数论

区间筛

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN = 1e5;
bool isPrime[MAXN]; //根据isPrime[i]判断i+a是不是质数，isPrime[i-a]=true 表示i是素数（下标偏移了a）
bool isSqrtPrime[MAXN]; //根据isSqrtPrime[i]判断i是不是质数
vector<int> prime;
//对区间[a,b)内的整数执行筛法
void SegmentSieve(ll a, ll b)   //对区间[a,b)进行筛选
{
    for (ll i = 0; i * i < b; i++) //初始化[2,sqrt(b))的全为质数
        isSqrtPrime[i] = true;
    for (ll i = 0; i < b - a; i++) //初始化偏移后的[a,b)全为质数
        isPrime[i] = true;

    for (ll i = 2; i * i < b; i++) //埃氏筛选
    {
        if (isSqrtPrime[i]) {
            for (ll j = 2 * i; j  <= sqrt(b); j += i)//筛选上限变为sqrt(b)
                isSqrtPrime[j] = false;
            //筛除[a,b)里的非素数
            //(a+i-1)/i 得到最接近（a/i）这个数，也即（a的i倍数），最低是2LL是表示最低2倍
            for (ll j = max(2LL, (a + i - 1) / i) * i; j < b; j += i)
                isPrime[j - a] = false;
        }
    }
    for (ll i = a; i < b; i++){
        if (isPrime[i - a]) prime.push_back(i);
    }
}


typedef long long ll;
const int MAXN = 1e6 + 5;
vector<int> primes;
void segment_sieve(ll a, ll b) {
    vector<bool> is_prime_small(sqrt(b) + 1, true); // 筛[2, sqrt(b)]
    vector<bool> is_prime(b - a, true);              // 筛[a, b)

    if (a == 1) is_prime[0] = false;

    for (ll p = 2; p * p < b; p++) {
        if (is_prime_small[p]) {
```

```
            // 筛小素数
            for (ll j = p * p; j * j < b; j += p)
                is_prime_small[j] = false;
            // 筛大区间
            ll start = max(p * p, (a + p - 1) / p * p);
            for (ll j = start; j < b; j += p)
                is_prime[j - a] = false;
        }
    }

    for (ll i = a; i < b; i++)
        if (is_prime[i - a]) primes.push_back(i);
}
int main(){
    segment_sieve(1000000000, 1001000000);
    for (auto v : primes){
        cout << v << " ";
    }
    return 0;
}



int main(){
    SegmentSieve(100, 10000);
    for (auto v : prime){
        cout << v << " ";
    }
    return 0;
}
```

FTT

```
#include <bits/stdc++.h>
using namespace std;
struct Polynomial
{
    double PI = acos(-1);
    struct Complex
    {
        double x, y;
        Complex(double _x = 0.0, double _y = 0.0)
        {
            x = _x;
            y = _y;
        }
        Complex operator-(const Complex &rhs) const
        {
            return Complex(x - rhs.x, y - rhs.y);
        }
```

```
        Complex operator+(const Complex &rhs) const
        {
            return Complex(x + rhs.x, y + rhs.y);
        }
        Complex operator*(const Complex &rhs) const
        {
            return Complex(x * rhs.x - y * rhs.y, x * rhs.y + y * rhs.x);
        }
    };
    vector<Complex> c;
    Polynomial(vector<int> &a)
    {
        int n = a.size();
        c.resize(n);
        for (int i = 0; i < n; i++)
        {
            c[i] = Complex(a[i], 0);
        }
        fft(c, n, 1);
    }
    // void change(vector<Complex> &a, int n)
    // {
    //     for (int i = 1, j = 0; i < n; i++)
    //     {
    //         int bit = n >> 1;
    //         for (; j & bit; bit >>= 1)
    //             j ^= bit;
    //         j |= bit;
    //         if (i < j)
    //             swap(a[i], a[j]);
    //     }
    // }
    void change(vector<Complex> &a, int n)
    {
        for (int i = 1, j = n / 2; i < n - 1; i++)
        {
            if (i < j)
                swap(a[i], a[j]);
            int k = n / 2;
            while (j >= k)
            {
                j -= k;
                k /= 2;
            }
            if (j < k)
                j += k;
        }
    }
    void fft(vector<Complex> &a, int n, int opt)
    {
        change(a, n);
        for (int h = 2; h <= n; h *= 2)
        {
            Complex wn(cos(2 * PI / h), sin(opt * 2 * PI / h));
```

