写像 12 相を題材にした，基本的な数え上げ問題に対するアプローチ

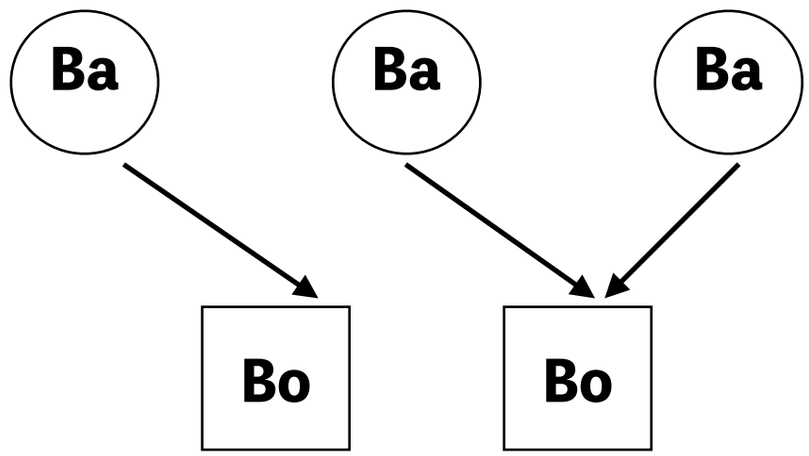
Hokkaido Univ. monkukui

写像 12 相とは

- **写像 12 相:**
  - n 個の玉を k 個の箱に入れる問題の総称
    - e.g. 3 個の区別できるボールを 2 個の区別できる箱へ入れるときの通り数は？？

- どうして写像と呼ばれるか
  - **写像:** 集合の各元を、他の集合の元のいずれかへ対応づけること

- ボールの箱への割り当てと考えると, これは写像



達成目標

- 写像 12 相を題材に, 基本的な数え上げ問題へのアプローチを考える

- AtCoder のレートが上がることが見込める

準備

- 制約は以下で統一されているものとする
  - $1 \leq n \leq 1000, 1 \leq k \leq 1000$

目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する			
区別しない	区別する			
区別する	区別しない			
区別しない	区別しない			

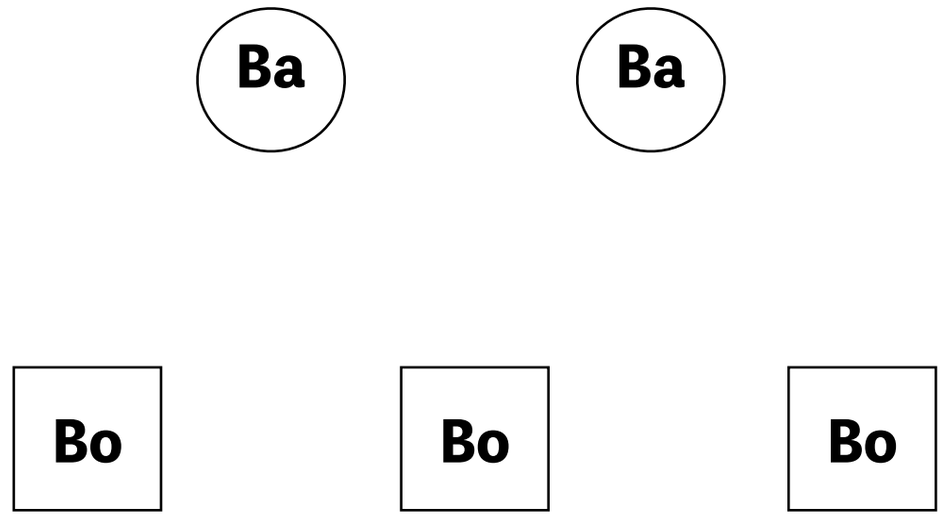
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	ここ		
区別しない	区別する			
区別する	区別しない			
区別しない	区別しない			

Balls and Boxes 1

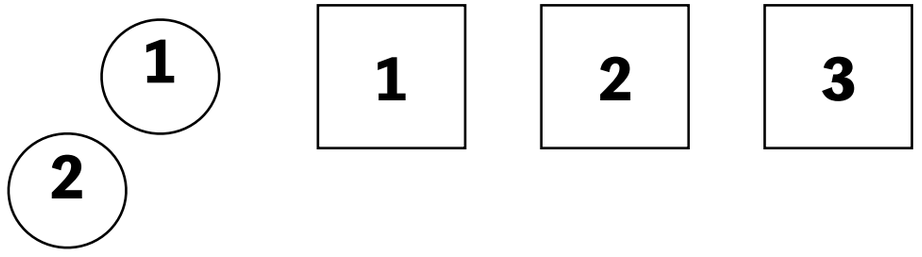
- ［問題］難易度: **茶**
  - n 個の区別できるボールを, k 個の区別できる箱に入れるとき、可能な入れ方の総数を求めよ.
  - ただし、箱に対する制限はないものとする.

