

Lesson3

高斯消元

作用：

- 可以在O(n^3)时间复杂度内求解一个包含n个方程n个未知数的多元线性方程组

解的三种情况

- 唯一解
- 无穷多组解
- 无解

矩阵的3种等价变换操作（初等行列变换，不会影响方程组的解）

- 某一行乘一个非零的数
- 交换某两行
- 把某行的若干倍加到另外一行之上

目的：**将增广矩阵变为上三角矩阵**

解的情况

- 唯一解：完美的阶梯型
- 无解：化简之后，存在方程左边无未知数，右边不为0（0 = 非零）
- 无穷多组解：化简之后，存在0 = 0情况的方程

算法步骤

- 1 枚举每一列c：
- 2 找到该列绝对值最大的一行（出于精度考虑）
- 3 将该行换至当前最上面（没有被换到过）
- 4 将该行的第一个非零数变为1。（初等行变换）
- 5 将下面的所有行的第c列消为0
- 6
- 7 判断解的情况，若有解则从最后一行的方程开始求解

具体实现：时间复杂度O(n^3)

```
1  #include <iostream>
2  #include <algorithm>
3  #include <cmath>
4
5  using namespace std;
6
7  const int N = 110;
8  const double eps = 1e-6;
9
10 int n;
11 double a[N][N];
12
13 int gauss() {
14     int c, r;    //c表示列，r表示行
15
16     for (c=0, r=0; c<n; c++) {
17         int t = r;
18
19         //找出第c列绝对值最大的一行
20         for (int i=r; i<n; i++)
21             if (fabs(a[i][c]) > fabs(a[t][c]))
22                 t = i;
23
24         //如果最大值是0，结束本次循环，跳到下一列
25         if (fabs(a[t][c]) < eps) continue;
26
27         //将该行换至当前最上面
28         for (int i=c; i<n+1; i++) swap(a[t][i], a[r][i]);
29         //将该行第一个非零数化为1，注意行数已经变为r
30         for (int i=n; i>=c; i--) a[r][i] /= a[r][c];
31
32         //将该行之下所有行的第c列消为0
33         for (int i=r+1; i<n; i++)
34             if (fabs(a[i][c]) > eps)
35                 for (int j=n; j>=c; j--)
36                     a[i][j] -= a[r][j]*a[i][c]; //a[i][c]为乘的系数
37
38         r++;
39     }
40 }
```

```

41     if (r < n) {      //无穷多解或者无解
42         for (int i=r; i<n; i++)      //从第r行开始看
43             if (fabs(a[i][n]) > eps)
44                 return 2;    //0 = 非零，无解
45
46         return 1;    // 无穷多组解
47     }
48
49     //若有解则从最后一行的方程开始求解
50     for (int i=n-1; i>=0; i--)    //从最后一行开始
51         for (int j=i+1; j<n; j++)
52             //最终结果为a[i][n]-主元外的所有系数a[i][j]*与系数对应的xj的值即a[j][n]
53             a[i][n] -= a[i][j]*a[j][n];
54
55     return 0;    //唯一解
56 }
57
58 int main(void) {
59     scanf("%d", &n);
60
61     for (int i=0; i<n; i++)    //行
62         for (int j=0; j<n+1; j++)    //列，循环n+1次，a[i][n]存储等号右侧常数
63             scanf("%lf", &a[i][j]); //注意输入格式控制！
64
65     int t = gauss();
66
67     if (t == 0) {    //存在唯一解
68         for (int i=0; i<n; i++) printf("%.2f\n", a[i][n]);
69     }
70     else if (t == 1) puts("Infinite group solutions");
71     else puts("No solution");
72
73     return 0;
74 }

```

- e2: 解异或线性方程组

异或：不进位加法

```

1  #include <iostream>
2  #include <algorithm>
3
4  using namespace std;
5
6  const int N = 110;
7
8  int n;
9  int a[N][N];
10
11 int gauss() {
12     int c, r;
13
14     for (c=0, r=0; c<n; c++) {
15         int t = r;
16
17         for (int i=r; i<n; i++)
18             if (a[i][c]) {
19                 t = i;
20                 break;
21             }
22
23         if (a[t][c] == 0) continue;
24
25         for (int i=0; i<n+1; i++) swap(a[t][i], a[r][i]);
26
27         for (int i=r+1; i<n; i++)
28             if (a[i][c])
29                 for (int j=n; j>=c; j--)
30                     a[i][j] ^= a[r][j];
31         r++;
32     }
33
34     if (r < n) {
35         for (int i=r; i<n; i++)
36             if (a[r][n])
37                 return 2;
38
39         return 1;
40     }
41 }

