プログラミングコンテストでの数え上げ＆整数論テクニック

保坂 和宏 (東京大学理学部数学科 3 年)

第 12 回 JOI 春合宿

2013/03/19

「数え上げ」

- 「○○を満たすような□□は存在するか？」→「○○を満たすような□□は何通りあるか？」
  - 解法の正当性が検証しやすくなる
  - アルゴリズムが難しくなることもある
    - 参考：二部グラフの完全マッチングは，存在判定問題は P だが，数え上げ問題は #P 完全 (⇒ NP 困難)
- 主要な道具
  - 漸化式，DP
  - 階乗や二項係数

「整数論」

- 「○○を満たすような□□は何通りあるか？」→「○○を満たすような□□は何通りあるか，その値を m で割った余りを求めよ(mod m で求めよ)」
  - 答が大きくても多倍長整数が不要になる
    - 計算量も答の大きさによらなくなる
  - そのままでは割り算ができなくなるので，整数論の知識が必要

本講義の流れ

- 数学パート
  - 数え上げに関する知識
    - 高校数学範囲とその必須な応用
    - やや高度な内容
  - 整数論に関する知識
    - 証明は省略して重要事実の紹介
- プログラミングコンテストパート
  - 頻出テクニック集
    - C++ コードの紹介や計算量の評価

1. 数え上げに関する知識

順列

- n 種類のものの中から重複なく k 個を選んできて並べる方法は何通り？
- ${}_n\mathrm{P}_k = n\,(n-1)\cdots(n-k+1)$ 通り
  - 1 個目は n 通り，2 個目は $n-1$ 通り，……
- 特に，n 個すべてを並べる方法は $n!$ 通り

- 例：$n = 4, k = 2$
  - { A, B, C, D } から重複なく 2 個選んで並べる方法は，AB, AC, AD, BA, BC, BD, CA, CB, CD, DA, DB, DC の 12 通り

組合せ

- n 種類のものの中から重複なく k 個を選んだ組合せ (順番を無視) の総数は何通り？
- ${}_nC_k = {}_nP_k / k! = n! / (k! (n - k)!)$ 通り
  - 各組合せに対して $k!$ 個の順列が対応

- 例：$n = 4, k = 2$
  - { A, B, C, D } から重複なく 2 個選んだ組合せは，AB, AC, AD, BC, BD, CD の 6 通り

重複順列

- n 種類のものの中から重複を許して k 個を選んできて並べる方法は何通り？
- n^k 通り

- 例：$n = 4, k = 2$
  - { A, B, C, D } から重複を許して 2 個選んで並べる方法は，AA, AB, AC, AD, BA, BB, BC, BD, CA, CB, CC, CD, DA, DB, DC, DD の 16 通り

重複組合せ

- n 種類のものの中から重複を許して k 個を選んだ組合せ (順番を無視) の総数は何通り？
- ${}_nH_k = {}_{n+k-1}C_k$ 通り
  - k 個の印と n - 1 個の「仕切り」の列に対応

    ○○ | ○○○ | | ○

    仕切りで区切られた印の個数に従って選ぶ

- 例：$n = 4, k = 2$
  - { A, B, C, D } から重複を許して 2 個選んだ組合せは，AA, AB, AC, AD, BB, BC, BD, CC, CD, DD の 10 通り

重複組合せ：例

- 以下の条件を満たす整数の組 $(x_1, \ldots, x_n)$ は何通りあるか？
  - $x_i \geqq a_i$
  - $x_1 + \cdots + x_n = s$

