矩阵

Matrix

```
1   struct Matrix{
2       LL ma[N][N];
3       int r, c;
4
5       Matrix(){ r = c = 0; }
6       Matrix(int rr, int cc){
7           r = rr, c = cc;
8           for(int i = 1; i <= r; i++)
9               for(int j = 1; j <= c; j++)
10                  ma[i][j] = 0;
11      }
12      void unit(int n){
13          r = c = n;
14          for(int i = 1; i <= n; i++)
15              for(int j = 1; j <= n; j++)
16                  ma[i][j] = i == j;
17      }
18      Matrix operator + (const Matrix &A){
19          Matrix res(r, c);
20          for(int i = 1; i <= r; i++)
21              for(int j = 1; j <= c; j++)
22                  res.ma[i][j] = (ma[i][j] + A.ma[i][j]) % MOD;
23          return res;
24      }
25      Matrix operator * (const Matrix &A){
26          Matrix res(r, A.c);
27          for(int i = 1; i <= res.r; i++)
28              for(int j = 1; j <= res.c; j++)
29                  for(int k = 1; k <= c; k++)
30                      (res.ma[i][j] += ma[i][k] * A.ma[k][j]) %= MOD;
31          return res;
32      }
33      void print(){
34          for(int i = 1; i <= r; i++){
35              for(int j = 1; j <= c; j++)
36                  printf("%lld ", ma[i][j]);
37              puts("");
38          }
39      }
40  };
41
42  Matrix fpow(Matrix bs, LL idx){
43      Matrix res;
44      res.unit(bs.r);
45      while(idx){
46          if(idx & 1) res = res * bs;
47          idx >>= 1;
48          bs = bs * bs;
49      }
50      return res;
51  }
52  LL fpow(LL bs, LL idx){
53      LL res = 1;
54      bs %= MOD;
55      while(idx){
56          if(idx & 1) (res *= bs) %= MOD;
57          (bs *= bs) %= MOD;
58          idx >>= 1;
59      }
60      return res;
61  }
62
63  bool getInverse(Matrix &A){ // return false: no inverse; true: A is the inverse
64      int n = A.r;
65      Matrix res; res.unit(n);
66      for(int j = 1; j <= n; j++){
67          int r = j;
68          for(int i = j + 1; i <= n; i++)
```

```
            if(A.ma[i][j] > A.ma[j][j])
                r = i;
        if(r != j)  swap(A.ma[r], A.ma[j]), swap(res.ma[r], res.ma[j]);
        if(A.ma[j][j] == 0) return false;
        for(int i = 1; i <= n; i++){
            if(i == j)  continue;
            LL div = A.ma[i][j] * fpow(A.ma[j][j], MOD-2) % MOD;
            for(int k = 1; k <= n; k++){
                A.ma[i][k] -= div * A.ma[j][k] % MOD;
                ((A.ma[i][k] %= MOD) += MOD) %= MOD;
                res.ma[i][k] -= div * res.ma[j][k] % MOD;
                ((res.ma[i][k] %= MOD) += MOD) %= MOD;
            }
        }
    }
    for(int i = 1; i <= n; i++){
        LL inv = fpow(A.ma[i][i], MOD-2);
        for(int j = 1; j <= n; j++)
            (res.ma[i][j] *= inv) %= MOD;
    }
    A = res;
    return true;
}
```