```

```
42     for (int i=n-1; ~i; i--)
43         for (int j=i+1; j<n; j++)
44             //对非主元的每一项a[i][j]*a[j][n]，等式两边同时^a[i][j]*a[j][n]
45             //从而将该项消零
46             a[i][n] ^= a[i][j]*a[j][n];
47
48     return 0;
49 }
50
51 int main(void) {
52     scanf("%d", &n);
53
54     for (int i=0; i<n; i++)
55         for (int j=0; j<n+1; j++)
56             scanf("%d", &a[i][j]);
57
58     int t = gauss();
59
60     if (!t)
61         for (int i=0; i<n; i++) printf("%d\n", a[i][n]);
62     else if (t == 1) puts("Multiple sets of solutions");
63     else puts("No solution");
64
65     return 0;
66 }
```

组合数

根据数据范围选择使用的算法

求组合数1:

- 范围:

$$1 \leq n \leq 10000$$
$$1 \leq b \leq a \leq 2000$$

- 递推式: $C(a, b) = C(a-1, b) + C(a-1, b-1)$;

证明:

$C(a, b)$ 表示从a个苹果中选出b个的方案数

把所有选法分成两种情况: 包含某一个苹果的选法 $C(a-1, b-1)$ ，不包含某一个苹果的选法 $C(a-1, b)$

具体实现: 将所有组合数预处理出来(递推) 时间复杂度 $O(n^2)$

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 2010, mod = 1e9+7;
6
7  int c[N][N];
8
9  void init() {
10     for (int i=0; i<N; i++)
11         for (int j=0; j<=i; j++)
12             if (!j) c[i][j] = 1;    //当j=0时，进行特判
13             else c[i][j] = (c[i-1][j-1]+c[i-1][j])%mod;
14 }
15
16 int main(void) {
17     init();
18
19     int n;
20     scanf("%d", &n);
21
22     while (n--) {
23         int a, b;
24         scanf("%d%d", &a, &b);
25         printf("%d\n", c[a][b]);
26     }
27
28     return 0;
29 }
```

求组合数2:

- 范围
 - $1 \leq n \leq 10000$,
 - $1 \leq b \leq a \leq 10^5$
- 思路：预处理阶乘
 - $\text{fact}[i] = i! \bmod 1e9+7$
 - $\text{infact}[i] = (i!)^{-1} \bmod 1e9+7$
 - $C(a, b) = \text{fact}[a] * \text{infact}[a-b] * \text{infact}[b]$

具体实现：时间复杂度 $O(n \log n)$