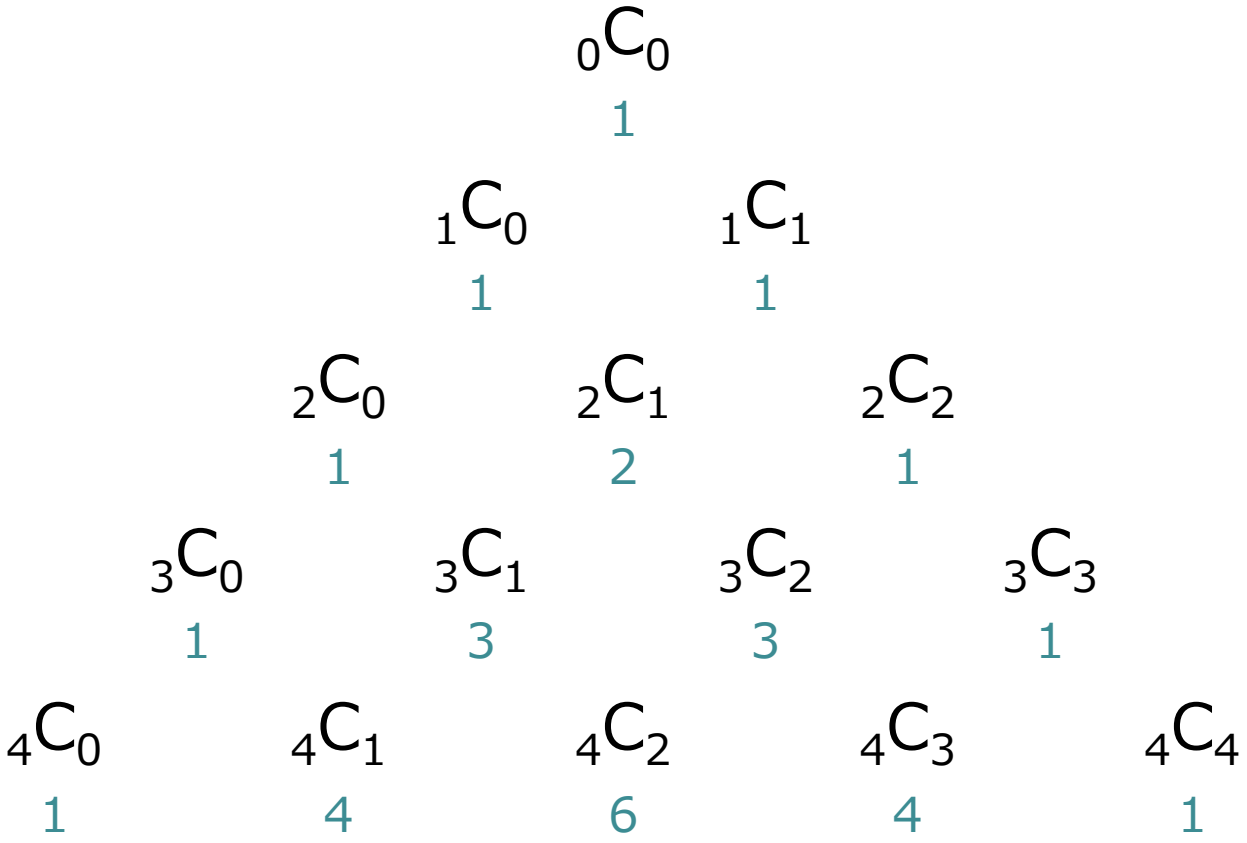
- $x'_i = x_i - a_i$, $s' = s - (a_1 + \cdots + a_n)$ とおくと，$x'_1 + \cdots + x'_n = s'$ を満たす非負整数の組 $(x'_1, \ldots, x'_n)$ を数える問題になる
- ${}_n\mathrm{H}_{s'}$ 通り

二項係数

- $_nC_k = n! / (k! (n - k)!)$ $(0 \leqq k \leqq n)$
  - $n \geqq 0$ であり $k < 0$ または $k > n$ のときは $_nC_k = 0$ であると約束することが多い
  - $\binom{n}{k}$ と書くことも
- 重要な性質
  - $_nC_0 = {}_nC_n = 1$
  - $_nC_k = {}_{n-1}C_{k-1} + {}_{n-1}C_k$ $(n \geqq 1)$
    - ある 1 個を選ぶか選ばないかで場合分け
  - $_nC_k = {}_nC_{n-k}$
    - 「$n - k$ 個選ぶ」は「k 個選ばない」と同じ
  - $_nC_k + {}_nC_1 + \cdots + {}_nC_{n-1} + {}_nC_n = 2^n$
    - 「何個か選ぶ」のは 2^n 通り

