
    int stk[maxn], top;
    void init(int n)
    {
        tot = 0, top = 0;
        root = newnode(0, -1);
        ch[root][1] = newnode(root, -1);
        for (int i = 0; i < n; i++) a[i] = i + 1;
        key_value = build(0, n - 1, ch[root][1]);
        pushup(ch[root][1]);
        pushup(root);
    }
    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;
    }
    void pushdown(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;
        }
    }
    void pushup(int x) { sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + 1; }
    void rotate(int x, int d)
    {
        int y = fa[x];
        pushdown(y), pushdown(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;
        pushup(y);
    }
    void splay(int x, int goal = 0)
    {
        pushdown(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);
            }
        }
        pushup(x);
        if (goal == 0) root = x;
    }
    int kth(int r, int k)
    {
        pushdown(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);
    }
    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);
        pushup(x);
        return x;
    }
    void select(int l, int r)
    {
        splay(kth(root, l), 0);
        splay(kth(ch[root][1], r - l + 2), root);
    }
    //
};
```

3.4 Functional Segment Tree

k

```
//poj 2104
#include<cstdio>
#include<iostream>
#include<cmath>
#include<queue>
#include<stack>
#include<set>
#include<map>
#include<algorithm>
#include<vector>
#include<string>
```

```
#include<cstring>
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 pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
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);
}
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;
}
int main(){
#ifdef superkunn
    freopen("input.txt","rt",stdin);
#endif
    work();
    return 0;
}
```

3.5 Sparse Table

```
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  [l,r] /
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]);
}
```

RMQ

```
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));
}
```

3.6 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]);
    }
};
```

3.7 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){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;
```

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

```
//cf 610 A
#include<bits/stdc++.h>
using namespace std;
const int INF=1e9;
const int MAXV=5e3+50;
const int MAXE=1e5+50;
int V;
struct edge{int to,cost;};
vector<edge> G[MAXV];
typedef pair<int,int> P;
int d[MAXV];
void dijkstra(int s){
    priority_queue<P,vector<P>,greater<P> > que;
    fill(d,d+V+1,INF);
    d[s]=0;
    que.push(P(0,s));
    while(!que.empty()){
        P t=que.top();
        que.pop();
        int v=t.second;
        if(d[v]<t.first)continue;
        for(int i=0;i<G[v].size();i++){
            edge e=G[v][i];
            if(d[e.to]>d[v]+e.cost){
                d[e.to]=d[v]+e.cost;
                que.push(P(d[e.to],e.to));
            }
        }
    }
}
int mat[405][405];
int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    V=n;
    for(int i=1;i<=m;i++){
        int u,v;
        scanf("%d%d",&u,&v);
        G[u].push_back(edge{v,1});
        G[v].push_back(edge{u,1});
        mat[u][v]=mat[v][u]=1;
    }
    dijkstra(1);
    int ans;
    if(d[n]==INF){
        printf("-1");
        return 0;
    }
    ans=d[n];
    for(int i=1;i<=n;i++)G[i].clear();
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            if(i==j)continue;
            if(mat[i][j]==0){
                G[i].push_back(edge{j,1});
            }
        }
    }
    dijkstra(1);
    if(d[n]==INF){
        printf("-1");
        return 0;
    }
    printf("%d",max(ans,d[n]));
    return 0;
}
```

4.3.2 Spfa

