数学

目录

线性同余方程最小正根

```
long long egcd(long long a, long long b, long long &rx, long long
&ry){
    if (b == 0) {
        rx = 1;
        ry = 0;
        return a;
    }
    long long p, q;
    long long d = egcd(b, a % b, p, q);
    rx = q;
    ry = p - a / b * q;
    return d;
}
int X, Y, M, N, L;
void work() {
    long long d, x, y;
    d = egcd(M - N, L, x, y);
    if ((Y - X) % d == 0) {
        long long x0 = (x * ((Y - X) / d) % L + L) % L;
        cout << x0 % (L / d) << endl;
    } else
        puts("Impossible");
}
```

模高斯消元

```
//N列M行，N+1列为增广阵的增广列。
void swapLine(int p,int q){
    int tmp[350];
    memcpy(tmp,mat[p],sizeof(tmp));
    memcpy(mat[p],mat[q],sizeof(tmp));
    memcpy(mat[q],tmp,sizeof(tmp));
}
void work(){
    int current=1;
    for (int i=1;i<=N;i++){
        for (int j=current;j<=M;j++){
            if (mat[j][i]){
                swapLine(current,j);
                for (int k=current+1;k<=M;k++){
                    int fac=mat[k][i];
                    for (int l=i;l<=N+1;l++){
                                        mat[k][l]=mat[k]
[l]*mat[current][i]-fac*mat[current][l];
                        mat[k][l]%=7;
                        mat[k][l]+=7;
                        mat[k][l]%=7;
                    }
                }
                current++;
                break;
            }
        }
    }
    int rank=0;
    for (int i=1;i<=M;i++){
        int flag=1;
        for (int j=1;j<=N;j++){
            if (mat[i][j]){
                flag=0;
                break;
            }
        }
        if (flag && mat[i][N+1]){
            puts("Inconsistent data.");
            return;
        }
        if (!flag) rank++;
    }
    if (rank<N){
        puts("Multiple solutions.");
        return;
    }
    for (int i=N;i>=1;i--){
        int s=0;
        for (int j=i+1;j<=N;j++){
            s+=result[j]*mat[i][j];
            s%=7;
        }
        s=mat[i][N+1]-s;
        s=(s%7+7)%7;
        result[i]=value[mat[i][i]][s];//计算mat[i][i]/s
    }
    for (int i=1;i<=N;i++){
        printf("%d",result[i]);
        if (i==N) puts("");
        else putchar(' ');
    }
}
```

在线高斯消元 Ural 1561

```
int N, M, C, crazy;
int mat[1010][1010], trans[1010], sol[1010], b[1010];
const char str[][10] = { "Sunday", "Monday", "Tuesday", "Wednesday",
        "Thursday", "Friday", "Saturday" };
int value[10][10];
void buildValue() {
    for (int i = 0; i < 7; i++)
        for (int j = 0; j < 7; j++)
            for (int k = 0; k < 7; k++)
                if (i * k % 7 == j) {
                    value[i][j] = k;
                    break;
                }
}
int gauss(int m, int n) {
    for (int i = 1; i <= n; i++) {
        if (!mat[m][i])
            continue;
        if (trans[i] == -1) {
            trans[i] = m;
            return i;
        }
        int t = trans[i];
        int fac = value[mat[t][i]][mat[m][i]];
        for (int j = i; j <= n; j++) {
            mat[m][j] -= mat[t][j] * fac;
            //mat[m][j]=mat[m][j]*mat[t][i]-fac*mat[t][j];
            mat[m][j] %= 7;
```

```
                mat[m][j] += 7;
                mat[m][j] %= 7;
            }
            b[m] -= fac * b[t];
            b[m] %= 7;
            b[m] += 7;
            b[m] %= 7;
        }
        return -1;
}
int getNum(char* s) {
    for (int i = 0; i < 7; i++) {
        if (strcmp(str[i], s) == 0)
            return i;
    }
}
void init() {
    scanf("%d", &N);
    memset(trans, -1, sizeof(trans));
}
void work() {
    char cmd[10];
    for (int i = 1; i <= N; i++) {
        scanf("%s", cmd);
        if (*cmd == 'A')
            M++;
        else if (*cmd == 'L') {
            for (int j = 1; j <= M; j++) {
                scanf("%d", mat[C] + j);
                mat[C][j] %= 7;
            }
            char p[10], q[10];
            scanf("%s%s", p, q);
            if (crazy)
                continue;
            b[C] = (getNum(q) - getNum(p) + 7) % 7;
            int state = gauss(C, M);
            if (state != -1) {
                C++;
            } else if (b[C])
                crazy = 1;
        } else {
            for (int j = 1; j <= M; j++) {
                scanf("%d", mat[C] + j);
                mat[C][j] %= 7;
            }
            char p[10];
            scanf("%s", p);
            if (crazy) {
                puts("Already crazy");
                continue;
            }
            b[C] = 0;
            int num = getNum(p);
            int state = gauss(C, M);
            if (state != -1) {
                trans[state] = -1;
                puts("Don't know");
            } else {
                b[C] = 7 - b[C];
                b[C] %= 7;
                b[C] += 7;
                b[C] %= 7;
                puts(str[(num + b[C] + 7) % 7]);
            }
        }
    }
}
```