二項係数

- 名前の由来：多項式 $(X + Y)^n$ を展開したときの $X^k Y^{n-k}$ の係数が ${}_nC_k$
- Pascal の三角形



多項係数

- 1 を k_1 個，…，m を k_m 個並べた列は何通りあるか？
- $(k_1 + \cdots + k_m)! / (k_1! \cdots k_m!)$ 通り
  - すべての数をそれぞれ区別したとき，各列に対して $(k_1! \cdots k_m!)$ 個の順列が対応
- 多項式 $(X_1 + \cdots + X_m)^n$ を展開したときの ${X_1}^{k1} \cdots {X_m}^{km}$ の係数 $(n = k_1 + \cdots + k_m)$ でもある

- 例：$m = 3, k_1 = 2, k_2 = 2, k_3 = 2$
  - 112233, 112323, 112332, ... の 90 通り

多項係数：例

- いくつかのグループごとに「相対的な順番」が何通りあるかがわかるときに，それらの積に多項係数をかけて全体の並べ方の場合の数が求まる
- 例
  - n 人の生徒がいて，所属するクラスと試験の得点が与えられる
  - n 人を一列に並べる方法であって，同じクラスで得点が異なる生徒は得点が高い生徒の方が前に並ぶ，ようなものは何通りあるか？