- e.g.
  - ［入力］$n = 2, k = 3$
  - ［出力］9



Balls and Boxes 1

- [入力] $n = 2, k = 3$



Balls and Boxes 1

- k^n が答え
- 時間計算量 $\Theta(n)$

```cpp
// ボール：区別あり，箱：区別あり，入れ方：制限なし
#include <iostream>
using namespace std;
const int MOD = 1000000007;

int main() {

  int n, k; cin >> n >> k;
  long long ans = 1;
  for(int i = 0; i < n; i++) {
    ans *= k;
    ans %= MOD;
  }
  cout << ans << endl;
  return 0;
}
```

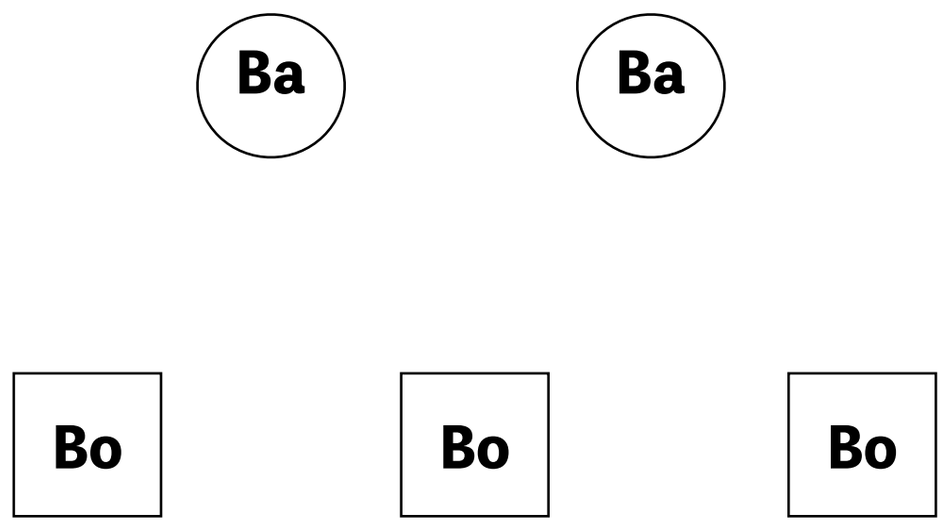
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	ここ	
区別しない	区別する			
区別する	区別しない			
区別しない	区別しない			

Balls and Boxes 2

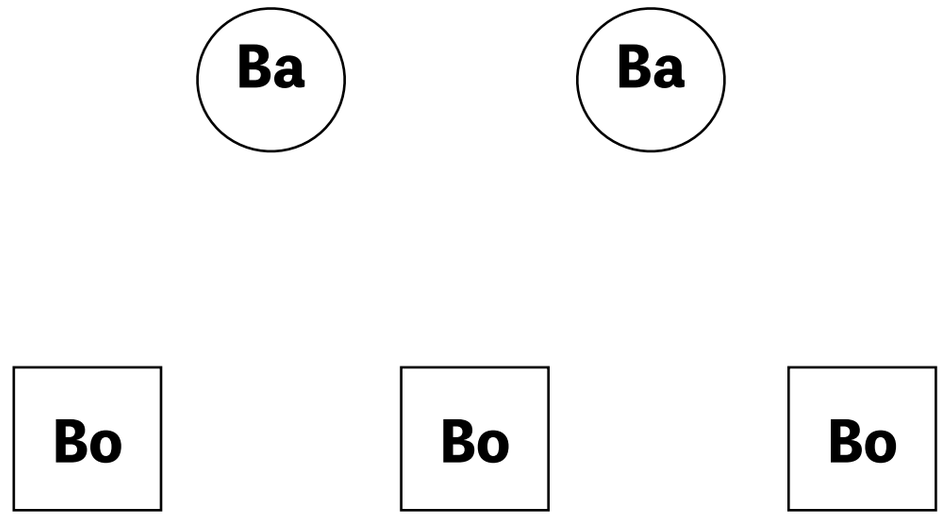
- ［問題］難易度: **茶**
  - n 個の区別できるボールを, k 個の区別できる箱に入れるとき、可能な入れ方の総数を求めよ.
  - 各箱に入れることのできるボールの数は高々 1 個とする.