线性同余方程组

```
int egcd(int a, int b, int &rx, int &ry) {
    if (b == 0) {
        rx = 1;
        ry = 0;
        return a;
    }
    int p, q;
    int d = egcd(b, a % b, p, q);
    rx = q;
    ry = p - a / b * q;
    return d;
}
int solve(int a[], int m[], int n, int &res) {
    if (n == 1)
        return -1;
    int ca = a[1], cm = m[1];
    for (int i = 2; i <= n; i++) {
        int x, y;
        int d = egcd(cm, m[i], x, y);
        int delta = a[i] - ca;
        if (delta % d)
            return -1;
        int t = m[i] / d;
        x = (x * delta / d % t + t) % t;
        ca = x * cm + ca;
        cm = cm * m[i] / d;
        ca = (ca % cm + cm) % cm;
    }
    if (ca == 0)
        ca += cm;
    res = cm;
    return ca;
}
```

欧拉函数

```
int phi(int n) {
    if (n == 1)
        return 0;
    int N = n;
    int mx = sqrt((double) N) + 3;
    long long ans = 1;
    for (int i = 2; i < min(mx, N); i++) {
        if (n == 1)
            break;
        if (n % i == 0) {
            long long tmp = 1;
            while (n % i == 0) {
                n /= i;
                tmp *= i;
            }
            ans *= i - 1;
            ans *= tmp / i;
        }
    }
    if (n != 1)
        ans *= n - 1;
    return (int) ans;
}
```

筛法求欧拉函数

```
void buildPrime() {
    for (int i = 2; i <= 1000000; i++) {
        if (!vis[i]) {
            for (int j = i; j <= 1000000; j += i) {
                vis[j] = 1;
                int n = j;
                if (res[j] == 0)
                    res[j] = 1;
                while (n % i == 0) {
                    res[j] *= i;
                    n /= i;
                }
                res[j] /= i;
                res[j] *= i - 1;
            }
        }
    }
}
```

反素数表 Ural 1748

```
int vis[10010], prime[10010];
int res[10010], P, cnt;
long long N, result;
struct Answer {
    long long value;
    int cnt;

    bool operator<(const Answer& p) const {
        if (value != p.value)
            return value < p.value;
        return cnt < p.cnt;
    }
};
int C;
Answer ans[2000010];
void buildPrime() {
    for (int i = 2; i <= 10000; i++) {
        if (!vis[i]) {
            prime[++P] = i;
            for (int j = 2 * i; j <= 10000; j += i) {
                vis[j] = 1;
            }
        }
    }
}
void DFS(int current, long long value, int sum) {
    if (value <= N) {
        ans[C].value = value;
        ans[C].cnt = sum;
        C++;
        if (cnt < sum || (cnt == sum && result > value)) {
            cnt = sum;
            result = value;
        }
    } else
        return;
    if (res[current] < res[current - 1]) {
        res[current]++;
        if (value <= (double) N / prime[current]) {
            DFS(current, value * prime[current], sum / (res[current])
* (1
                    + res[current]));
            DFS(current + 1, value * prime[current], sum /
(res[current]) * (1
                    + res[current]));
        }
        res[current]--;
    }
    if (res[current + 1] < res[current] && res[current]) {
        res[current + 1]++;
        if (value <= (double) N / prime[current + 1]) {
            DFS(current + 1, value * prime[current + 1], sum * (1 +
res[current
                    + 1]));
        }
        res[current + 1]--;
    }
}
void init() {
    scanf("%lld", &N);
}
Answer queue[2000010];
int head, tail;
void build() {
    memset(res, 0, sizeof(res));
    result = 0;
    cnt = 0;
    res[0] = 0x7FFFFFFF;
    N = 1000000000000000000LL;
    DFS(1, 1, 1);
    sort(ans, ans + C);
    int c = C;
    C = 0;
    for (int i = 0; i < c;) {
        Answer current = ans[i];
        ans[C++] = current;
        int mn = 0;
        for (; i < c && ans[i].value == current.value; i++) {
            mn = max(mn, ans[i].cnt);
        }
        ans[C - 1].cnt = mn;
    }
    head = 0;
    tail = 0;
    for (int i = 0; i < C; i++) {
        if (head != tail && ans[i].cnt <= queue[tail - 1].cnt)
            continue;
        else
            queue[tail++] = ans[i];
    }
}
void work() {
    int low = 0, high = tail - 1, ret = -1;
    while (low <= high) {
        int mid = (low + high) >> 1;
        if (queue[mid].value <= N) {
            ret = mid;
            low = mid + 1;
        } else
            high = mid - 1;
    }
    printf("%lld %d\n", queue[ret].value, queue[ret].cnt);
}
int main() {
    buildPrime();
    build();
    int t;
    scanf("%d", &t);
    for (int i = 1; i <= t; i++) {
        init();
        work();
    }
    return 0;
}
```