包除原理

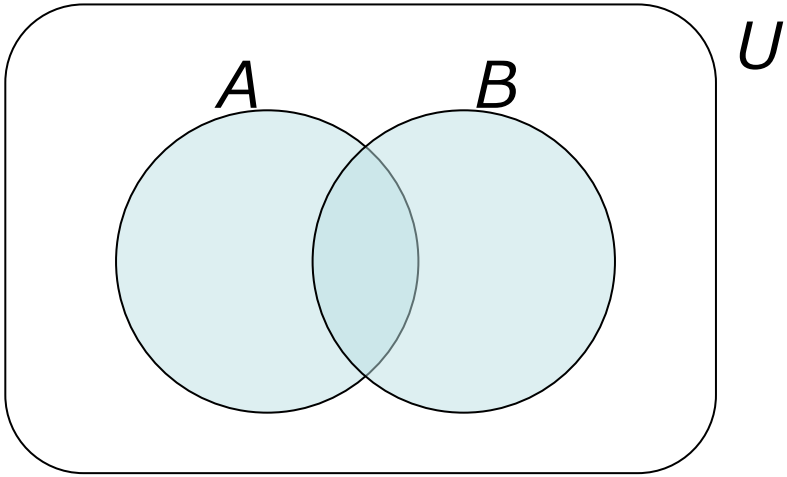
- 有限集合 A, B に対し,

$$|A \cup B| = |A| + |B| - |A \cap B|$$

- 全体集合を U とすると,

$$|A^c \cap B^c| = |U| - |A| - |B| + |A \cap B|$$

  - $|X|$ で集合 X の元 (要素) の個数を表す
  - $X^c = U \setminus X$ を X の補集合とする



包除原理：例

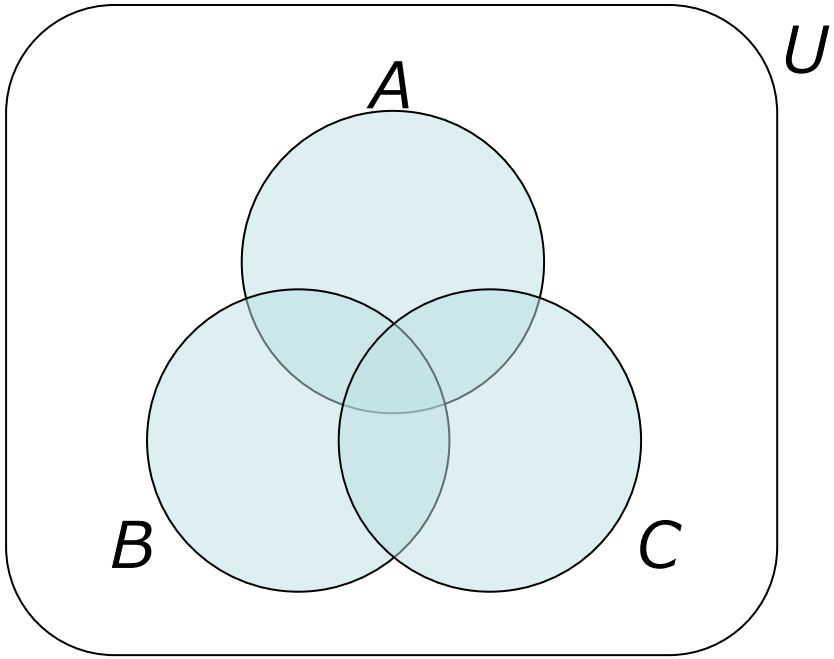
- m 以下の正の整数で，4 でも 6 でも割りきれないものは何個あるか？

- m 以下の正の整数を全体として，そのうち 4 の倍数の集合，6 の倍数の集合を考える
  - 全体で m 個：足す
  - 4 の倍数は $[m / 4]$ 個：引く
  - 6 の倍数は $[m / 6]$ 個：引く
  - 4 の倍数かつ 6 の倍数，すなわち 12 の倍数は $[m / 12]$ 個：足す
- $m - [m / 4] - [m / 6] + [m / 12]$ 個

包除原理

- $A, B, C \subset U$ に対し，

$$|A^c \cap B^c \cap C^c|$$
$$= |U| - |A| - |B| - |C|$$
$$+ |A \cap B| + |A \cap C| + |B \cap C| - |A \cap B \cap C|$$



包除原理

- $A_1, \ldots, A_n \subset U$ に対し，

$$\left| \bigcap_{1 \leqq i \leqq n} {A_i}^{\mathrm{c}} \right| = \sum_{I \subset \{1,\ldots,n\}} (-1)^{|I|} \left| \bigcap_{i \in I} A_i \right|$$

  - $I = \emptyset$ に対応する共通部分は U とする

- 「いくつかの条件を同時に満たすもの」が数えられるとき，「どの条件も満たさないもの」が数えられる
  - n 個の条件のうちのいくつか (p 個とする) を選び (2^n 通り)，それらをすべて満たすものを数え，p が偶数なら足し奇数なら引く

包除原理：例

- m 以下の正の整数で，$d_1, \ldots, d_n$ のいずれでも割りきれないものは何個あるか？

- m 以下の正の整数全体の集合を U，そのうち d_i で割りきれるものの集合を A_i として包除原理を適用
  - 「$d_{i1}, \ldots, d_{ik}$ のいずれでも割りきれるもの」の個数は「$d_{i1}, \ldots, d_{ik}$ の最小公倍数で割りきれるもの」なので求められる

1. 数え上げに関する知識
(ちょっと高度なもの)

第 2 種 Stirling 数

- { 1, ..., k } をちょうど n 個の空でない集合に分割する方法は何通りあるか？

- 例 : $k = 4, n = 2$
  - 以下の 7 通り
    - { { 1 }, { 2, 3, 4 } }
    - { { 2 }, { 1, 3, 4 } }
    - { { 3 }, { 1, 2, 4 } }
    - { { 4 }, { 1, 2, 3 } }
    - { { 1, 2 }, { 3, 4 } }
    - { { 1, 3 }, { 2, 4 } }
    - { { 1, 4 }, { 2, 3 } }

