高斯消元

Gauss-Jordan Elimination

Complexity：$O(n^3)$

Code（浮点）：

```
namespace LA{
    int n;
    double a[N][N], b[N];

    void init(int nn){
        n = nn;
        for(int i = 1; i <= n; i++){
            b[i] = 0;
            for(int j = 0; j <= n; j++) a[i][j] = 0;
        }
    }
    bool Gauss(){
    // false: no solution or multiple solutions; true: a[][n+1] is the only solution
    /*  a[1,1]x1 + a[1,2]x2 + ... + a[1,n]xn = b[1]
        a[2,1]x1 + a[2,2]x2 + ... + a[2,n]xn = b[2]
        ...
        a[n,1]x1 + a[n,2]x2 + ... + a[n,n]xn = b[3]  */

        for(int i = 1; i <= n; i++) a[i][n+1] = b[i];
        for(int j = 1; j <= n; j++){
            int r = j;
            for(int i = j + 1; i <= n; i++)
                if(a[i][j] > a[j][j])
                    r = i;
            if(r != j)  swap(a[r], a[j]);
            if(a[j][j] == 0)    return false;
            for(int i = 1; i <= n; i++){
                if(i == j)  continue;
                double div = a[i][j] / a[j][j];
                for(int k = j; k <= n + 1; k++)
                    a[i][k] -= div * a[j][k];
            }
        }
        for(int i = 1; i <= n; i++) a[i][n+1] /= a[i][i];
        return true;
    }
    double det(){ // get determinant
        double res = 1;
        int flag = 1;
        for(int j = 1; j <= n; j++){
            int r = j;
            for(int i = j + 1; i <= n; i++)
                if(a[i][j] > a[j][j])
                    r = i;
            if(r != j)  swap(a[r], a[j]), flag = -flag;
            if(a[j][j] == 0)    return 0;
            for(int i = 1; i <= n; i++){
                if(i == j)  continue;
                double div = a[i][j] / a[j][j];
                for(int k = j; k <= n; k++)
                    a[i][k] -= div * a[j][k];
            }
```

```
        }
        for(int i = 1; i <= n; i++) res *= a[i][i];
        return flag ? res : -res;
    }
}
```

Code（取模）：

```
namespace LA{
    int n;
    LL a[N][N], b[N];

    void init(int nn){
        n = nn;
        for(int i = 1; i <= n; i++){
            b[i] = 0;
            for(int j = 0; j <= n; j++) a[i][j] = 0;
        }
    }
    bool Gauss(){
    // false: no solution or multiple solutions; true: a[][n+1] is the only solution
    /*  a[1,1]x1 + a[1,2]x2 + ... + a[1,n]xn = b[1]
        a[2,1]x1 + a[2,2]x2 + ... + a[2,n]xn = b[2]
        ...
        a[n,1]x1 + a[n,2]x2 + ... + a[n,n]xn = b[3]  */

        for(int i = 1; i <= n; i++) a[i][n+1] = b[i];
        for(int j = 1; j <= n; j++){
            int r = j;
            for(int i = j + 1; i <= n; i++)
                if(a[i][j] > a[j][j])
                    r = i;
            if(r != j)  swap(a[r], a[j]);
            if(a[j][j] == 0)    return false;
            for(int i = 1; i <= n; i++){
                if(i == j)  continue;
                LL div = a[i][j] * fpow(a[j][j], MOD-2) % MOD;
                for(int k = j; k <= n + 1; k++){
                    a[i][k] -= div * a[j][k];
                    ((a[i][k] %= MOD) += MOD) %= MOD;
                }
            }
        }
        for(int i = 1; i <= n; i++) (a[i][n+1] *= fpow(a[i][i], MOD-2)) %= MOD;
        return true;
    }
    LL det(){ // get determinant
        LL res = 1;
        int flag = 1;
        for(int j = 1; j <= n; j++){
            int r = j;
            for(int i = j + 1; i <= n; i++)
                if(a[i][j] > a[j][j])
                    r = i;
            if(r != j)  swap(a[r], a[j]), flag = -flag;
            if(a[j][j] == 0)    return 0;
            for(int i = 1; i <= n; i++){
                if(i == j)  continue;
                LL div = a[i][j] * fpow(a[j][j], MOD-2) % MOD;
                for(int k = j; k <= n; k++){
                    a[i][k] -= div * a[j][k] % MOD;
                    ((a[i][k] %= MOD) += MOD) %= MOD;
                }
            }
        }
```

```
        for(int i = 1; i <= n; i++) (res *= a[i][i]) %= MOD;
        return flag > 0 ? res : MOD - res;
    }
}
```