```
//poj 3259
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<queue>
#include<cstring>
using namespace std;
const int INF=1e9;
const int MAXV=500+5;
const int MAXE=2700+5;
int tot;
int head[MAXV];
struct node{
    int to,cost,next;
}edge[MAXE<<1];
int d[MAXV];
queue<int> que;
bool inq[MAXV];
int qtime[MAXV];
void init(){
    tot=0;
    memset(head,-1,sizeof(head));
}
void add_edge(int u,int v,int x){
    edge[tot].to=v;
    edge[tot].cost=x;
    edge[tot].next=head[u];
    head[u]=tot++;
}
bool spfa(int n){
    memset(d,-1,sizeof(d));
    d[1]=0;
    while(!que.empty())que.pop();
    memset(inq,0,sizeof(inq));
    memset(qtime,0,sizeof(qtime));
    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].next){
            int v=edge[i].to;
            int w=edge[i].cost;
            if(d[v]==-1||d[u]+w<d[v]){
                d[v]=d[u]+w;
                if(!inq[v]){
                    inq[v]=1;
                    que.push(v);
                    qtime[v]++;
                    if(qtime[v]>n){
                        return false;
                    }
                }
            }
        }
    }
    return true;
}
int main(){
    int kase;
    scanf("%d",&kase);
    while(kase--){
        init();
        int n,m,w;
        scanf("%d%d%d",&n,&m,&w);
        while(m--){
            int u,v,x;
            scanf("%d%d%d",&u,&v,&x);
            add_edge(u,v,x);
            add_edge(v,u,x);
        }
        while(w--){
            int u,v,x;
            scanf("%d%d%d",&u,&v,&x);
            add_edge(u,v,-x);
        }
        if(!spfa(n)){
            puts("YES");
        }else{
            puts("NO");
        }
    }
    return 0;
}
```

4.4 Topo Sort

Ans ,G ,deg ,map
1, 0

```
const int maxn = "Edit";
int Ans[maxn];
vector<int> G[maxn];
```

```cpp
int deg[maxn];
map<PII, bool> S;
void init(int n)
{
    S.clear();
    for (int i = 0; i < n; i++) G[i].clear();
    clr(deg, 0), clr(Ans, 0);
}
void add_edge(int u, int v)
{
    if (S[mp(u, v)]) return;
    G[u].pb(v), S[mp(u, v)] = 1, deg[v]++;
}
bool Toposort(int n)
{
    int tot = 0;
    queue<int> q;
    for (int i = 0; i < n; ++i)
        if (deg[i] == 0) q.push(i);
    while (!q.empty())
    {
        int u = q.front();
        que.pop();
        Ans[tot++] = u;
        for (auto& v : G[u])
            if (--deg[v] == 0) q.push(t);
    }
    if (tot < n - 1) return false;
    return true;
}
```

4.5 LCA

4.5.1 Tarjan

Tarjan

$O(n + q)$

```cpp
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 DFS+ST

DFS+ST

$O(nlogn + q)$

```
const int maxn = "Edit";
vector<int> G[maxn], sp;
int dep[maxn], dfn[maxn];
PII dp[21][maxn << 1];
void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
    sp.clear();
}
void dfs(int u, int fa)
{
    dep[u] = dep[fa] + 1;
    dfn[u] = sp.size();
    sp.push_back(u);
    for (auto& v : G[u])
    {
        if (v == fa) continue;
        dfs(v, u);
        sp.push_back(u);
    }
}
void initrmq()
{
    int n = sp.size();
    for (int i = 0; i < n; i++) dp[0][i] = {dfn[sp[i]], sp[i]};
    for (int i = 1; (1 << i) <= n; i++)
        for (int j = 0; j + (1 << i) - 1 < n; j++)
            dp[i][j] = min(dp[i - 1][j], dp[i - 1][j + (1 << (i - 1))]);
}
int lca(int u, int v)
{
    int l = dfn[u], r = dfn[v];
    if (l > r) swap(l, r);
    int k = 31 - __builtin_clz(r - l + 1);
    return min(dp[k][l], dp[k][r - (1 << k) + 1]).Y;
}
```