- e.g.
  - ［入力］$n = 2, k = 3$
  - ［出力］6



Balls and Boxes 2

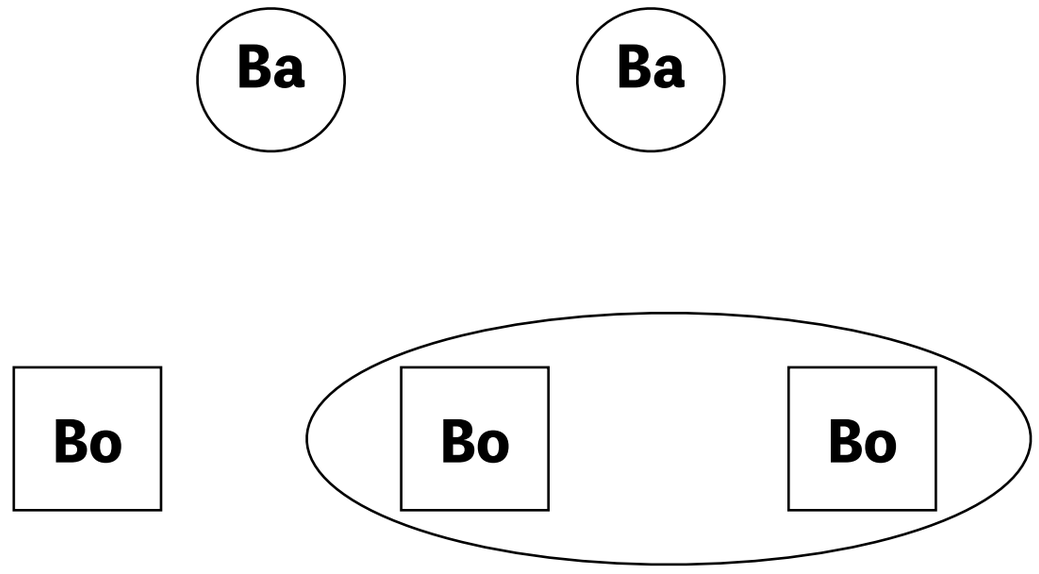
- 入れる箱を n 個選んで
- ボールを入れる



Balls and Boxes 2

- 入れる箱を n 個選んで

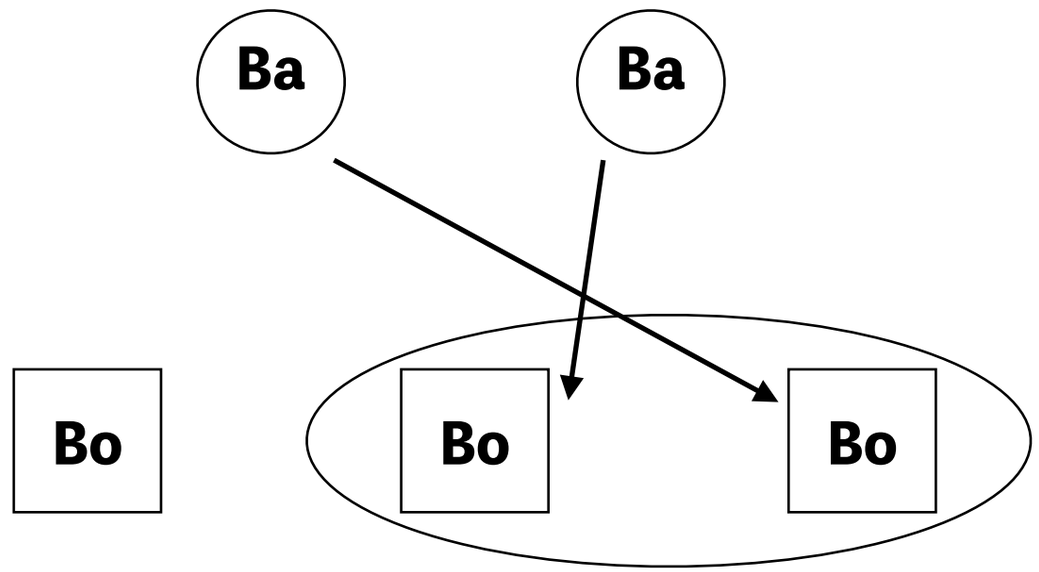
- ボールを入れる



Balls and Boxes 2

- 入れる箱を n 個選んで

- ボールを入れる



Balls and Boxes 2

- ${}_kP_n$ が答え
- 時間計算量 $\Theta(n)$

```cpp
// ボール：区別あり，箱：区別あり，入れ方：箱の中身は 1 つ以下
#include <iostream>
using namespace std;
const int MOD = 1000000007;

int perm(int k, int n) {
  if(k < n) return 0;
  long long ret = 1;
  for(int i = 0; i < n; i++) {
    ret *= (k - i);
    ret %= MOD;
  }
  return ret;
}

int main() {
  int n, k; cin >> n >> k;
  cout << perm(k, n) << endl;
  return 0;
}
```

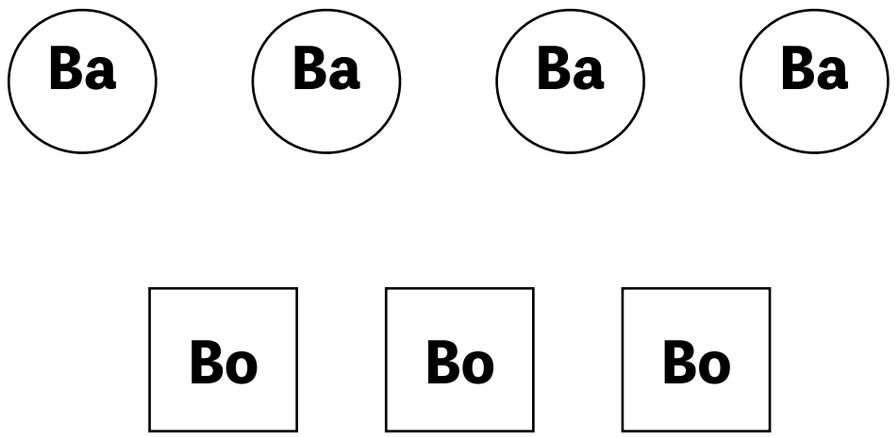
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	ここ