第 2 種 Stirling 数：包除原理

- 分割した後の集合の順番を区別したものを数えて $n!$ で割ることにする
- 整数 1, ..., k を色 1, ..., n のいずれかで塗る方法であって，どの色も 1 回以上使う方法は何通りあるか，を求めればよい

- 「整数 1, ..., k を色 1, ..., n のいずれかで塗る方法」全体の集合を U，そのうち色 i を **1 回も使わない**塗り方の集合を A_i として包除原理を適用

第 2 種 Stirling 数：包除原理

- $|A_{i1} \cap \cdots \cap A_{ip}| = (n - p)^k$
  - 色 $i_1, \ldots, i_p$ のいずれも 1 回も使わない
  - この値は p のみによる
- $p = |I|$ ごとに項をまとめられる

$$\sum_{I \subset \{1,\ldots,n\}} (-1)^{|I|} \left| \bigcap_{i \in I} A_i \right| = \sum_{0 \leqq p \leqq n} (-1)^p {}_n\mathrm{C}_p (n-p)^k$$

- 最初の問題の答は，これを $n!$ で割った値
- $S(k, n)$ などと表す
  - 統一された表記はない様子

第 2 種 Stirling 数：漸化式

- $S(k, 0) = 0$ $(k \geqq 1)$
- $S(k, k) = 1$ $(k \geqq 0)$
- $S(k, n) = S(k - 1, n - 1) + n\, S(k - 1, n)$ $(k \geqq 1)$
  - $\{ k \}$ を単独の集合にして，$\{ 1, \ldots, k - 1 \}$ を $n - 1$ 個の空でない集合に分割する方法が $S(k - 1, n - 1)$ 通り
  - k を単独にしない場合，$\{ 1, \ldots, k - 1 \}$ を n 個の空でない集合に分割してそのいずれかに k を加えるので，$n\, S(k - 1, n)$ 通り

第 2 種 Stirling 数

- $\{ 1, \ldots, k \}$ から $\{ 1, \ldots, n \}$ へ,
  - 写像は n^k 通り
  - 単射は ${}_nP_k$ 通り
  - 全射は $S(k, n)$ 通り
- $S(k, n)$ は X^k を $X\,(X - 1) \cdots (X - n + 1)$ たちの和で書いたときの係数
  - 例 : $X^3 = X + 3\,X\,(X - 1) + X\,(X - 1)\,(X - 2)$
    - $S(3, 0) = 0$, $S(3, 1) = 1$, $S(3, 2) = 3$, $S(3, 3) = 1$
  - 逆に, $X\,(X - 1) \cdots (X - n + 1)$ を展開したときの係数が第 1 種 Stirling 数

漸近評価

- $n! \approx \sqrt{(2\pi)}\, n^{n+1/2}\, e^{-n}$ が知られている (Stirling の公式)

- たとえば ${}_{2n}C_n \approx 4^n / \sqrt{(\pi\, n)}$ が得られる
- 数え上げというよりも計算量の評価に用いることが多い

2. 整数論に関する知識

記法

- 整数 a, b に対し，b が a で割りきれるとき，すなわち c $a = b$ なる整数 c が存在するとき $a \mid b$ と書く
- 整数 a, b, m ($m \neq 0$) に対し，$m \mid a - b$ のとき $a \equiv b \pmod{m}$ と書く
  - 合同式
  - a と b は m を法として合同である，という
  - m で割った余りが同じということ
- $a_1, \ldots, a_n$ の最大公約数，すなわち $d \mid a_1, \ldots, d \mid a_n$ を満たす最大の正の整数 d を $\gcd(a_1, \ldots, a_n)$ と書く
  - ただし $\gcd(0, \ldots, 0) = 0$ とする