4.6 Depth-First Traversal

4.6.1 Biconnected-Component

```
// bccno
const int maxn = "Edit";
int pre[maxn], iscut[maxn], bccno[maxn], dfs_clock, bcc_cnt;
vector<int> G[maxn], bcc[maxn];
stack<PII> s;
void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
}
inline void add_edge(int u, int v) { G[u].pb(v), G[v].pb(u); }
int dfs(int u, int fa)
{
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for (auto& v : G[u])
    {
        PII e = mp(u, v);
        if (!pre[v])
        {
            //  v
            s.push(e);
            child++;
            int lowv = dfs(v, u);
            lowu = min(lowu, lowv); //  low
            if (lowv >= pre[u])
            {
                iscut[u] = true;
                bcc_cnt++;
                bcc[bcc_cnt].clear(); // !bcc 1
                for (;;)
                {
                    PII x = s.top();
                    s.pop();
                    if (bccno[x.X] != bcc_cnt)
                        bcc[bcc_cnt].pb(x.X), bcc[x.X] = bcc_cnt;
                    if (bccno[x.Y] != bcc_cnt)
                        bcc[bcc_cnt].pb(x.Y), bcc[x.Y] = bcc_cnt;
                    if (x.X == u && x.Y == v) break;
                }
            }
        }
        else if (pre[v] < pre[u] && v != fa)
        {
            s.push(e);
            lowu = min(lowu, pre[v]); //
        }
    }
    if (fa < 0 && child == 1) iscut[u] = 0;
    return lowu;
}
void find_bcc(int n)
{
    //  s ,
    clr(pre, 0), clr(iscut, 0), clr(bccno, 0);
    dfs_clock = bcc_cnt = 0;
    for (int i = 0; i < n; i++)
        if (!pre[i]) dfs(i, -1);
}
```

4.6.2 Strongly Connected Component

```
//cf 999 E
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN = 5005;//
const int MAXM = 5005;//
struct Edge{
    int to,next;
} edge[MAXM];
int head[MAXN],tot;
int Low[MAXN],DFN[MAXN],Stack[MAXN],Belong[MAXN];//Belong  1~scc
int Index,top;
int scc;//
bool Instack[MAXN];
void init(){
    tot = 0;
    memset(head,-1,sizeof(head));
}
void addedge(int u,int v){
    edge[tot].to = v;
    edge[tot].next = head[u];
    head[u] = tot++;
}
void Tarjan(int u){
    int v;
    Low[u] = DFN[u] = ++Index;
    Stack[top++] = u;
    Instack[u] = true;
    for(int i = head[u]; i != -1; i = edge[i].next){
        v = edge[i].to;
        if( !DFN[v] ){
            Tarjan(v);
            if( Low[u] > Low[v] )Low[u] = Low[v];
        }
        else if(Instack[v] && Low[u] > DFN[v])
            Low[u] = DFN[v];
    }
    if(Low[u] == DFN[u]){
        scc++;
        do{
            v = Stack[--top];
            Instack[v] = false;
            Belong[v] = scc;
        }
        while( v != u);
    }
}
void solve(int N){
    memset(DFN,0,sizeof(DFN));
    memset(Instack,0,sizeof(Instack));
    Index = scc = top = 0;
    for(int i = 1; i <= N; i++)
```

```
        if(!DFN[i])
            Tarjan(i);
}
int u[MAXM],v[MAXM],in[MAXN],vis[MAXN];
int n,m,s;
void dfs(int x){
    Belong[x]=Belong[s];
    vis[x]=true;
    for(int i=head[x];i!=-1;i=edge[i].next){
        int e=edge[i].to;
        if(!vis[e])dfs(e);
    }
}
int main(){
    scanf("%d%d%d",&n,&m,&s);
    init();
    for(int i=1;i<=m;i++){
        scanf("%d%d",&u[i],&v[i]);
        addedge(u[i],v[i]);
    }
    solve(n);
    dfs(s);
    int ans=0;
    for(int i=1;i<=m;i++){
        if(Belong[u[i]]!=Belong[v[i]]){
            in[Belong[v[i]]]++;
        }
    }
    set<int> ss;
    for(int i=1;i<=n;i++){
        ss.insert(Belong[i]);
    }
    set<int>::iterator it;
    for(it=ss.begin();it!=ss.end();it++){
        if(*it!=Belong[s]){
            if(in[*it]==0){
                ans++;
            }
        }
    }
    printf("%d",ans);
    return 0;
}
```