```
            for (int j = 0; j < n; j += h)
            {
                Complex w(1, 0);
                for (int k = j; k < j + h / 2; k++)
                {
                    Complex u = a[k];
                    Complex t = w * a[k + h / 2];
                    a[k] = u + t;
                    a[k + h / 2] = u - t;
                    w = w * wn;
                }
            }
        }
        if (opt == -1)
        {
            for (int i = 0; i < n; i++)
            {
                a[i].x /= n;
            }
        }
    }
};
int main()
{
    // 1 * (x^2) + 2 * (x^1) + 3
    // 3 * (x^1) + 4
    vector<int> a = {3, 2, 1}, b = {4, 3};
    int n = a.size(), m = b.size();
    int len = 1;
    while (len < n * 2 || len < m * 2)
        len <<= 1;
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    a.resize(len);
    b.resize(len);
    Polynomial f1(a), f2(b);
    for (int i = 0; i < len; i++)
    {
        f1.c[i] = f1.c[i] * f2.c[i];
    }
    f1.fft(f1.c, len, -1);
    len = n + m - 1;
    while (int(f1.c[len].x + 0.5) == 0)
        len--;
    for (int i = 0; i <= len; i++)
    {
        cout << int(f1.c[i].x + 0.5) << endl;
    }
    return 0;
}
```

NTT (与FTT相比，都是整数，精度更准确)

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

template <int T>
struct MInt {
    int x;
    MInt() : x(0) {}
    MInt(int y) : x(y % T) { if (x < 0) x += T; }
    MInt operator+(const MInt &y) const { return MInt((x + y.x) % T); }
    MInt operator-(const MInt &y) const { return MInt((x - y.x + T) % T); }
    MInt operator*(const MInt &y) const { return MInt(1LL * x * y.x % T); }
    MInt& operator+=(const MInt &y) { x = (x + y.x) % T; return *this; }
    MInt& operator-=(const MInt &y) { x = (x - y.x + T) % T; return *this; }
    MInt& operator*=(const MInt &y) { x = 1LL * x * y.x % T; return *this; }
    bool operator==(const MInt &y) const { return x == y.x; }
    bool operator!=(const MInt &y) const { return x != y.x; }
    friend ostream& operator<<(ostream &os, const MInt &m) { return os << m.x; }
};
constexpr int mod=1e9+7;

using Z=MInt<mod>;
// mod  一定要满足 mod = len * k + 1; len 为 2 的次幂长度  一定要注意模数的选取！！！
// ntt 从 0 - n 的位置依次存放权值从小到大的
struct Polynomial
{
    vector<Z> z;
    vector<int> r;
    Polynomial(vector<int> &a)
    {
        int n = a.size();
        z.resize(n);
        r.resize(n);
        for (int i = 0; i < n; i++)
        {
            z[i] = a[i];
            r[i] = (i & 1) * (n / 2) + r[i / 2] / 2;
        }
        ntt(z, n, 1);
    }
    LL power(LL a, int b)
    {
        LL res = 1;
        for (; b; b /= 2, a = a * a % mod)
        {
            if (b % 2)
            {
                res = res * a % mod;
            }
```

```
        }
        return res;
    }
    void ntt(vector<Z> &a, int n, int opt)
    {
        for (int i = 0; i < n; i++)
        {
            if (r[i] < i)
            {
                swap(a[i], a[r[i]]);
            }
        }
        for (int k = 2; k <= n; k *= 2)
        {
            Z gn = power(3, (mod - 1) / k);
            for (int i = 0; i < n; i += k)
            {
                Z g = 1;
                for (int j = 0; j < k / 2; j++, g *= gn)
                {
                    Z t = a[i + j + k / 2] * g;
                    a[i + j + k / 2] = a[i + j] - t;
                    a[i + j] = a[i + j] + t;
                }
            }
        }
        if (opt == -1)
        {
            reverse(a.begin() + 1, a.end());
            Z inv = power(n, mod - 2);
            for (int i = 0; i < n; i++)
            {
                a[i] *= inv;
            }
        }
    }
};
/*输入：
5
0 1
10 10
10 11
101 101
110 110
*/

/*输出：进制为 sqrt(-2)时
0
100
110
1
10110100
*/
```

```
void solve(){
    string s1, s2;
    cin >> s1 >> s2;
    int n = s1.size(), m = s2.size();
    int len = 1;
    while (len < n * 2 || len < m * 2) len <<= 1;
    vector<int> a(n), b(m);
    for (int i = 0; i < n; i++){
        a[i] = s1[i] - '0';
    }
    for (int i = 0; i < m; i++){
        b[i] = s2[i] - '0';
    }
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    a.resize(len);
    b.resize(len);
    Polynomial f1(a), f2(b);
    for (int i = 0; i < len; i++){
        f1.z[i] = f1.z[i] * f2.z[i];
    }
    f1.ntt(f1.z, len, -1);
    len = n + m - 1;
    vector<int> ans(len + 100);
    for (int i = 0; i < len; i++){
        // cout << f1.z[i].x << " ";
        ans[i] = f1.z[i].x;
    }
    // cout << endl;
    for (int i = 0; i <= len; i++){
        // cout << ans[i] << " ";
        if (i == ans.size()) break;
        if (ans[i] > 1){
            ans[i + 2] -= ans[i] / 2;
            ans[i] %= 2;
            len = max(len, i + 2);
        }
        if(ans[i] < 0){
            ans[i + 2] += -ans[i] / 2;
            ans[i] %= 2;
            if (ans[i] < 0){
                ans[i] = 1;
                ans[i + 2]++;
            }
            len = max(len, i + 2);
        }
    }
    // cout << endl;
    while (ans[len] == 0) len--;
    len = max(len, 0);
    for (int i = len; i >= 0; i--) cout << ans[i];
    cout << endl;
}
int main()
```

```
{
    ios::sync_with_stdio(false);
    cin.tie(0), cout.tie(0);
    int t = 1;
    cin >> t;
    while (t--) solve();
    return 0;
}
```