高次同余方程 A^B = C mod D

求A

```
//X^K = A mod P
//先求原根g, 然后用babysetp求g的指标使g^p==A. 则A=g^(ik). 那么ik == p mod
phi(P)
typedef long long LL;
LL P, K, A;
LL fastMod(LL a, LL n, LL mod) {
    if (n == 0)
        return 1;
    else if (n == 1)
        return a % mod;
    else if (n == 2)
        return a * a % mod;
    else {
        if (n % 2)
            return fastMod(a, n - 1, mod) * a % mod;
        else
            return fastMod(fastMod(a, n / 2, mod), 2, mod);
    }
}
LL primRoot(LL n) {//暴力原根
    LL g = n - 1;
    for (LL i = 2; i <= n; i++) {
        LL mn = 1LL << 60;
        for (LL j = 1; j * j <= g + 5; j++) {
            if (g % j)
                continue;
            if (fastMod(i, j, P) == 1)
                mn = min(mn, j);
            if (fastMod(i, g / j, P) == 1)
                mn = min(mn, g / j);
        }
        if (mn == g)
            return i;
    }
}
struct NumberPair {
    LL idx, value;
    bool operator<(const NumberPair& p) const {
        return value < p.value;
    }
    bool operator==(const NumberPair& p) const {
        return value == p.value;
    }
};
NumberPair arr[1000010];
int BS(const NumberPair& key, int P) {
    int low = 1, high = P;
    while (low <= high) {
        int mid = (low + high) >> 1;
        if (arr[mid] == key)
            return mid;
        else if (key < arr[mid])
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}
LL gcd(LL a, LL b) {
    if (a < b)
        return gcd(b, a);
    while (b) {
        LL t = a;
        a = b;
        b = t % b;
    }
    return a;
}
LL egcd(LL a, LL b, LL& rx, LL& ry) {
    if (b == 0) {
        rx = 1;
        ry = 0;
        return a;
    }
    LL x, y;
    LL d = egcd(b, a % b, x, y);
    rx = y;
    ry = x - a / b * y;
    return d;
}
LL modLine(LL a, LL b, LL n) {
    LL x, y;
    LL d = egcd(a, n, x, y);
    if (b % d == 0) {
        LL x0 = x * (b / d) % n;
        return (x0 % n + n) % n;
    }
    return -1;
}
LL babyStep(LL X, LL Z, LL K) {
    LL temp = 1;
    for (int i = 0; i <= 100; i++) {
        if (temp == Z)
            return i;
        temp *= X;
        temp %= K;
    }
    LL d = 0, D = 1 % K;
    while ((temp = gcd(X, K)) != 1) {
        if (Z % temp)
            return -1;
        ++d;
        K /= temp;
        Z /= temp;
        D = D * X / temp % K;
    }
    int m = ceil(sqrt((double) K));
    int P = 0;
    LL current = 1 % K;
    for (int i = 0; i <= m; i++) {
        ++P;
        arr[P].idx = i;
        arr[P].value = current;
        current *= X;
        current %= K;
    }
    sort(arr + 1, arr + 1 + P);
    int p = P;
    P = 0;
    for (int i = 1; i <= p;) {
        NumberPair current = arr[i];
        ++P;
        arr[P] = current;
        for (; i <= p && arr[i].value == current.value; i++) {
            if (arr[P].idx > arr[i].idx) {
                arr[P] = arr[i];
            }
        }
    }
    LL tmp = fastMod(X, m, K);
    for (int i = 0; i <= m; i++) {
        LL res = modLine(D, Z, K);
        if (res == -1)
            continue;
        NumberPair p;
        p.value = res;
        int t = BS(p, P);
        if (t != -1) {
            return i * m + arr[t].idx + d;
        }
        D = D * tmp % K;
    }
    return -1;
}
LL result[1000010];
void work() {
    A %= P;
    if (A == 0) {
        puts("1");
        puts("0");
        return;
    }
    int R = 0;
    LL g = primRoot(P);
    LL p = babyStep(g, A, P);
    LL x, y;
    LL d = egcd(K, P - 1, x, y);
    if (p % d == 0) {
        LL x0 = x * (p / d) % (P - 1);
        for (int i = 0; i < d; i++) {
            result[R++] = ((x0 + i * ((P - 1) / d)) % (P - 1) + P - 1)
                    % (P - 1);
        }
        for (int i = 0; i < R; i++) {
            result[i] = fastMod(g, result[i], P);
        }
        sort(result, result + R);
        int r = R;
        R = 0;
        for (int i = 0; i < r;) {
            LL current = result[i];
            result[R++] = current;
            for (; i < r && current == result[i]; i++)
                ;
        }
        cout << R << endl;
        for (int i = 0; i < R; i++) {
            cout << result[i];
            if (i == R - 1)
                cout << endl;
            else
                cout << " ";
        }
    } else
        puts("0");
}
void init() {
    cin >> P >> K >> A;
}
```