区別しない	区別する			
区別する	区別しない			
区別しない	区別しない			

Balls and Boxes 3

- ［問題］難易度: **青**
  - n 個の区別できるボールを, k 個の区別できる箱に入れるとき、可能な入れ方の総数を求めよ.
  - どの箱にも、1 つ以上のボールを入れる必要がある.

- e.g.
  - ［入力］$n = 4, k = 3$
  - ［出力］36



Balls and Boxes 3

- $W \subseteq \{1,2,...,k\}$ とする.
- $N(W) :=$ $i \in W$ に対して, 箱 i にボールが入らないような入れ方の総数.
- $X :=$ どの箱にも、1 つ以上のボールを入れる入れ方の総数.
- 包除原理より,

$$X = \sum_{W \subseteq \{1,2,...,k\}} (-1)^{|W|} N(W) \text{ となる.}$$

これを愚直に計算すると, $\Omega(2^k)$ かかってしまう.

Balls and Boxes 3

- 対称性を利用する
  - 例えば, $N(\{1,2\})$ と $N(\{1,5\})$ は同じになる.
  - **$N(W)$ は W の濃度にのみ依存した関数である.**
- $P(i) := i$ 箇所の箱にボールが入らないような入れ方の総数

$$X = \sum_{W \subseteq \{1,2,\ldots,k\}} (-1)^{|W|} N(W) = P(0) - P(1) + P(2) + \ldots + (-1)^k P(k)$$

となる.