```
1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  const int N = 100010, mod = 1e9+7;
8
9  // fact[i]表示i! % mod, infact[i]表示(i!)^(-1) % mod
10 LL fact[N], infact[N];
11
12 LL qmi(int a, int k, int p) {
13     LL res = 1 % p;
14
15     while (k) {
16         if (k & 1) res = res*a%p;
17         k >>= 1;
18         a = (LL)a*a%p;
19     }
20
21     return res;
22 }
23
24 int main(void) {
25     int n;
26     scanf("%d", &n);
27
28     //预处理fact（阶乘）与infact（阶乘的逆）的数组
29     fact[0] = infact[0] = 1;
30     for (int i=1; i<N; i++) {
31         fact[i] = (LL)fact[i-1]*i%mod;
32         infact[i] = (LL)infact[i-1]*qmi(i, mod-2, mod)%mod;
33     }
34
35     while (n--) {
36         int a, b;
37         scanf("%d%d", &a, &b);
38         printf("%lld\n", fact[a]*infact[b]%mod * infact[a-b]%mod);    //组合数公式
39     }
40
41     return 0;
42 }
```

求组合数3

- 范围
 - $1 \leq n \leq 20$,
 - $1 \leq b \leq a \leq 10^{18}$,
 - $1 \leq p \leq 10^5$
- 思路

卢卡斯(lucas)定理： $C(a, b) = C(a \bmod p, b \bmod p) * C(a / p, b / p) \bmod p$ （等号表示同余）

时间复杂度 $O(\log(p)N * p * \log p) \approx O(p * \log N * \log p)$ ，当N为 10^{18} 级别时， $\log N \approx 64$

但p增大时 $\log(p)N$ 急速减少，所以计算次数为 $10^5 * 20 * 20 \approx 4 * 10^7$

证明：

$a = a_k * p^k + a_{(k-1)} * P^{(k-1)} + \dots + a_0 * p^0$ （将a化为类p进制）

$b = b_k * p^k + b_{(k-1)} * P^{(k-1)} + \dots + b_0 * p^0$ （将b化为类p进制）

$(1+x)^p = C(p, 0) * x^0 + C(p, 1) * x^1 + \dots + C(p, p) * x^p$

且p为质数，故 $C(p, 1), C(p, 2), \dots, C(p, p-1)$ 模p为0，故 $(1+x)^p = 1+x^p \bmod p$ （等号同余）

将a分解

$(1+x)^a = ((1+x)^{a_0}) * (((1+x)^p)^{a_1}) * \dots * (((1+x)^{p^k})^{a_k})$

$= (1+x)^{a_0} * (1+x^p)^{a_1} * \dots * (1+x^{p^k})^{a_k}$

对于 x^b , 等式左边其系数为 $C(a, b)$, 等式右边其系数为 $C(ak, bk) * C(a(k-1), b(k-1)) * \dots * C(a_0, b_0)$

所以左=右即: $C(a, b) = C(ak, bk) * C(a(k-1), b(k-1)) * \dots * C(a_0, b_0) \pmod p$

若存在 $b_i > a_i$, 则 $C(a, b) = 0$

P.S. 最后一步对应没太懂QAQ

Lucas 定理 设 p 是一个素数, 将 m, n 写成 p 进制数:

$$m = a_k p^k + a_{k-1} p^{k-1} + \dots + a_1 p + a_0,$$

$$n = b_k p^k + b_{k-1} p^{k-1} + \dots + b_1 p + b_0, \text{ 其中 } 0 \leq a_i, b_i \leq p (i = 1, 2, \dots, k) . \text{ 则}$$

$$C_m^n \equiv \prod_{i=0}^k C_{a_i}^{b_i} \pmod p .$$

证明 不难看出, 只需证: 当 $m = m_1 p + a_0, n = n_1 p + b_0$ 时, $C_m^n \equiv C_{m_1}^{n_1} C_{a_0}^{b_0} \pmod p$.

$$\text{注意到, } (1+x)^m = \sum_{n=0}^m C_m^n x^n = \sum_{n_1=0}^{m_1} \sum_{b_0=0}^{a_0} C_m^n x^{n_1 p + b_0} ,$$

$$(1+x)^{m_1 p + a_0} = ((1+x)^p)^{m_1} (1+x)^{a_0}$$

$$= \left(\sum_{i=0}^p C_p^i x^i \right)^{m_1} (1+x)^{a_0}$$

$$\equiv (1+x^p)^{m_1} (1+x)^{a_0}$$

$$= \left(\sum_{n_1=0}^{m_1} C_{m_1}^{n_1} x^{n_1 p} \right) \left(\sum_{b_0=0}^{a_0} C_{a_0}^{b_0} x^{b_0} \right)$$

$$\equiv \sum_{n_1=0}^{m_1} \sum_{b_0=0}^{a_0} C_{m_1}^{n_1} C_{a_0}^{b_0} x^{n_1 p + b_0} \pmod p .$$

$$\text{而 } (1+x)^m = (1+x)^{m_1 p + a_0} , \text{ 故 } C_m^n \equiv C_{m_1}^{n_1} C_{a_0}^{b_0} \pmod p .$$

具体实现: $O(\log(n)p \times p \times \log(2)p) = O(p \log(2)n)$

```
1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  //快速幂用于求逆元
8  LL qmi(int a, int k, int p) {
9      LL res = 1%p;
10
11      while (k) {
12          if (k & 1) res = res*a%p;
13          k >>= 1;
14          a = (LL)a*a%p;
15      }
16
17      return res;
18  }
19
20  //求C(a, b, p)
21  int C(int a, int b, int p) {
22      if (b > a) return 0;
23
24      int res = 1;
25      for (int i=1, j=a; i<=b; i++, j--) {
26          res = (LL)res*j%p;
27          res = (LL)res*qmi(i, p-2, p)%p;
28      }
29
30      return res;
31  }
32
```