求最小B

```
//X^y = Z mod K
typedef long long LL;
struct NumberPair {
    LL idx, value;
    bool operator<(const NumberPair& p) const {
        return value < p.value;
    }
    bool operator==(const NumberPair& p) const {
        return value == p.value;
    }
};
int P;
NumberPair arr[100000];
LL X, Z, K;
int BS(const NumberPair& key) {
    int low = 1, high = P;
    while (low <= high) {
        int mid = (low + high) >> 1;
        if (arr[mid] == key)
            return mid;
        else if (key < arr[mid])
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}
LL fastMod(LL a, int n) {
    if (n == 0)
        return 1;
    else if (n == 1)
        return a % K;
    else if (n == 2)
        return a * a % K;
    else {
        if (n % 2 == 0)
            return fastMod(fastMod(a, n / 2), 2);
        return fastMod(a, n - 1) * a % K;
    }
}
LL gcd(LL a, LL b) {
    if (a < b)
        return gcd(b, a);
    while (b) {
        LL t = a;
        a = b;
        b = t % b;
    }
    return a;
}
LL egcd(LL a, LL b, LL& rx, LL& ry) {
    if (b == 0) {
        rx = 1;
        ry = 0;
        return a;
    }
    LL x, y;
    LL d = egcd(b, a % b, x, y);
    rx = y;
    ry = x - a / b * y;
    return d;
}
LL modLine(LL a, LL b, LL n) {
    LL x, y;
    LL d = egcd(a, n, x, y);
    if (b % d == 0) {
        LL x0 = x * (b / d) % n;
        return (x0 % n + n) % n;
    }
    return -1;
}
LL solve() {
    LL temp = 1;
    for (int i = 0; i <= 100; i++) {
        if (temp == Z)
            return i;
        temp *= X;
        temp %= K;
    }
    LL d = 0, D = 1 % K;
    while ((temp = gcd(X, K)) != 1) {
        if (Z % temp)
            return -1;
        ++d;
        K /= temp;
        Z /= temp;
        D = D * X / temp % K;
    }
    int m = ceil(sqrt((double) K));
    P = 0;
    LL current = 1 % K;
    for (int i = 0; i <= m; i++) {
        ++P;
        arr[P].idx = i;
        arr[P].value = current;
        current *= X;
        current %= K;
    }
    sort(arr + 1, arr + 1 + P);
    int p = P;
    P = 0;
    for (int i = 1; i <= p;) {
        NumberPair current = arr[i];
        ++P;
        arr[P] = current;
        for (; i <= p && arr[i].value == current.value; i++) {
            if (arr[P].idx > arr[i].idx) {
                arr[P] = arr[i];
            }
        }
    }
    LL tmp = fastMod(X, m);
    for (int i = 0; i <= m; i++) {
        LL res = modLine(D, Z, K);
        if (res == -1)
            continue;
        NumberPair p;
        p.value = res;
        int t = BS(p);
        if (t != -1) {
            return i * m + arr[t].idx + d;
        }
        D = D * tmp % K;
    }
    return -1;
}
void work() {
    if (Z >= K) {
        puts("Orz,I can't find D!");
        return;
    }
    LL res = solve();
    if (res < 0)
        puts("Orz,I can't find D!");
    else
        cout << res << endl;
}
```

求所有B_循环节 ZOJ 3254

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <iostream>
#include <algorithm>
#include <vector>
#include <set>

using namespace std;

//X^y = Z mod K, 0 <= y <= M
typedef long long LL;
struct NumberPair {
    LL idx, value;
    inline bool operator<(const NumberPair& p) const {
        return value < p.value;
    }
    inline bool operator==(const NumberPair& p) const {
        return value == p.value;
    }
};
int P;
NumberPair arr[100000];
LL X, Z, K, M;
```