高斯消元

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

const int N = 110;
const double eps = 1e-8;
LL n;
vector<vector<double>> a(N , vector<double>(N));
LL gauss(){
    LL c, r;
    for (c = 0, r = 0; c < n; c++){
        LL t = r;
        for (int i = r; i < n; i++){
            if (fabs(a[i][c]) > fabs(a[t][c])){
                t = i;
            }
        }
        if (fabs(a[t][c]) < eps) continue;
        for (int j = c; j < n + 1; j++){
            swap(a[t][j], a[r][j]);
        }
        for (int j = n; j >= c; j--) a[r][j] /= a[r][c];
        for (int i = r + 1; i < n; i++){
            if (fabs(a[i][c]) > eps){
                for (int j = n; j >= c; j--){
                    a[i][j] -= a[r][j] * a[i][c];
                }
            }
        }
        r++;
    }
    if (r < n){
        for (int i = r; i < n; i++){
            if (fabs(a[i][n]) > eps) return 2;
        }
        return 1;
    }
    for (int i = n - 1; i >= 0; i--){
        for (int j = i + 1; j < n; j++){
```

```
            a[i][n] -= a[i][j] * a[j][n];
        }
    }
    return 0;
}
/* 最后第n列存储等式结果， 前0 - (n - 1) 列为未知量系数
a11 * x1 + a12 * x2 + ... + a1n * xn = k1
a21 * x1 + ... = k2
*/
int main(){
    cin >> n;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n + 1; j++){
            cin >> a[i][j];
        }
    }
    LL t = gauss();
    if (t == 0){
        for (int i = 0; i < n; i++){
            if (fabs(a[i][n]) < eps) a[i][n] = abs(a[i][n]);
            cout << a[i][n] << endl;
        }
    }
    else if (t == 1) cout << "多个解" << endl;
    else cout << "无解" << endl;
}
```

Min25筛

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
namespace min25{
    // 求解 1 - n 的质数的和，n 最大为1e10
    const int N = 1e6 + 10;
    int prime[N], id1[N], id2[N], flag[N], ncnt, m;
    ll g[N], sum[N], a[N], T;
    ll n, mod;
    inline ll ps(ll n, ll k){
        ll r = 1;
        for (; k; k >>= 1){
            if (k & 1){
                r = r * n % mod;
            }
            n = n * n % mod;
        }
        return r;
    }
    void finit(){
        // 最开始清 0
```

```
        memset(g, 0, sizeof(g));
        memset(a, 0, sizeof(a));
        memset(sum, 0, sizeof(sum));
        memset(prime, 0, sizeof(prime));
        memset(id1, 0, sizeof(id1));
        memset(id2, 0, sizeof(id2));
        memset(flag, 0, sizeof(flag));
        ncnt = m = 0;
    }
    int ID(ll x){
        return x <= T ? id1[x] : id2[n / x];
    }
    ll calc(ll x){
        return x * (x + 1) / 2 - 1;
    }
    ll init(ll x){
        T = sqrt(x + 0.5);
        for (int i = 2; i <= T; i++){
            if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
            for (int j = 1; j <= ncnt && i * prime[j] <= T; j++){
                flag[i * prime[j]] = 1;
                if (i % prime[j] == 0) break;
            }
        }
        for (ll L = 1; L <= x; L = x / (x / L) + 1){
            a[++m] = x / L;
            if (a[m] <= T) id1[a[m]] = m;
            else id2[x / a[m]] = m;
            g[m] = calc(a[m]);
        }
        for (int i = 1; i <= ncnt; i++){
            for (int j = 1; j <= m && (ll) prime[i] * prime[i] <= a[j]; j++){
                g[j] = g[j] - (ll) prime[i] * (g[ID(a[j] / prime[i])] - sum[i -
1]);
            }
        }
    }
    ll solve(ll x){
        if (x <= 1) return x;
        return n = x, init(n), g[ID(n)];
    }
}

using namespace min25;

int main(){
    int tt;
    cin >> tt;
    while (tt--){
        finit();
        cin >> n >> mod;
        // cout << n << " " << mod << endl;
        // ll ans = (n + 3) % mod * n % mod * ps(2, mod - 2) % mod + solve(n + 1)
- 4;
```

```
        // ans = (ans + mod) % mod;
        cout << solve(n) << endl;
    }
}
```