4.6.3 2-SAT

```
//hdu 3062
#include<bits/stdc++.h>
using namespace std;
const int MAXV=1e4;
int V;
vector<int> G[MAXV];
vector<int> rG[MAXV];
vector<int> vs;
bool used[MAXV];
int Belong[MAXV];
void init(int x){
    V=x;
    for(int i=0;i<MAXV;i++){
        G[i].clear();
        rG[i].clear();
    }
}
void add_edge(int u,int v){
    G[u].push_back(v);
    rG[v].push_back(u);
}
void dfs(int v){
    used[v]=true;
    for(int i=0;i<G[v].size();i++)
        if(!used[G[v][i]]) dfs(G[v][i]);
    vs.push_back(v);
}
void rdfs(int v,int k){
    used[v]=true;
    Belong[v]=k;
    for(int i=0;i<rG[v].size();i++)
        if(!used[rG[v][i]]) rdfs(rG[v][i],k);
}
int scc(){
    memset(used,0,sizeof(used));
    vs.clear();
    for(int v=1;v<=V;v++){//from 1 to V
        if(!used[v]) dfs(v);
    }
    int k=0;
    memset(used,0,sizeof(used));
    for(int i=vs.size()-1;i>=0;i--){
        if(!used[vs[i]]) rdfs(vs[i],k++);
    }
    return k;
}
bool judge(){
    for(int i=1;i<V;i+=2){
        if(Belong[i]==Belong[i+1])return false;
    }
    return true;
}
int main(){
    int n,m;
    while(scanf("%d%d",&n,&m)!=EOF){
        init(2*n);
        for(int i=1;i<=m;i++){
            int a1,a2,c1,c2;
            scanf("%d%d%d%d",&a1,&a2,&c1,&c2);
            add_edge(((a1*2+c1))+1,((a2*2+c2)^1)+1);
            add_edge(((a2*2+c2))+1,((a1*2+c1)^1)+1);
        }
        scc();
        printf("%s\n",judge()?"YES":"NO");
    }
    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

1.
2. =$|G|$-
   $N \times N$, , , ;
   (,); , .
   :
   (a) ;
   (b) $p_1, p_2, ...p_k$, p_1 ,p_k , , $p_1, p2, ...p_k$.
   , G .
   : =$|G|$- ;
3. = -
   :

4.8.1 Hungry(Matrix)

:$O(VE)$.
0

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

4.8.2 Hungry(List)

init()
addedge(u,v)

```
const int maxn = "Edit";
int n;
vector<int> G[maxn];
int linker[maxn];
bool used[maxn];
inline void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
}
inline void addedge(int u, int v) { G[u].pb(v); }
bool dfs(int u)
{
    for (auto& v : G[u])
    {
        if (!used[v])
        {
            used[v] = true;
            if (linker[v] == -1 || dfs(linker[v]))
            {
                linker[v] = u;
                return true;
            }
        }
    }
    return false;
}
int hungary()
{
    int ans = 0;
    clr(linker, -1);
    for (int u = 0; u < n; v++)
    {
        clr(used, 0);
        if (dfs(u)) ans++;
    }
    return ans;
}
```

4.8.3 Hopcroft-Carp

$O(\sqrt{n} * E)$

uN , (0)

```
const int maxn = "Edit";
vector<int> G[maxn];
int uN;
int Mx[maxn], My[maxn];
int dx[maxn], dy[maxn];
int dis;
bool used[maxn];
inline void init(int n)
{
    for (int i = 0; i < n; i++) G[i].clear();
}
inline void addedge(int u, int v) { G[u].pb(v); }
bool bfs()
{
    queue<int> q;
    dis = INF;
    clr(dx, -1), clr(dy, -1);
    for (int i = 0; i < uN; i++)
        if (Mx[i] == -1)
            q.push(i), dx[i] = 0;
    while (!q.empty())
    {
        int u = q.front();
        q.pop();
        if (dx[u] > dis) break;
        for (auto& v : G[u])
        {
            if (dy[v] == -1)
            {
                dy[v] = dx[u] + 1;
                if (My[v] == -1)
                    dis = dy[v];
                else
                {
                    dx[My[v]] = dy[v] + 1;
                    q.push(My[v]);
                }
            }
        }
    }
    return dis != INF;
}
bool dfs(int u)
{
    for (auto& v : G[u])
    {
        if (!used[v] && dy[v] == dx[u] + 1)
        {
            used[v] = true;
            if (My[v] != -1 && dy[v] == dis) continue;
            if (My[v] == -1 || dfs(My[v]))
            {
                My[v] = u, Mx[u] = v;
                return true;
            }
        }
    }
    return false;
}
int MaxMatch()
{
    int res = 0;
    clr(Mx, -1), clr(My, -1);
    while (bfs())
    {
        clr(used, false);
        for (int i = 0; i < uN; i++)
            if (Mx[i] == -1 && dfs(i)) res++;
    }
    return res;
}
```

4.8.4 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;
}
```