```
int BS(const NumberPair& key) {
    int low = 1, high = P;
    while (low <= high) {
        int mid = (low + high) >> 1;
        if (arr[mid] == key)
            return mid;
        else if (key < arr[mid])
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}
LL fastMod(LL a, LL n) {
    if (K == 0)
        while (true)
            puts("FUCK");
    LL ret = 1 % K;
    while (n) {
        if (n & 1)
            ret = ret * a % K;
        a = a * a % K;
        n >>= 1;
    }
    return ret;
}
LL gcd(LL a, LL b) {
    if (a < b)
        return gcd(b, a);
    while (b) {
        LL t = a;
        a = b;
        b = t % b;
    }
    return a;
}
LL egcd(LL a, LL b, LL& rx, LL& ry) {
    if (b == 0) {
        rx = 1;
        ry = 0;
        return a;
    }
    LL x, y;
    LL d = egcd(b, a % b, x, y);
    rx = y;
    ry = x - a / b * y;
    return d;
}
LL modLine(LL a, LL b, LL n) {
    LL x, y;
    LL d = egcd(a, n, x, y);
    if (b % d == 0) {
        LL x0 = x * (b / d) % n;
        return (x0 % n + n) % n;
    }
    return -1;
}
LL solve() {
    LL temp = 1;
    for (int i = 0; i <= 100; i++) {
        if (temp == Z)
            return i;
        temp *= X;
        temp %= K;
    }
    LL d = 0, D = 1 % K;
    while ((temp = gcd(X, K)) != 1) {
        if (Z % temp) {
            return -1;
        }
        ++d;
        K /= temp;
        Z /= temp;
        D = D * X / temp % K;
    }
    int m = ceil(sqrt((double) K));
    P = 0;
    LL current = 1 % K;
    for (int i = 0; i <= m; i++) {
        ++P;
        arr[P].idx = i;
        arr[P].value = current;
        current *= X;
        current %= K;
    }
    sort(arr + 1, arr + 1 + P);
    int p = P;
    P = 0;
    for (int i = 1; i <= p;) {
        NumberPair current = arr[i];
        ++P;
        arr[P] = current;
        for (; i <= p && arr[i].value == current.value; i++) {
            if (arr[P].idx > arr[i].idx) {
                arr[P] = arr[i];
            }
        }
    }
    LL tmp = fastMod(X, m);
    for (int i = 0; i <= m; i++) {
        //  LL res=D;
        LL res = modLine(D, Z, K);
        if (res == -1)
            continue;
        NumberPair p;
        p.value = res;
        int t = BS(p);
        if (t != -1) {
            return i * m + arr[t].idx + d;
        }
        D = D * tmp % K;
    }
    return -1;
}
LL phi(LL n) {
    //if (n==1) return 0;
    LL N = n;
    LL mx = sqrt((double) N) + 3;
    LL ans = 1;
    for (LL i = 2; i < min(mx, N); i++) {
        if (n == 1)
            break;
        if (n % i == 0) {
            LL tmp = 1;
            while (n % i == 0) {
                n /= i;
                tmp *= i;
            }
            ans *= i - 1;
            ans *= tmp / i;
        }
    }
    if (n != 1)
        ans *= n - 1;
    return ans;
}
void work() {
    LL X1 = X, K1 = K, Z1 = Z;
    LL res = solve();
    if (res == -1) {
        puts("0");
        return;
    }
    if (res > M) {
        puts("0");
        return;
    }
    X = X1;
    K = K1;
    Z = Z1;
    if (K == 1) {
        cout << (unsigned long long) M + 1 << endl;
        return;
    }
    LL p = phi(K);
    if (fastMod(X, res + p) != Z) {
        puts("1");
        return;
    }
    LL mx = 1LL << 62;
    //找循环节
    for (LL i = 1; i * i <= p + 5; i++) {
        if (p % i)
            continue;

        if (fastMod(X, i + res) == Z)
            mx = min(mx, (LL) i);

        if (fastMod(X, p / i + res) == Z)
            mx = min(mx, (LL) p / i);
    }
    {
        unsigned long long r = M - res;
        r /= mx;
        cout << r + 1 << endl;
    }
}
int main() {
    while (scanf("%lld%lld%lld%lld", &X, &K, &Z, &M) != EOF) {
        X %= K;
        Z %= K;
        work();
    }
    return 0;
}
```

求C

```
//A^source = X mod C
int phi(int n) {
    if (n == 1)
        return 0;
    int N = n;
    int mx = sqrt((double) N) + 3;
    long long ans = 1;
    for (int i = 2; i < min(mx, N); i++) {
        if (n == 1)
            break;
        if (n % i == 0) {
            long long tmp = 1;
            while (n % i == 0) {
                n /= i;
                tmp *= i;
            }
            ans *= i - 1;
            ans *= tmp / i;
        }
    }
    if (n != 1)
        ans *= n - 1;
    return (int) ans;
}
int A, B, C;
char source[1000010];
int fastPow(int a, int n) {
    if (n == 0)
        return 1;
    else if (n == 1)
        return a % C;
    else {
        if (n % 2 == 0) {
            long long t = fastPow(a, n / 2);
            t *= t;
            return (int) (t % C);
        } else {
            long long t = fastPow(a, n - 1);
            t *= a;
            return (int) (t % C);
        }
    }
}
void work() {
    int ph = phi(C);
    long long pw = 1;
    int len = strlen(source);
    long long ans = 0;
    for (int i = len - 1; i >= 0; i--) {
        ans += pw * (source[i] - '0') % ph;
        ans %= ph;
        pw *= 10;
        pw %= ph;
    }
    printf("%d\n", fastPow(A, (int) ans + ph));
}
```

阶乘分解式