- $P(i) =_n C_r \ (k-i)^n$ で計算可能

Balls and Boxes 3

```cpp
// ボール: 区別あり, 箱 : 区別あり, 入れ方: 1 つ以上
#include <iostream>
using namespace std;
const int MOD = 1000000007;
const long long MAXN = 1001024;

// 繰り返し 2 乗法
long long mod_pow(long long a, long long n) {
  long long res = 1;
  while (n > 0) {
    if (n & 1) res = res * a % MOD;
    a = a * a % MOD;
    n >>= 1;
  }
  return res;
}

long long fac[MAXN], finv[MAXN], inv[MAXN];

// 前処理 O(n)
void comb_init(){
  fac[0] = fac[1] = 1;
  finv[0] = finv[1] = 1;
  inv[1] = 1;
  for(long long i = 2; i < MAXN; i++){
    fac[i] = fac[i - 1] * i % MOD;
    inv[i] = MOD - inv[MOD%i] * (MOD / i) % MOD;
    finv[i] = finv[i - 1] * inv[i] % MOD;
  }
}

// 二項係数計算 O(1)
long long mod_comb(long long n, long long k){
  if (n < k) return 0;
  if (n < 0 || k < 0) return 0;
  return fac[n] * (finv[k] * finv[n - k] % MOD) % MOD;
}
```

```cpp
int main() {

  int n, k; cin >> n >> k;
  long long ans = 0;
  comb_init();
  for(int i = 0; i <= k; i++) {
    // i 個の箱を選んで, 絶対ボールを入れないことにする
    // 残りの箱に自由に入れる
    int rest = k - i;
    long long add = mod_comb(k, i) * mod_pow(rest, n) % MOD;

    if(i % 2 == 0) ans = (ans + add      ) % MOD;
    else           ans = (ans - add + MOD) % MOD;
  }

  cout << ans << endl;
  return 0;
}
```

前処理: $\Theta(k)$

本処理: $\Theta(k)$

nCr や べき乗の処理は
以下の検索ワードでググりましょう
(繰り返し 2 乗法, 逆元)

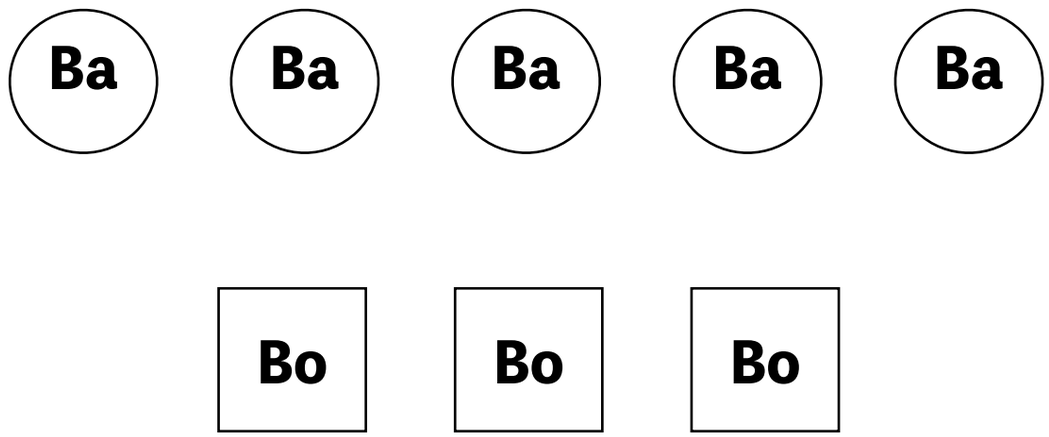
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)^i{}_kC_i(k-i)^n$
区別しない	区別する	ここ		
区別する	区別しない			
区別しない	区別しない			