4.8.5 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

```
struct Edge
{
    int from, to, cap, flow;
    Edge(int u, int v, int c, int f)
        : from(u), to(v), cap(c), flow(f) {}
};
```

```
struct Edge
{
    int from, to, cap, flow, cost;
    Edge(int u, int v, int c, int f, int w)
        : from(u), to(v), cap(c), flow(f), cost(w) {}
};
```

, , ,

: S T, S X , Y T , X Y , , , , ,

m n , , $\{p_1, p_2\}$

: , X ,Y x, , S , 1, , , 1, T, x, x
S , , T m , $O(\log m)$

k , k

: , , w $[u,v)$ $u \to v$, 1, $-w$ $i \to i+1$, k, 0 , , ,

$G($ $)$, , ,

: s t, s , ; , ,$S - \{s\}$

4.9.1 EdmondKarp

```cpp
const int maxn = "Edit";
struct EdmonsKarp //  O(v*E*E)
{
    int n, m;
    vector<Edge> edges;  //
    vector<int> G[maxn]; // ,G[i][j]  i j  e
    int a[maxn];         // i
    int p[maxn];         //  p
    void init(int n)
    {
        for (int i = 0; i < n; i++) G[i].clear();
        edges.clear();
    }
    void AddEdge(int from, int to, int cap)
    {
        edges.pb(Edge(from, to, cap, 0));
        edges.pb(Edge(to, from, 0, 0)); //
        m = edges.size();
        G[from].pb(m - 2);
        G[to].pb(m - 1);
    }
    int Maxflow(int s, int t)
    {
        int flow = 0;
        for (;;)
        {
            clr(a, 0);
            queue<int> q;
            q.push(s);
            a[s] = INF;
            while (!q.empty())
            {
                int x = q.front();
                q.pop();
                for (int i = 0; i < G[x].size(); i++)
                {
                    Edge& e = edges[G[x][i]];
                    if (!a[e.to] && e.cap > e.flow)
                    {
                        p[e.to] = G[x][i];
                        a[e.to] = min(a[x], e.cap - e.flow);
                        q.push(e.to);
                    }
                }
                if (a[t]) break;
            }
            if (!a[t]) break;
            for (int u = t; u != s; u = edges[p[u]].from)
            {
                edges[p[u]].flow += a[t];
                edges[p[u] ^ 1].flow -= a[t];
            }
            flow += a[t];
        }
        return flow;
    }
};
```

4.9.2 Dinic

```
const int maxn = "Edit";
struct Dinic
{
    int n, m, s, t;       // , (   ),
    vector<Edge> edges;   //  edge[e] edge[e^1]
    vector<int> G[maxn];  // ,G[i][j]  i j  e
    bool vis[maxn];       //BFS
    int d[maxn];          //  i
    int cur[maxn];        //
    void init(int n)
    {
        this->n = n;
        for (int i = 0; i < n; i++) G[i].clear();
        edges.clear();
    }
    void AddEdge(int from, int to, int cap)
    {
        edges.pb(Edge(from, to, cap, 0));
        edges.pb(Edge(to, from, 0, 0));
        m = edges.size();
        G[from].pb(m - 2);
        G[to].pb(m - 1);
    }
    bool BFS()
    {
        clr(vis, 0);
        clr(d, 0);
        queue<int> q;
        q.push(s);
        d[s] = 0;
        vis[s] = 1;
        while (!q.empty())
        {
            int x = q.front();
            q.pop();
            for (int i = 0; i < G[x].size(); i++)
            {
                Edge& e = edges[G[x][i]];
                if (!vis[e.to] && e.cap > e.flow)
                {
                    vis[e.to] = 1;
                    d[e.to] = d[x] + 1;
                    q.push(e.to);
                }
            }
        }
        return vis[t];
    }
    int DFS(int x, int a)
    {
        if (x == t || a == 0) return a;
        int flow = 0, f;
        for (int& i = cur[x]; i < G[x].size(); i++)
        {
            //
            Edge& e = edges[G[x][i]];
            if (d[x] + 1 == d[e.to] && (f = DFS(e.to, min(a, e.cap - e.flow))) > 0)
            {
                e.flow += f;
                edges[G[x][i] ^ 1].flow -= f;
                flow += f;
                a -= f;
                if (a == 0) break;
            }
        }
        return flow;
    }
    int Maxflow(int s, int t)
    {
        this->s = s;
        this->t = t;
        int flow = 0;
        while (BFS())
        {
            clr(cur, 0);
            flow += DFS(s, INF);
        }
        return flow;
    }
};
```