合同式の基本性質

- ≡ は同値関係である
  - $a \equiv a \pmod{m}$
  - $a \equiv b \pmod{m} \Rightarrow b \equiv a \pmod{m}$
  - $a \equiv b, b \equiv c \pmod{m} \Rightarrow a \equiv c \pmod{m}$
- $a \equiv b, c \equiv d \pmod{m}$ のとき,
  - $a + c \equiv b + d \pmod{m}$
  - $a - c \equiv b - d \pmod{m}$
  - $a\,c \equiv b\,d \pmod{m}$
- よって，足し算・引き算・掛け算を行う限りにおいては，途中で m で割った余りにおきかえて計算しても mod m での値が求まる

mod m での逆元

- a と m が**互いに素**，つまり $\gcd(a, m) = 1$ のときかつそのときに限り，$ax \equiv 1 \pmod{m}$ なる整数 x が存在し，それは mod m で一意
  - "1 / a" のようなもの
    - 1 / a と書いてしまうこともある
  - mod m での (乗法の) 逆元と呼ばれ，a^{-1} とも書かれる
- 例 : $m = 13, a = -2 \Rightarrow a^{-1} \equiv 6 \pmod{13}$
- 例 : $m = 10, a = 3 \Rightarrow a^{-1} \equiv 7 \pmod{10}$

mod m での逆元

- 割る数が法 m と互いに素である限りは，mod m で割り算を行うことができる
  - $a\,x \equiv b \pmod m$ のとき，両辺に a^{-1} をかけて，$x \equiv a^{-1}\,b \pmod m$
    - $a^{-1}\,b$ も分数で b / a と書いてしまうことも
    - $a / b + c / d \equiv (a\,d + b\,c) / (b\,d) \pmod m$ などが成り立つ
- 例：1274 / 13 を 10 で割った余り
  - $1274 / 13 = 98 \equiv 8 \pmod{10}$
  - $1274 / 13 \equiv 4 / 3 \equiv 3^{-1} \cdot 4 \equiv 7 \cdot 4 \equiv 28 \equiv 8 \pmod{10}$
    - 割り算が入るときも先に mod をとってよい

mod p での逆元

- 法 $m = p$ が素数のときが重要
  - $p \nmid a$ ならば $\gcd(a, p) = 1$ となるので，a^{-1} が存在する
- $p \nmid a$ ならば，$a^{p-1} \equiv 1 \pmod{p}$ (Fermat の小定理)
  - 逆元が $1 / a \equiv a^{p-2} \pmod{p}$ として求まる
- 例：$p = 7, a = 2$
  - $a^{p-1} = 2^6 = 64 \equiv 1 \pmod{7}$
  - $a^{p-2} = 2^5 = 32 \equiv 4 \pmod{7}$, $2 \cdot 4 \equiv 1 \pmod{7}$

二項係数 mod p

- ${}_nC_k = n! / (k! (n - k)!)$

- $n < p$ ならば $k!$ や $(n - k)!$ は p で割れないので，この形で分子分母を先に p で割った余りを求めていても割り算できる
- $k < p$ ならば $k!$ は p で割れないので，${}_nC_k = n (n - 1) \cdots (n - k + 1) / k!$ の形にすれば割り算できる

二項係数 mod p

- $n = n_l\, p^l + \cdots + n_1\, p + n_0$ $(0 \leqq n_i < p)$, $k = k_l\, p^l + \cdots + k_1\, p + k_0$ $(0 \leqq k_i < p)$ のとき，${}_n\mathrm{C}_k \equiv {}_{nl}\mathrm{C}_{kl} \cdots {}_{n1}\mathrm{C}_{k1}\ {}_{n0}\mathrm{C}_{k0} \pmod p$ (Lucas の定理)
  - p 進法で表して，各桁どうしの二項係数の積をとる
    - ${}_0\mathrm{C}_0 = 1$ なので先頭に余分な 0 がついてよい
  - 1 箇所でも $n_i < k_i$ となれば ${}_n\mathrm{C}_k \equiv 0$