```
//求若干C(N,K)的GCD
int T;
int vis[100010], primes[100010], C;
void buildPrime() {
    for (int i = 2; i <= 100000; i++) {
        if (!vis[i]) {
            primes[++C] = i;
            for (int j = 2 * i; j <= 100000; j += i) {
                vis[j] = 1;
            }
        }
    }
}
int res[10010];
int getFac(int n, int div) {
    int ans = 0;
    n /= div;
    while (n) {
        ans += n;
        n /= div;
    }
    return ans;
}
void work() {
    memset(res, -1, sizeof(res));
    int mn = C;
    for (int j = 1; j <= T; j++) {
        int m, n;
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= mn; i++) {
            if (primes[i] > n) {
                mn = min(mn, i - 1);
                break;
            }
            int t = getFac(n, primes[i]) - getFac(m, primes[i]) -
getFac(n - m,
                    primes[i]);
            if (res[i] == -1 || res[i] > t)
                res[i] = t;
        }
    }
    long long ans = 1;
    for (int i = 1; i <= mn; i++) {
        for (int j = 1; j <= res[i]; j++) {
            ans *= primes[i];
        }
    }
    cout << ans << endl;
}
```

阶乘末位非零位 logN

```
int table[] = { 1, 1, 2, 6, 4, 4, 4, 8, 4, 6 };
int f(int n) {
    if (n < 5)
        return table[n];
    int t = table[n % 10] * 6 % 10;
    for (int i = 1, r = n / 5 % 4; i <= r; i++) {
        if (t == 2 || t == 6)
            t += 10;
        t /= 2;
    }
    return f(n / 5) * t % 10;
}
```

阶乘末位非零位 分解

```
long long N;
int RES = 1;
int f2 = 0, f3 = 0, f5 = 0, f7 = 0, f9 = 0;
void getFac(long long n) {
    if (n == 0)
        return;
    long long i;
    for (i = n; i > 0; i /= 5) {
        long long p = i / 10, q = i % 10;
        f3 += p + (q >= 3);
        f5 += p + (q >= 5);
        f7 += p + (q >= 7);
        f9 += p + (q >= 9);
    }
    f2 += n / 2;
    getFac(n / 2);
}
int main() {
    char x[101];
    while (scanf("%s", x) != EOF) {
        int i;
        f2 = 0;f3 = 0;f5 = 0;f7 = 0;f9 = 0;
        long long mul = 1;
        N = 0;
        for (i = strlen(x) - 1; i >= 0; i--) {
            N += (x[i] - '0') * mul;
            mul *= 10;
        }
        getFac(N);
        RES = 1;
        int temp = (f2 - f5) % 4;
        if (temp == 1) RES *= 2;
        else if (temp == 2) RES *= 4;
        else if (temp == 3) RES *= 8;
        else RES *= 6;
        temp = f3 % 4;
        if (temp == 1) RES *= 3;
        else if (temp == 2) RES *= 9;
        else if (temp == 3) RES *= 7;
        else RES *= 1;
        temp = f7 % 4;
        if (temp == 1) RES *= 7;
        else if (temp == 2) RES *= 9;
        else if (temp == 3) RES *= 3;
        else RES *= 1;
        temp = f9 % 2;
        if (temp == 1) RES *= 9;
        else RES *= 1;
        RES %= 10;
        if (N == 1) RES = 1;
        int len = 0;
        int n = N;
        while (n) {
            len++;
            n /= 10;
        }
        for (i = 1; i <= 5 - len; i++)
            printf(" ");
        printf("%lld", N);
        printf(" -> %d\n", RES);
    }
    return 0;
}
```

容斥定理求互素