4.9.3 ISAP

```
const int maxn = "Edit";
struct ISAP
{
    int n, m, s, t;      // , (   ),
    vector<Edge> edges;  // edges[e] edges[e^1]
    vector<int> G[maxn]; // ,G[i][j]  i j  e
    bool vis[maxn];      //BFS
    int d[maxn];         // i
    int cur[maxn];       //
    int p[maxn];         //
    int num[maxn];       //
    void init(int n)
    {
        this->n = n;
        for (int i = 0; i < n; i++) G[i].clear();
        edges.clear();
    }
    void AddEdge(int from, int to, int cap)
    {
        edges.pb(Edge(from, to, cap, 0));
        edges.pb(Edge(to, from, 0, 0));
        int m = edges.size();
        G[from].pb(m - 2);
        G[to].pb(m - 1);
    }
    int Augumemt()
    {
        int x = t, a = INF;
        while (x != s)
        {
            Edge& e = edges[p[x]];
            a = min(a, e.cap - e.flow);
            x = edges[p[x]].from;
        }
        x = t;
        while (x != s)
        {
            edges[p[x]].flow += a;
            edges[p[x] ^ 1].flow -= a;
            x = edges[p[x]].from;
        }
        return a;
    }
    void BFS()
    {
        clr(vis, 0);
        clr(d, 0);
        queue<int> q;
        q.push(t);
        d[t] = 0;
        vis[t] = 1;
        while (!q.empty())
        {
            int x = q.front();
            q.pop();
            int len = G[x].size();
            for (int i = 0; i < len; i++)
            {
                Edge& e = edges[G[x][i]];
                if (!vis[e.from] && e.cap > e.flow)
                {
                    vis[e.from] = 1;
                    d[e.from] = d[x] + 1;
                    q.push(e.from);
                }
            }
        }
    }
    int Maxflow(int s, int t)
    {
        this->s = s;
        this->t = t;
        int flow = 0;
        BFS();
        clr(num, 0);
        for (int i = 0; i < n; i++)
            if (d[i] < INF) num[d[i]]++;
        int x = s;
        clr(cur, 0);
        while (d[s] < n)
        {
            if (x == t)
            {
                flow += Augumemt();
                x = s;
            }
            int ok = 0;
            for (int i = cur[x]; i < G[x].size(); i++)
            {
                Edge& e = edges[G[x][i]];
                if (e.cap > e.flow && d[x] == d[e.to] + 1)
                {
                    ok = 1;
                    p[e.to] = G[x][i];
                    cur[x] = i;
                    x = e.to;
                    break;
                }
            }
            if (!ok) //Retreat
            {
                int m = n - 1;
                for (int i = 0; i < G[x].size(); i++)
                {
                    Edge& e = edges[G[x][i]];
                    if (e.cap > e.flow) m = min(m, d[e.to]);
                }
                if (--num[d[x]] == 0) break; //gap
                num[d[x] = m + 1]++;
                cur[x] = 0;
                if (x != s) x = edges[p[x]].from;
            }
        }
        return flow;
    }
};
```