```
33 //lucas定理
34 int lucas(LL a, LL b, int p) {
35     if (a<p && b<p) return C(a, b, p);
36     // C(a, b) = C(a mod p, b mod p) * C(a / p, b / p) (mod p)
37     return (LL)C(a%p, b%p, p)*lucas(a/p, b/p, p)%p;
38 }
39
40 int main(void) {
41     int n;
42     scanf("%d", &n);
43
44     while (n--) {
45         LL a, b;
46         int p;
47         scanf("%lld%lld%d", &a, &b, &p);
48
49         printf("%d\n", lucas(a, b, p));
50     }
51
52     return 0;
53 }
```

求组合数4

范围

- 输入 a, b,求 C(a, b) 的值
- $1 \leq b \leq a \leq 5000$

思路

- 将C(a, b)分解质因数

$C(a, b) = a! / (b! (a - b)!)$

优化：对于一个质因数P，用分子的指数减去分母的指数即可得最终指数

- **a! 中包含p（质因子）的个数 = $a/p + a/p^2 + \dots + a/p^n$ (当 $p^n > a$ 截止)（/表示下取整）**

证明：

a/p表示1~a中p的倍数个数（从每个p的倍数中拿出一个质因子，可能存在漏网之鱼，例如 p^2 中有两个p，a/p表示只从其中取了一个p）， a/p^2 表示1~a中 p^2 的倍数个数（从每个 p^2 的倍数中继续拿出一个质因子，仍可能存在漏网之鱼，继续重复该过程）..., a/p^n 表示1~a中 p^n 的倍数个数。

从单独每个数角度考虑，若1~a!中某数ai中含有k个质因子p，则在a/p会被计算1次， a/p^2 会被计算1次，... a/p^k 会被计算1次，总共会被计算k次，不重不漏