```
//容斥求不大于M的与N互素的数个数
int B, D, K;
struct Number {
    int value[20];
    int cnt;

    void add(int p) {
        value[cnt++] = p;
    }
};
Number number[100010];
int vis[100010];
int phi[100010];
long long sum[100010];
void buildPrime() {
    for (int i = 2; i <= 100000; i++)
        phi[i] = 1;
    for (int i = 2; i <= 100000; i++) {
        if (!vis[i]) {
            number[i].add(i);
            for (int j = i; j <= 100000; j += i) {
                vis[j] = 1;
                number[j].add(i);
                long long r = 1, t = j;
                while (t % i == 0) {
                    r *= i;
                    t /= i;
                }
                r /= i;
                phi[j] *= r * (i - 1);
            }
        }
    }
    for (int i = 2; i <= 100000; i++)
        sum[i] = sum[i - 1] + phi[i];
}
void init() {
    int a, c;
    scanf("%d%d%d%d%d", &a, &B, &c, &D, &K);
}
int numIndex, M;
long long res;
void DFS(int current, int value, int cnt) {
    if (M / value == 0)
        return;
    if (current == number[numIndex].cnt) {
        if (cnt & 1)
            res -= M / value;
        else
            res += M / value;
        return;
    }
    DFS(current + 1, value, cnt);
    DFS(current + 1, value * number[numIndex].value[current], cnt +
1);
}
void work(int caseNum) {
    if (K == 0) {
        printf("Case %d: 0\n", caseNum);
        return;
    }
    int a = B / K;
    int b = D / K;
    if (a > b)
        swap(a, b);
    long long ans = sum[a];
    for (int i = a + 1; i <= b; i++) {
        res = 0;
        numIndex = i;
        M = a;
        DFS(0, 1, 0);
        ans += res;
    }
    if (a)
        ans++;
    cout << "Case " << caseNum << ": " << ans << endl;
}
```

容斥定理求矩形面积并

```
struct Rect {
    int x1, y1, x2, y2;
    int area;
    Rect intersect(const Rect& p) {
        Rect result;
        result.x1 = max(p.x1, x1);
        result.x2 = min(p.x2, x2);
        result.y1 = max(p.y1, y1);
        result.y2 = min(p.y2, y2);
        result.buildArea();
        return result;
    }
    void buildArea() {
        if (y2 <= y1 || x2 <= x1)
            area = 0;
        else
            area = (y2 - y1) * (x2 - x1);
    }
};
Rect rect[25];
int N, M;
int q[25], R;
int res;
void DFS(int current, Rect p, int cnt) {
    if (p.area <= 0)
        return;
    if (current == R) {
        if (cnt == 0)
            return;
        if (cnt & 1)
            res += p.area;
        else
            res -= p.area;
        return;
    }
    DFS(current + 1, p, cnt);
    DFS(current + 1, p.intersect(rect[q[current]]), cnt + 1);
}
void init() {
    for (int i = 1; i <= N; i++) {
        scanf("%d%d%d%d", &rect[i].x1, &rect[i].y1, &rect[i].x2,
&rect[i].y2);
    }
}
void work(int num) {
    printf("Case %d:\n", num);
    for (int i = 1; i <= M; i++) {
        scanf("%d", &R);
        for (int j = 0; j < R; j++) {
            scanf("%d", q + j);
        }
        res = 0;
        Rect tmp;
        tmp.x1 = tmp.y1 = 0;
        tmp.x2 = tmp.y2 = 1010;
        tmp.buildArea();
        DFS(0, tmp, 0);
        printf("Query %d: %d\n", i, res);
    }
}
```

矩阵变换 POJ 3735

```
typedef long long ULL;
int N, M, K;
struct Matrix {
    ULL mat[110][110];
    int m, n;
    void clear() {
        memset(mat, 0, sizeof(mat));
    }
    void init() {
        for (int i = 1; i <= N + 1; i++) {
            mat[i][i] = 1;
        }
    }
    inline Matrix operator*(const Matrix& p) const {
        Matrix res;
        res.clear();
        res.m = m;
        res.n = p.n;
        for (int i = 1; i <= m; i++) {
            for (int k = 1; k <= n; k++) {
                if (mat[i][k]) {
                    for (int j = 1; j <= p.n; j++)
                        res.mat[i][j] += mat[i][k] * p.mat[k][j];
                }
            }
        }
        return res;
    }
};
Matrix fastMult(const Matrix& a, int n) {
    if (n == 1)
        return a;
    else if (n == 2) {
        return a * a;
    } else {
        if (n % 2 == 0) {
            return fastMult(fastMult(a, n / 2), 2);
        } else {
            Matrix tmp = fastMult(a, n - 1);
            return tmp * a;
        }
    }
}
```