4.9.4 MinCost MaxFlow

```
const int maxn = "Edit";
struct MCMF
{
    int n, m;
    vector<Edge> edges;
    vector<int> G[maxn];
    int inq[maxn]; //
    int d[maxn];   //bellmanford
    int p[maxn];   //
    int a[maxn];   //
    void init(int n)
    {
        this->n = n;
        for (int i = 0; i < n; i++) G[i].clear();
        edges.clear();
    }
    void AddEdge(int from, int to, int cap, int cost)
    {
        edges.pb(Edge(from, to, cap, 0, cost));
        edges.pb(Edge(to, from, 0, 0, -cost));
        m = edges.size();
        G[from].pb(m - 2);
        G[to].pb(m - 1);
    }
    bool BellmanFord(int s, int t, int& flow, ll& cost)
    {
        for (int i = 0; i < n; i++) d[i] = INF;
        clr(inq, 0);
        d[s] = 0;
        inq[s] = 1;
        p[s] = 0;
        a[s] = INF;
        queue<int> q;
        q.push(s);
        while (!q.empty())
        {
            int u = q.front();
            q.pop();
            inq[u] = 0;
            for (int i = 0; i < G[u].size(); i++)
            {
                Edge& e = edges[G[u][i]];
                if (e.cap > e.flow && d[e.to] > d[u] + e.cost)
                {
                    d[e.to] = d[u] + e.cost;
                    p[e.to] = G[u][i];
                    a[e.to] = min(a[u], e.cap - e.flow);
                    if (!inq[e.to])
                    {
                        q.push(e.to);
                        inq[e.to] = 1;
                    }
                }
            }
        }
        if (d[t] == INF) return false; //
        flow += a[t];
        cost += (ll)d[t] * (ll)a[t];
        for (int u = t; u != s; u = edges[p[u]].from)
        {
            edges[p[u]].flow += a[t];
            edges[p[u] ^ 1].flow -= a[t];
        }
        return true;
    }
    int MincostMaxflow(int s, int t, ll& cost)
    {
        int flow = 0;
        cost = 0;
        while (BellmanFord(s, t, flow, cost));
        return flow;
    }
};
```

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

```cpp
//    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

```cpp
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 > 9) out(x / 10);
    putchar(x % 10 + '0');
}
```

7.2.3 Strok-Sscanf

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

7.3 Mo Algorithm

, , $\sqrt{x}$,

```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

```
//
struct bint
{
    int l;
    short int w[100];
    bint(int x = 0)
    {
        l = x == 0, clr(w, 0);
        while (x) w[l++] = x % 10, x /= 10;
    }
    bool operator<(const bint& x) const
    {
        if (l != x.l) return l < x.l;
        int i = l - 1;
        while (i >= 0 && w[i] == x.w[i]) i--;
        return (i >= 0 && w[i] < x.w[i]);
    }
    bint operator+(const bint& x) const
    {
        bint ans;
        ans.l = l > x.l ? l : x.l;
        for (int i = 0; i < ans.l; i++)
        {
            ans.w[i] += w[i] + x.w[i];
            ans.w[i + 1] += ans.w[i] / 10;
            ans.w[i] = ans.w[i] % 10;
        }
        if (ans.w[ans.l] != 0) ans.l++;
        return ans;
    }
    bint operator*(const bint& x) const
    {
        bint res;
        int up, tmp;
        for (int i = 0; i < l; i++)
        {
            up = 0;
            for (int j = 0; j < x.l; j++)
            {
                tmp = w[i] * x.w[j] + res.w[i + j] + up;
                res.w[i + j] = tmp % 10;
                up = tmp / 10;
            }
            if (up != 0) res.w[i + x.l] = up;
        }
        res.l = l + x.l;
        while (res.w[res.l - 1] == 0 && res.l > 1) res.l--;
        return res;
    }
    void print()
    {
        for (int i = l - 1; ~i; i--) printf("%d", w[i]);
        puts("");
    }
};
```