Balls and Boxes 4

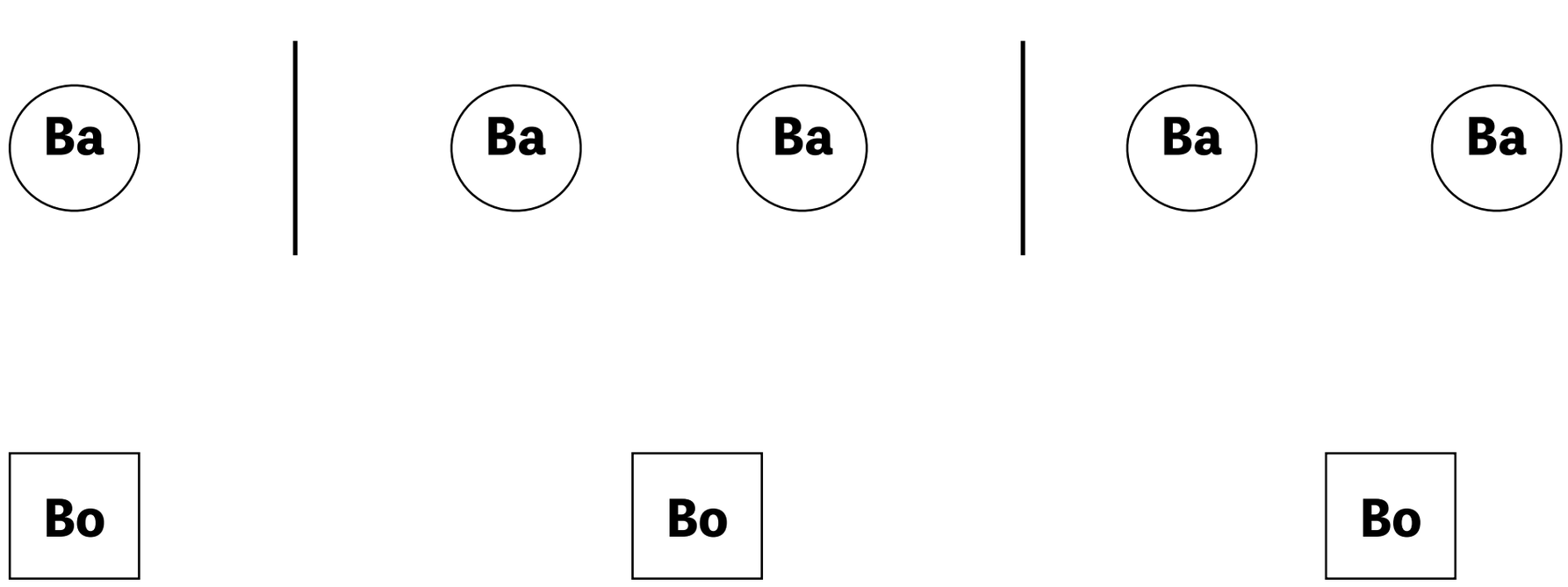
- ［問題］難易度: **緑**
  - n 個の区別できないボールを, k 個の区別できる箱に入れるとき、可能な入れ方の総数を求めよ.
  - ただし、箱に対する制限はないものとする.

- e.g.
  - ［入力］$n = 5, k = 3$
  - ［出力］21



Balls and Boxes 4

- （区別できない）棒とボールの並び替えに帰着



Balls and Boxes 4

```
int main() {

  comb_init();
  int n, k; cin >> n >> k;
  cout << mod_comb(n + k - 1, k - 1) << endl;
  return 0;
}
```

- ${}_{n+k-1}C_{k-1}$ が答え
- 時間計算量 $\Theta(k)$

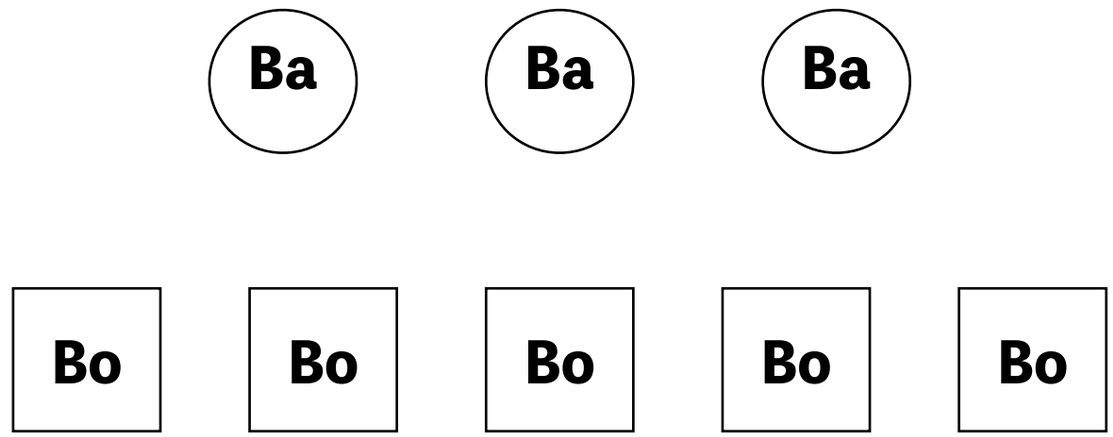
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)^i{}_kC_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	ここ	
区別する	区別しない			
区別しない	区別しない			

