

高斯消元

Gauss-Jordan Elimination

Complexity: $O(n^3)$

Code (浮点) :

```
1 namespace LA{
2     int n;
3     double a[N][N], b[N];
4
5     void init(int nn){
6         n = nn;
7         for(int i = 1; i <= n; i++){
8             b[i] = 0;
9             for(int j = 0; j <= n; j++) a[i][j] = 0;
10        }
11    }
12    bool Gauss(){
13        // false: no solution or multiple solutions; true: a[][n+1] is the only solution
14        /* a[1,1]x1 + a[1,2]x2 + ... + a[1,n]xn = b[1]
15         a[2,1]x1 + a[2,2]x2 + ... + a[2,n]xn = b[2]
16         ...
17         a[n,1]x1 + a[n,2]x2 + ... + a[n,n]xn = b[n] */
18
19        for(int i = 1; i <= n; i++) a[i][n+1] = b[i];
20        for(int j = 1; j <= n; j++){
21            int r = j;
22            for(int i = j + 1; i <= n; i++)
23                if(a[i][j] > a[j][j])
24                    r = i;
25            if(r != j) swap(a[r], a[j]);
26            if(a[j][j] == 0) return false;
27            for(int i = 1; i <= n; i++){
28                if(i == j) continue;
29                double div = a[i][j] / a[j][j];
30                for(int k = j; k <= n + 1; k++)
31                    a[i][k] -= div * a[j][k];
32            }
33        }
34        for(int i = 1; i <= n; i++) a[i][n+1] /= a[i][i];
35        return true;
36    }
37    double det(){ // get determinant
38        double res = 1;
39        int flag = 1;
40        for(int j = 1; j <= n; j++){
41            int r = j;
42            for(int i = j + 1; i <= n; i++)
43                if(a[i][j] > a[j][j])
44                    r = i;
45            if(r != j) swap(a[r], a[j]), flag = -flag;
46            if(a[j][j] == 0) return 0;
47            for(int i = 1; i <= n; i++){
48                if(i == j) continue;
49                double div = a[i][j] / a[j][j];
50                for(int k = j; k <= n; k++)
51                    a[i][k] -= div * a[j][k];
52            }
53        }
54        for(int i = 1; i <= n; i++) res *= a[i][i];
55        return flag > 0 ? res : -res;
56    }
57 }
```

Code (取模) :

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

```

4
5 void init(int nn){
6     n = nn;
7     for(int i = 1; i <= n; i++){
8         b[i] = 0;
9         for(int j = 0; j <= n; j++) a[i][j] = 0;
10    }
11 }
12 bool Gauss(){
13     // false: no solution or multiple solutions; true: a[][n+1] is the only solution
14     /* a[1,1]x1 + a[1,2]x2 + ... + a[1,n]xn = b[1]
15        a[2,1]x1 + a[2,2]x2 + ... + a[2,n]xn = b[2]
16        ...
17        a[n,1]x1 + a[n,2]x2 + ... + a[n,n]xn = b[n] */
18
19     for(int i = 1; i <= n; i++) a[i][n+1] = b[i];
20     for(int j = 1; j <= n; j++){
21         int r = j;
22         for(int i = j + 1; i <= n; i++)
23             if(a[i][j] > a[j][j])
24                 r = i;
25         if(r != j) swap(a[r], a[j]);
26         if(a[j][j] == 0) return false;
27         for(int i = 1; i <= n; i++){
28             if(i == j) continue;
29             LL div = a[i][j] * fpow(a[j][j], MOD-2) % MOD;
30             for(int k = j; k <= n + 1; k++){
31                 a[i][k] -= div * a[j][k];
32                 ((a[i][k] %= MOD) += MOD) %= MOD;
33             }
34         }
35     }
36     for(int i = 1; i <= n; i++) (a[i][n+1] *= fpow(a[i][i], MOD-2)) %= MOD;
37     return true;
38 }
39 LL det(){ // get determinant
40     LL res = 1;
41     int flag = 1;
42     for(int j = 1; j <= n; j++){
43         int r = j;
44         for(int i = j + 1; i <= n; i++)
45             if(a[i][j] > a[j][j])
46                 r = i;
47         if(r != j) swap(a[r], a[j]), flag = -flag;
48         if(a[j][j] == 0) return 0;
49         for(int i = 1; i <= n; i++){
50             if(i == j) continue;
51             LL div = a[i][j] * fpow(a[j][j], MOD-2) % MOD;
52             for(int k = j; k <= n; k++){
53                 a[i][k] -= div * a[j][k] % MOD;
54                 ((a[i][k] %= MOD) += MOD) %= MOD;
55             }
56         }
57     }
58     for(int i = 1; i <= n; i++) (res *= a[i][i]) %= MOD;
59     return flag > 0 ? res : MOD - res;
60 }
61 }

```