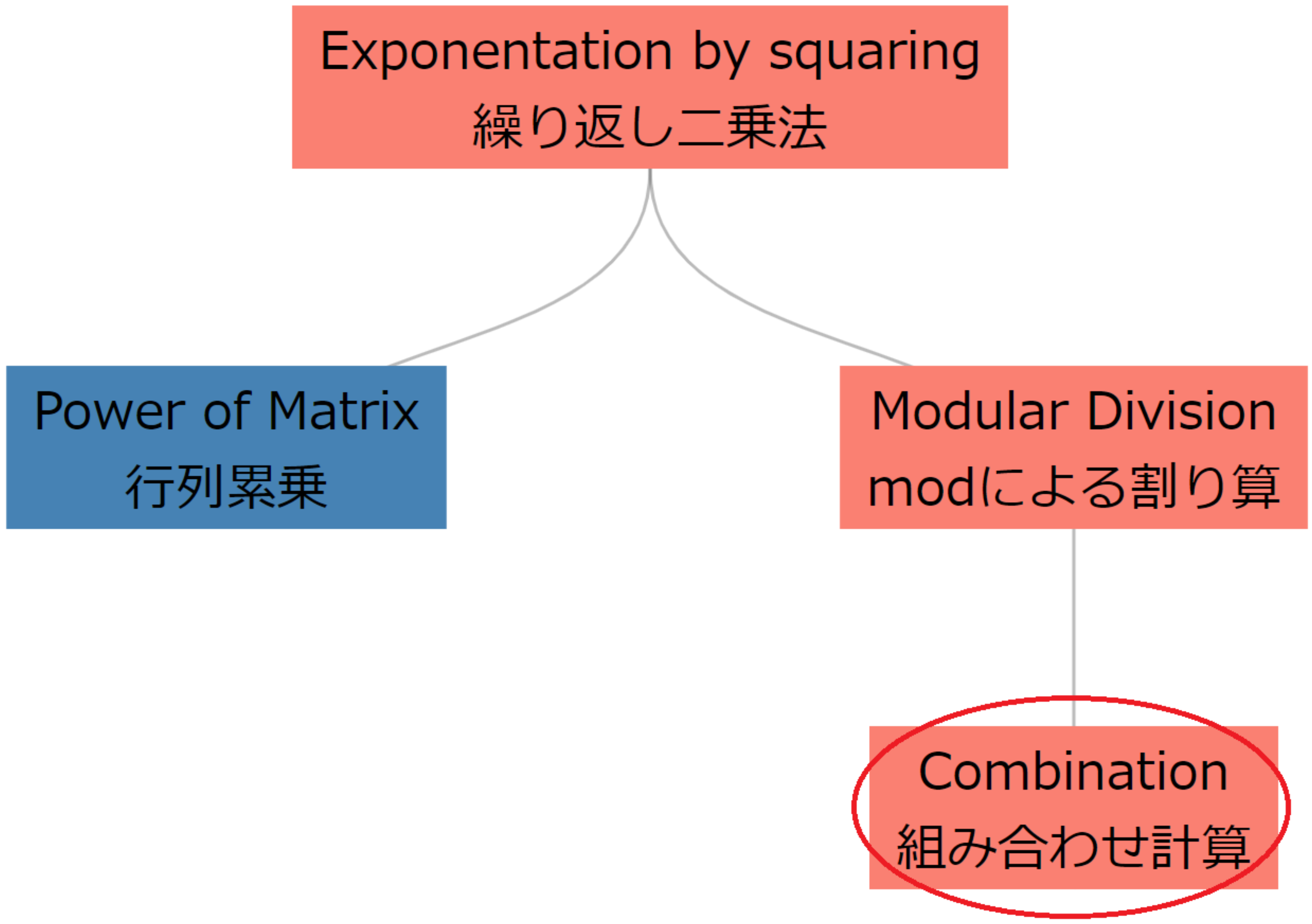
Combination

Relationships



Contents

1. How to calculate ${}_nC_r$ with Pascal's triangle
2. How to calculate ${}_nC_r$ with definition

1. Combination with Pascal's triangle

$$_{n}C_{r} = _{n-1}C_{r-1} + _{n-1}C_{r}$$

1									
1	1								
1	2	1							
1	3	3	1						
1	4	6	4	1					
1	5	10	10	5	1				
1	6	15	20	15	6	1			
1	7	21	35	35	21	7	1		
1	8	28	56	70	56	28	8	1	
1	9	36	84	126	126	84	36	9	1

$_{0}C_{0}$									
$_{1}C_{0}$	$_{1}C_{1}$								
$_{2}C_{0}$	$_{2}C_{1}$	$_{2}C_{2}$							
$_{3}C_{0}$	$_{3}C_{1}$	$_{3}C_{2}$	$_{3}C_{3}$						
$_{4}C_{0}$	$_{4}C_{1}$	$_{4}C_{2}$	$_{4}C_{3}$	$_{4}C_{4}$					
$_{5}C_{0}$	$_{5}C_{1}$	$_{5}C_{2}$	$_{5}C_{3}$	$_{5}C_{4}$	$_{5}C_{5}$				
$_{6}C_{0}$	$_{6}C_{1}$	$_{6}C_{2}$	$_{6}C_{3}$	$_{6}C_{4}$	$_{6}C_{5}$	$_{6}C_{6}$			
$_{7}C_{0}$	$_{7}C_{1}$	$_{7}C_{2}$	$_{7}C_{3}$	$_{7}C_{4}$	$_{7}C_{5}$	$_{7}C_{6}$	$_{7}C_{7}$		
$_{8}C_{0}$	$_{8}C_{1}$	$_{8}C_{2}$	$_{8}C_{3}$	$_{8}C_{4}$	$_{8}C_{5}$	$_{8}C_{6}$	$_{8}C_{7}$	$_{8}C_{8}$	
$_{9}C_{0}$	$_{9}C_{1}$	$_{9}C_{2}$	$_{9}C_{3}$	$_{9}C_{4}$	$_{9}C_{5}$	$_{9}C_{6}$	$_{9}C_{7}$	$_{9}C_{8}$	$_{9}C_{9}$