```
Matrix ope, source;
int order[110], value[110];
void init() {
    ope.clear();
    ope.m = ope.n = N + 1;
    source.clear();
    source.m = N + 1;
    source.n = 1;
    char tmp[10];
    for (int i = 1; i <= N; i++) {
        order[i] = i;
        value[i] = 0;
        ope.mat[i][i] = 1;
    }
    ope.mat[N + 1][N + 1] = 1;
    source.mat[N + 1][1] = 1;
    for (int i = 1; i <= K; i++) {
        int p, q;
        scanf("%s", tmp);
        if (*tmp == 'g')//加一
        {
            scanf("%d", &p);
            ope.mat[p][N + 1]++;
        } else if (*tmp == 'e')//清零
        {
            scanf("%d", &p);
            for (int j = 1; j <= N + 1; j++)
                ope.mat[p][j] = 0;
        } else//交换
        {
            scanf("%d%d", &p, &q);
            for (int j = 1; j <= N + 1; j++) {
                swap(ope.mat[p][j], ope.mat[q][j]);
            }
        }
    }
}
void work() {
    if (M == 0) {
        for (int i = 1; i <= N; i++) {
            printf("0");
            if (i == N)
                puts("");
            else
                putchar(' ');
        }
        return;
    }
    Matrix res = fastMult(ope, M) * source;
    for (int i = 1; i <= N; i++) {
        cout << res.mat[i][1];
        if (i == N)
            cout << endl;
        else
            cout << " ";
    }
}
```

伯恩塞得引理, 最大公约数优化

```
int N, M, K;
int mat[11][11];
int unit[11][11];
const int P = 9973;
void buildUnit() {
    for (int i = 0; i <= 10; i++) {
        unit[i][i] = 1;
    }
}
void matMult(int a[][11], int b[][11], int res[][11]) {
    for (int i = 1; i <= M; i++) {
        for (int j = 1; j <= M; j++) {
            res[i][j] = 0;
            for (int k = 1; k <= M; k++) {
                res[i][j] += a[i][k] * b[k][j];
            }
            res[i][j] %= P;
        }
    }
}
void fastMult(int a[][11], int n, int res[][11]) {
    if (n == 0) {
        memcpy(res, unit, sizeof(unit));
    } else if (n == 1)
        memcpy(res, a, sizeof(mat));
    else if (n == 2)
        matMult(a, a, res);
    else {
        if (n & 1) {
            int tmp[11][11];
            fastMult(a, n - 1, tmp);
            matMult(tmp, a, res);
        } else {
            int tmp[11][11];
            fastMult(a, n / 2, tmp);
            matMult(tmp, tmp, res);
        }
    }
}
int fac[11][11];
void build() {
    for (int i = 1; i <= M; i++) {
        for (int j = 1; j <= M; j++) {
            fac[i][j] = !mat[i][j];
        }
    }
}
int getSolution(int len) {
    if (len == 0)
        return 1;
    int result = 0;
    int source[11];
    int tmp[11][11];
    fastMult(fac, len, tmp);
    for (int i = 1; i <= M; i++) {
        result += tmp[i][i];
    }
    return result % P;
}
int primes[10010], cnt[10010], C;
void buildPrime() {
    C = 0;
    int mx = sqrt((double) N) + 3;
    mx = min(N, mx);
    int n = N;
    for (int i = 2; i < mx; i++) {
        if (n == 1)
            break;
        if (n % i == 0) {
            primes[++C] = i;
            cnt[C] = 0;
            while (n % i == 0) {
                n /= i;
                cnt[C]++;
            }
        }
    }
    if (n != 1) {
        primes[++C] = n;
        cnt[C] = 1;
    }
}
int ans;
void DFS(int current, int value, int euler) {
    if (current == C + 1) {
        ans += euler % P * getSolution(N / value) % P;
        ans %= P;
        return;
    }
    DFS(current + 1, value, euler);
    int tmp = primes[current];
    for (int i = 1; i <= cnt[current]; i++) {
        DFS(current + 1, value * tmp, euler * (primes[current] - 1) *
(tmp
                / primes[current]));
        tmp *= primes[current];
    }
}
int egcd(int a, int b, int &rx, int &ry) {
    if (b == 0) {
        rx = 1;
        ry = 0;
        return a;
    }
    int x, y, d;
    d = egcd(b, a % b, x, y);
    rx = y;
    ry = x - a / b * y;
    return d;
}
int solve(int a, int b, int n) {
    int x, y, d;
    d = egcd(a, n, x, y);
    if (b % d == 0) {
        int x0 = (x * (b / d) % n + n) % n;
        return x0 % n;
    }
}
void init() {
    scanf("%d%d%d", &N, &M, &K);
    memset(mat, 0, sizeof(mat));
    for (int i = 1; i <= K; i++) {
        int t, s;
        scanf("%d%d", &t, &s);
        mat[t][s] = 1;
        mat[s][t] = 1;
    }
    build();
    buildPrime();
}
void work() {
    ans = 0;
    DFS(1, 1, 1);
    int t = solve(N, ans, P);
    printf("%d\n", t % P);
}
```

约瑟夫问题

```
//N人初始在M间隔K报数
int N, K, M;
int f(int n, int k) {
    if (n == 1)
        return 0;
    return (k + f(n - 1, k)) % n;
}
int main() {
    scanf("%d%d%d", &N, &K, &M);
    while (N || K || M) {
        printf("%d\n", (f(N - 1, K) + M) % N + 1);
        scanf("%d%d%d", &N, &K, &M);
    }
    return 0;
}
```