- 例：$p = 13$
  - $43 = 3 \cdot 13 + 4$, $15 = 1 \cdot 13 + 2$ より，${}_{43}\mathrm{C}_{15} \equiv {}_3\mathrm{C}_1 \cdot {}_4\mathrm{C}_2 = 3 \cdot 6 \equiv 5 \pmod{13}$

3. 頻出テクニック集

オーバーフロー

- 32-bit 整数型のオーバーフローに注意
  - 数え上げ問題では掛け算をよく行う

- 実行時間がシビアでないならば 64-bit 整数型を使おう
- 本講義では long long を Int と書きます

```
typedef long long Int;
```

/, % 演算子の仕様

- 割り算の商や余りを求める演算子の符号に関する仕様に注意
- C/C++ では,
  - *a* / *m* は 0 に近い方に丸める
  - *a* % *m* は *a* と同じ符号をもつ (または 0)
  - *a* = *m* · (*a* / *m*) + (*a* % *m*) は成り立つ
  - 例 :
    - 14 / 5 = 2, 14 % 5 = 4
    - (-14) / 5 = -2, (-14) % 5 = -4
    - 14 / (-5) = -2, 14 % (-5) = 4
    - (-14) / (-5) = 2, (-14) % (-5) = -4

/, % 演算子の仕様

- $m > 0$ のとき，余りは 0 以上 m 未満にとった方が何かと都合が良いことが多い

```
Int mod(Int a, Int m) {
    return (a % m + m) % m;
}
```

- 割り算を 1 回にして高速化？

```
Int mod(Int a, Int m) {
    a %= m;
    if (a < 0) a += m;
    return a;
}
```

累乗

- (大きい) 非負整数 e に対して a^e を (mod m で) 求めたいことがよくある
- $a^2 = a \cdot a$, $a^4 = a^2 \cdot a^2$, $a^8 = a^4 \cdot a^4$, ... が簡単に求まるので，それらの積として表す
  - e を 2 進法で表す
- 例：$e = 10 = 2^3 + 2^1$ のとき $a^{10} = a^8 \cdot a^2$

累乗

- $O(\log e)$ 時間

```
Int power(Int a, Int e, Int m) {
    Int ret = 1;
    for (; e > 0; e /= 2) {
        if (e % 2 != 0) ret = (ret * a) % m;
        a = (a * a) % m;
    }
    return ret;
}
```

累乗

- 次のように再帰的に考えてもよい
  - $e = 0$ のとき $a^e = 1$
  - $e > 0$ のとき $a^e = a^{e-1} \cdot a$
  - e が偶数のとき $a^e = a^{e/2} \cdot a^{e/2}$
- $O(\log e)$ 時間

```
Int power(Int a, Int e, Int m) {
    if (e == 0) return 1;
    if (e % 2 != 0)
        return (power(a, e - 1, m) * a) % m;
    Int res = power(a, e / 2, m);
    return (res * res) % m;
}
```

mod p での逆元

- $a^{-1} \equiv a^{p-2} \pmod{p}$ を用いれば，$O(\log p)$ 時間で計算できる

mod p での逆元

- 1, 2, ..., n ($n < p$) の mod p での逆元を $O(n)$ 時間で前計算しておくことができる
- $p = i \cdot (p / i) + (p \% i)$ より，$i > 1$ なら

$$0 \equiv i \cdot (p / i) + (p \% i) \pmod{p}$$

$$i \equiv -(p / i)^{-1} \cdot (p \% i) \pmod{p}$$

$$i^{-1} \equiv -(p / i) \cdot (p \% i)^{-1} \pmod{p}$$

```
Int inv[1000010];

inv[1] = 1;
for (int i = 2; i <= N; ++i)
    inv[i] = P - (P / i) * inv[P % i] % P;
```

mod m での割り算

- m が素数でない場合でも，mod m での逆元は Euclid の互除法を用いて $O(\log m)$ 時間で求められる
- 2 や 3 など小さい数での割り算は以下の方法が簡単かつ高速
  - 割る数 b は m と互いに素であるとする
  - $O(b)$ 時間

```
Int divide(Int a, Int b, Int m) {
    for (; ; a += m) {
        if (a % b == 0) return a / b;
    }
}
```