Implementation

Many combinational problems ask a number modulo 1000000007

```
#define MOD 1000000007
#define MAX_N 1000
long long comb[MAX_N][MAX_N];
void makeComb()
{
  comb[0][0] = 1;
  for (int i = 1; i < MAX_N; i++) {
    for (int j = 0; j <= i; j++) {
      if (j == 0 || j == i) comb[i][j] = 1;
      else comb[i][j] = (comb[i-1][j-1] + comb[i-1][j]) % MOD;
    }
  }
}
// You can use comb[n][r].
```

Complexity

- Array size to make ${}_nC_r$: $n \times n$
  $\Rightarrow$ time complexity to initialize table is $O(n^2)$.
  $\Rightarrow$ time complexity to answer ${}_nC_r$ is $O(1)$.
- $n \le 10^3$: OK.
- $n = 10^4$: maybe MLE (Memory Limit Exceeded)

2. Combination with definition

$$ {}_nC_r = \frac{n!}{r!(n-r)!} $$

- n is big
  ⇒ $n!$ and $r!(n-r)!$: overflow.
  ⇒ need modulo operation
- need to (numer % MOD) and (denom % MOD)
  ⇒ usual divition: ❌
  ⇒ need **the division in modular arthmetics**

Implementation

- It is ok to use this code as library.
- If you want to know *moddiv*, please see the slide "mod"

The code is next page.

```
#define MAX_N 2000000
long long fact[MAX_N];
void factInit() {
  fact[0] = 1;
  for (int i = 1; i < MAX_N; i++) {
    fact[i] = (i * fact[i - 1]) % MOD;
  }
}
// Combination(binomial coefficients)
long long comb(int n, int r) {
  if (n < r || n < 0 || r < 0) return 0;
  return moddiv(fact[n], (fact[r] * fact[n - r]) % MOD);
}
```

Complexity

- The array size to make $n!$: n
  $\Rightarrow$ time complexity to initialize table is $O(n)$.
- On moddiv(a, b), b^{p-2} is calculated.
  On calculation of b^{p-2}, exponentation by squaring is used .
  $\Rightarrow$ time complexity to answer ${}_nC_r$ is $O(\log p)$.
- In sammary, time complexity to answer ${}_nC_r$ k times is $O(n + k \log p)$.

Addition: Permutation

$$ {}_nP_r = \frac{n!}{(n-r)!} $$

```
// Permutation
long long perm(int n, int r) {
  if (n < r || n < 0 || r < 0) return 0;
  return moddiv(fact[n], fact[n - r]);
}
```

- Time complexity to answer ${}_nP_r$ k times is $O(n + k \log p)$.

- If you don't have to answer ${}_nP_r$ many times, bellow is better.

$$ {}_nP_r = n \cdot (n-1) \cdots (n-r-2) \cdot (n-r-1) $$

```
long long perm(int n, int r) {
  long long ret = 1;
  for (int i = n; i > n-r; i--) {
    (ret *= i) %= MOD;
  }
  return ret;
}
```

- Time complexity to answer ${}_nP_r$ one time is $O(n)$.

Addition: Combination with repetitions

$$ {}_nH_r = {}_{n+r-1} C_r $$

- Remember that formula completely: 🙅‍♂️
- Remember as **ball and bar**: 🙆‍♂️

Example

Counting nonnegative integer solutions on linear Diophantine equation.

$$x_1 + x_2 + \cdots + x_n = r$$

$$\bigcirc\bigcirc\bigcirc|\bigcirc\bigcirc|\bigcirc|||\bigcirc|\ \ldots\ |\bigcirc$$

- r balls
- $n-1$ bars
  ⇒ pattern is ${}_{n+r-1}C_r$

Anyway, I wrote code (but I think you don't have to use it).

```
// Combination with repetitions
// (Homogeneous product, Multiset coefficients)
long long homo(int n, int r) {
  if (n == 0 && r == 0) return 1;
  return comb(n + r - 1, r);
}
```

Verify

This: https://yukicoder.me/problems/no/117

Exercises

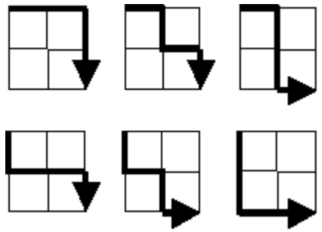
- ABC034C: 経路

  Statement: Japanese only but same as Project Eular No.15

  Lattice paths

  Problem 15

  Starting in the top left corner of a 2×2 grid, and only being able to move to the right and down, there are exactly 6 routes to the bottom right corner.

  How many such routes are there through a $\boxed{H \times W \text{ Grid}}$

  $2 \leqq H \leqq 10^5$
  $2 \leqq W \leqq 10^5$

  Input: H W
  Output: the number of routes modulo 1000000007

- ABC110D: Factorization