7.4.2 Complete High-precision

```
#define N 10000
class bint
{
private:
    int a[N]; //   N
    int len;  //
public:
    //
    bint() { len = 1, clr(a, 0); }
    // int -> bint
    bint(int n)
    {
        len = 0;
        clr(a, 0);
        int d = n;
        while (n)
            d = n / 10 * 10, a[len++] = n - d, n = d / 10;
    }
    // char[] -> int
    bint(const char s[])
    {
        clr(a, 0);
        len = 0;
        int l = strlen(s);
        for (int i = l - 1; ~i; i--) a[len++] = s[i];
    }
    //
    bint(const bint& b)
    {
        clr(a, 0);
        len = b.len;
        for (int i = 0; i < len; i++) a[i] = b.a[i];
    }
    //    bint = bint
    bint& operator=(const bint& n)
    {
        len = n.len;
        for (int i = 0; i < len; i++) a[i] = n.a[i];
        return *this;
    }
    //    bint + bint
    bint operator+(const bint& b) const
    {
        bint t(*this);
        int res = b.len > len ? b.len : len;
        for (int i = 0; i < res; i++)
        {
            t.a[i] += b.a[i];
            if (t.a[i] >= 10) t.a[i + 1]++, t.a[i] -= 10;
        }
        t.len = res + a[res] == 0;
        return t;
    }
    //    bint - bint
    bint operator-(const bint& b) const
    {
        bool f = *this > b;
        bint t1 = f ? *this : b;
        bint t2 = f ? b : *this;
        int res = t1.len, j;
        for (int i = 0; i < res; i++)
            if (t1.a[i] < t2.a[i])
            {
                j = i + 1;
                while (t1.a[j] == 0) j++;
                t1.a[j--]--;
                while (j > i) t1.a[j--] += 9;
                t1.a[i] += 10 - t1.a[i];
            }
            else
                t1.a[i] -= t2.a[i];
        t1.len = res;
        while (t1.a[len - 1] == 0 && t1.len > 1) t1.len--, res--;
        if (f) t1.a[res - 1] = 0 - t1.a[res - 1];
        return t1;
    }
    //    bint * bint
    bint operator*(const bint& b) const
    {
        bint t;
        int i, j, up, tmp, tmp1;
        for (i = 0; i < len; i++)
        {
            up = 0;
            for (j = 0; j < b.len; j++)
            {
                tmp = a[i] * b.a[j] + t.a[i + j] + up;
                if (tmp > 9)
                    tmp1 = tmp - tmp / 10 * 10, up = tmp / 10, t.a[i + j] = tmp1;
                else
                    up = 0, t.a[i + j] = tmp;
            }
            if (up) t.a[i + j] = up;
        }
        t.len = i + j;
        while (t.a[t.len - 1] == 0 && t.len > 1) t.len--;
        return t;
    }
    //    bint / int
    bint operator/(const int& b) const
    {
        bint t;
        int down = 0;
        for (int i = len - 1; ~i; i--)
            t.a[i] = (a[i] + down * 10) / b, down = a[i] + down * 10 - t.a[i] * b;
        t.len = len;
        while (t.a[t.len - 1] == 0 && t.len > 1) t.len--;
        return t;
    }
    //    bint ^ n (n   ,   n )
    bint operator^(const int n) const
    {
        bint t(*this), rt(1);
        if (n == 0) return 1;
        if (n == 1) return *this;
        int m = n;
        for (; m; m >>= 1, t = t * t)
            if (m & 1) rt = rt * t;
        return rt;
    }
    //    bint > bint
    bool operator>(const bint& b) const
    {
        int p;
        if (len > b.len) return 1;
        if (len == b.len)
        {
            p = len - 1;
            while (a[p] == b.a[p] && p >= 0) p--;
            return p >= 0 && a[p] > b.a[p];
        }
        return 0;
    }
    //    bint > int
    bool operator>(const int& n) const { return *this > bint(n); }
    //
    void out()
    {
        for (int i = len - 1; ~i; i--) printf("%d", a[i]);
        puts("");
    }
};
```

7.5 VIM

```
syntax on
set cindent
set nu
set tabstop=4
set shiftwidth=4
set background=dark
set mouse=a

map<C-A> ggvG"+y
map<F5> :call Run()<CR>

func! Run()
    exec "w"
    exec "!g++ -std=c++11 -O2 % -o %<"
    exec "!time ./%<"
endfunc

autocmd BufNewFile *.cpp 0r ~/include.cpp
autocmd BufNewFile *.cpp normal G

inoremap ( ()<Esc>i
inoremap [ []<Esc>i
inoremap { {<CR>}<Esc>O
inoremap ' ''<Esc>i
inoremap " ""<Esc>i

inoremap ) <c-r>=ClosePair(')')<CR>
inoremap ] <c-r>=ClosePair(']')<CR>

func ClosePair(char)
    if getline('.')[col('.')-1]==a:char
        return "\<Right>"
    else
        return a:char
    endif
endfunc
```