階乗 mod p

- 階乗 $n!$ は，$0! = 1$, $n! = (n - 1)! \cdot n$ を用いて順次求められる
- 階乗の逆元 $(n!)^{-1}$ も，$(0!)^{-1} = 1$, $(n!)^{-1} = ((n - 1)!)^{-1} \cdot n^{-1}$ を用いればよい
- $O(n)$ 時間

```
Int fac[1000010], facInv[1000010];

fac[0] = facInv[0] = 1;
for (int i = 1; i <= N; ++i) {
    fac[i] = (fac[i - 1] * i) % P;
    facInv[i] = (facInv[i - 1] * inv[i]) % P;
}
```

二項係数 mod p

- 階乗とその逆元を準備しておけば，$n < p$ のとき ${}_nC_k$ は毎回 $O(1)$ 時間で計算できる

```
Int choose(int n, int k) {
    if (!(0 <= k && k <= n)) return 0;
    return (((fac[n] * facInv[k]) % P)
                * facInv[n - k]) % P;
}
```

二項係数 mod p

- 一般の場合は Lucas の定理を用いる
- 前計算 $O(p)$ 時間，各回 $O(\log k)$ 時間

```
Int choose(Int n, Int k) {
    if (!(0 <= k && k <= n)) return 0;
    Int ret = 1;
    for (; k > 0; n /= P, k /= P) {
        Int n0 = n % P, k0 = k % P;
        if (n0 < k0) return 0;
        ret = ret * ((((fac[n0] * facInv[k0])
            % P) * facInv[n0 - k0]) % P) % P;
    }
    return ret;
}
```

二項係数 mod *m*

- ${}_nC_0 = {}_nC_n = 1$ と ${}_nC_k = {}_{n-1}C_{k-1} + {}_{n-1}C_k$ を用いて二項係数の値を計算し，2 次元配列に入れる

```
Int C[1010][1010];

for (int n = 0; n <= 1000; ++n) {
    C[n][0] = C[n][n] = 1;
    for (int k = 1; k < n; ++k) {
        C[n][k] =
         (C[n - 1][k - 1] + C[n - 1][k]) % M;
    }
}
```

二項係数：mod をとらない

- ${}_nC_k = n\ (n - 1) \cdots (n - k + 1) / k!$ に従って計算するとオーバーフローしやすい
- ${}_nC_k = (n / 1)\ ((n - 1) / 2) \cdots ((n - k + 1) / k)$ の順に掛け算する
  - 途中までの積は ${}_nC_i$ なのですべて整数
  - $k \leqq n / 2$ なら途中までの積は単調増加
    - そうでなければ k を $n - k$ にすればよい
  - ただしこれでも ${}_nC_k \cdot k$ がオーバーフローする場合は注意

法が素数でない場合

- 「1000000003 で割った余りを求めよ」のように，素数でない法のときに注意
  - よく登場する素数：10007, 1000000007, 1000000009, 100000007, 1234567891
  - factor コマンドで素因数分解してみよう
- 解法の可能性として
  - 割り算を行わないで解ける？
    - m が奇数なら 2 で割るくらいはするかも？
  - より高度な数学的知識が必要？
    - 中国剰余定理など

扱った内容まとめ

- 順列，組合せ，重複順列，重複組合せ
- 二項係数，多項係数
- 包除原理

- 合同式，mod m での逆元
- Lucas の定理

- a^e を $O(\log e)$ 時間で
- 1, ..., n の mod p での逆元を $O(n)$ 時間で

http://hos.ac/slides/