```
1 //求出n!中某个质因子p的个数
2 int get(int n, int p) {
3     int res = 0;
4     //计算公式 a!中包含p的个数 = a/p + a/p^2 + ... + a/p^n (当a/p^n==0截止)
5     //第一次循环 res += n/p
6     //第二次循环 res += n/p^2
7     //...
8     while (n) {
9         res += n/p;
10        n /= p;
11    }
12
13    return res;
14 }
```

- 实现高精度乘法

具体实现

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 using namespace std;
6
7 typedef long long LL;
8
9 const int N = 5010;
10
11 int primes[N], cnt;
12 bool st[N];
13 int sum[N];
```

```

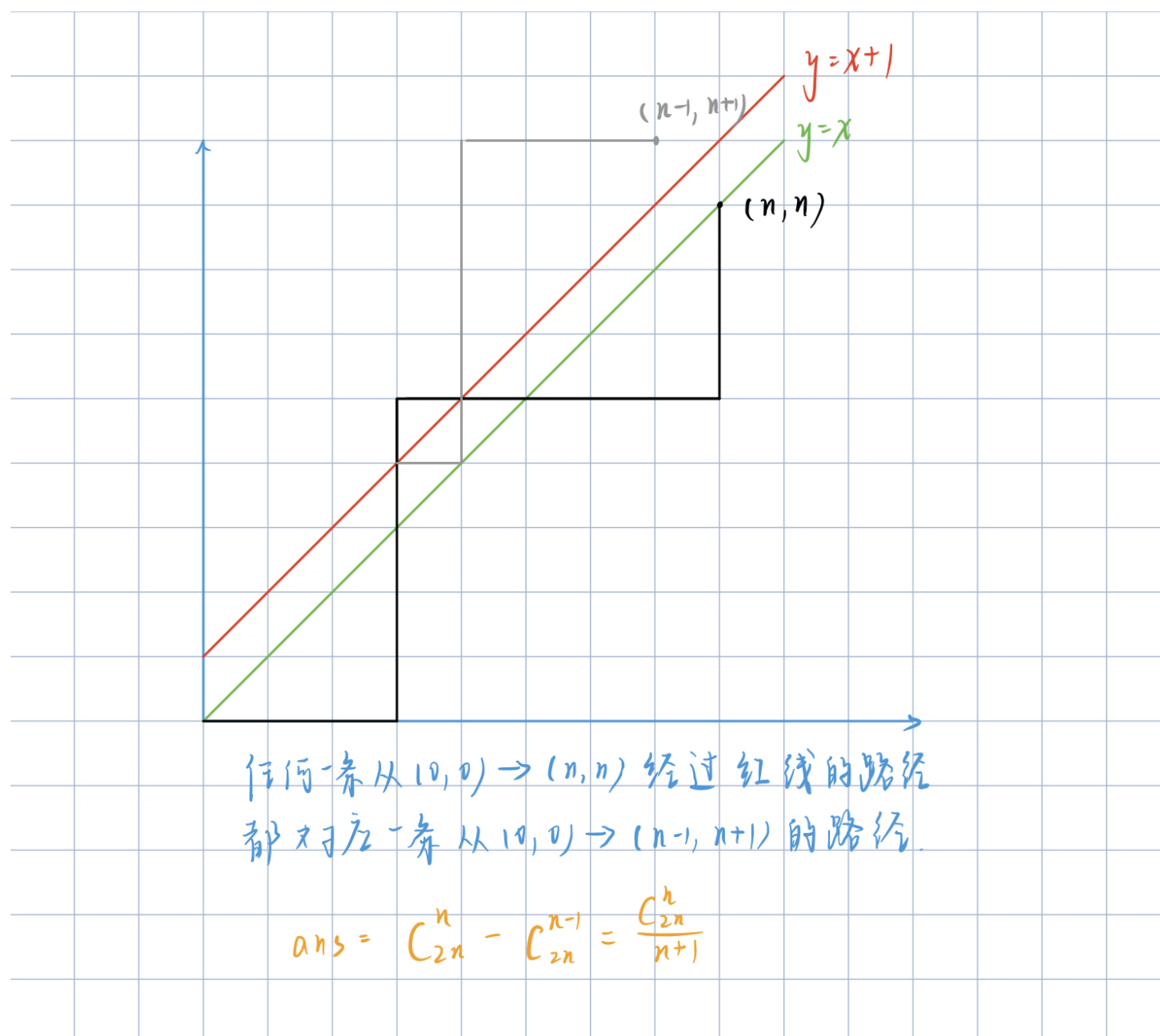
14
15 //线性筛
16 void get_primes(int n) {
17     for (int i=2; i<=n; i++) {
18         if (!st[i])
19             primes[cnt++] = i;
20
21         for (int j=0; primes[j]<=n/i; j++) {
22             st[primes[j]*i] = true;
23             if (i%primes[j] == 0) break;
24         }
25     }
26 }
27
28 //求出n!中某个质因数p的个数
29 int get(int n, int p) {
30     int res = 0;
31     //计算公式 a!中包含p的个数 = a/p + a/p^2 + ... + a/p^n (当a/p^n==0截止)
32     //第一次循环 res += n/p
33     //第二次循环 res += n/p^2
34     //...
35     while (n) {
36         res += n/p;
37         n /= p;
38     }
39
40     return res;
41 }
42
43 //高精度乘法
44 vector<int> mul(vector<int> &A, int b) {
45     vector<int> C;
46
47     for (int i=0, t=0; i<A.size() || t; i++) {
48         if (i < A.size()) t += A[i]*b;
49         C.push_back(t%10);
50         t /= 10;
51     }
52
53     while (C.size()>1 && C.back()==0) C.pop_back();
54
55     return C;
56 }
57
58
59 int main(void) {
60     int a, b;
61     scanf("%d%d", &a, &b);
62
63     get_primes(a);
64
65     //对小于a的每一个质数，处理出其指数
66     for (int i=0; i<cnt; i++) {
67         int p = primes[i];
68         sum[i] = get(a, p) - get(b, p) - get(a-b, p);
69     }
70
71     //保存结果
72     vector<int> res;
73     res.push_back(1);
74
75     //将所有质数相乘
76     for (int i=0; i<cnt; i++)
77         for (int j=0; j<sum[i]; j++)
78             res = mul(res, primes[i]);
79
80     for (int i=res.size()-1; i>=0; i--) printf("%d", res[i]);
81     puts("");
82
83     return 0;
84 }

```

卡特兰数

- e1: 满足条件的01序列
 - $C(2n, n) - C(2n, n-1) = \frac{1}{n+1} C(2n, n)$ (卡特兰数)

证明：等式左半部分画图进行理解证明，参见《ACWing数学知识4》 2:00



```

1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  const int mod = 1e9+7;
8
9  //快速幂
10 LL qmi(int a, int k, int p) {
11     LL res = 1%p;
12
13     while (k) {
14         if (k & 1) res = res*a%p;
15         k >>= 1;
16         a = (LL)a*a%p;
17     }
18
19     return res;
20 }
21
22 int main(void) {
23     int n;
24     scanf("%d", &n);
25
26     int a = 2*n, b = n;
27     int res = 1;
28
29     //计算C(2n, n)
30     for (int i=1, j=a; i<=b; i++, j--) {
31         res = (LL)res*j%mod;
32         res = (LL)res*qmi(i, mod-2, mod)%mod;
33     }
34
35     //计算(1/(n+1))*C(2n, n) (卡特兰数)
36     res = (LL)res*qmi(n+1, mod-2, mod)%mod;
37
38     cout << res;
39
40     return 0;
41 }

```