Balls and Boxes 5

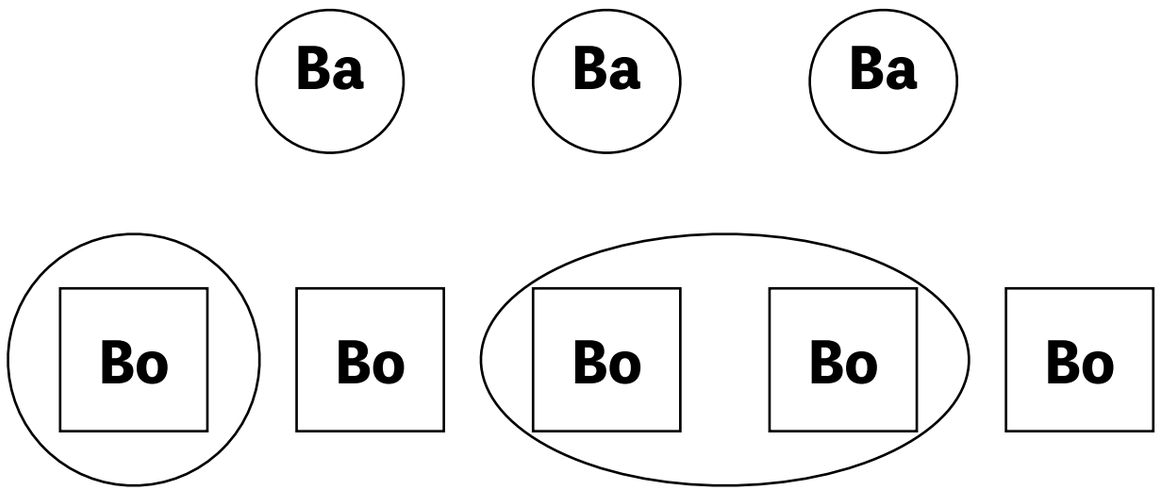
- ［問題］難易度: **茶**
  - n 個の区別できないボールを, k 個の区別できる箱に入れるとき、可能な入れ方の総数を求めよ.
  - 各箱に入れることのできるボールの数は高々 1 個とする.

- e.g.
  - ［入力］$n = 3, k = 5$
  - ［出力］10



Balls and Boxes 5

- 入れる箱を n 個選べば良い



Balls and Boxes 5

- ${}_kC_n$ が答え

```
int main() {

  comb_init();
  int n, k; cin >> n >> k;
  cout << mod_comb(k, n) << endl;
  return 0;
}
```

- 時間計算量 $\Theta(k)$

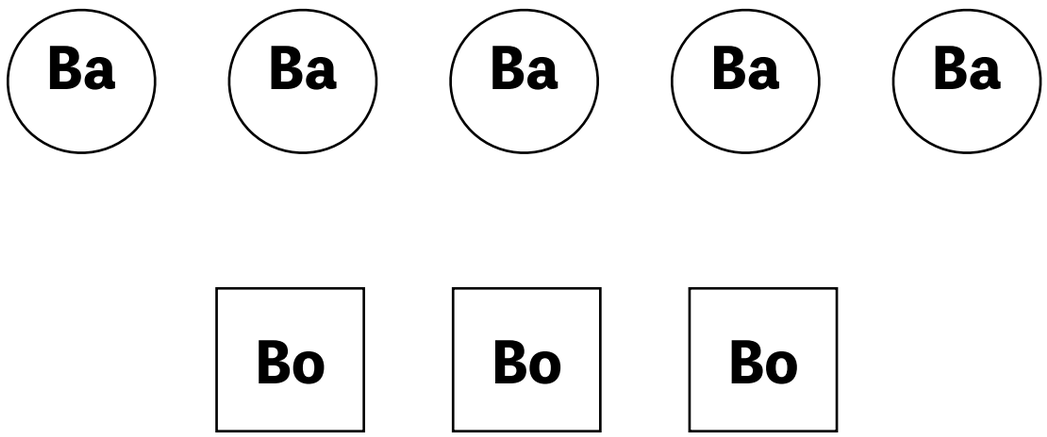
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)_k^i C_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	まだ半分..
区別する	区別しない			
区別しない	区別しない			

Balls and Boxes 6

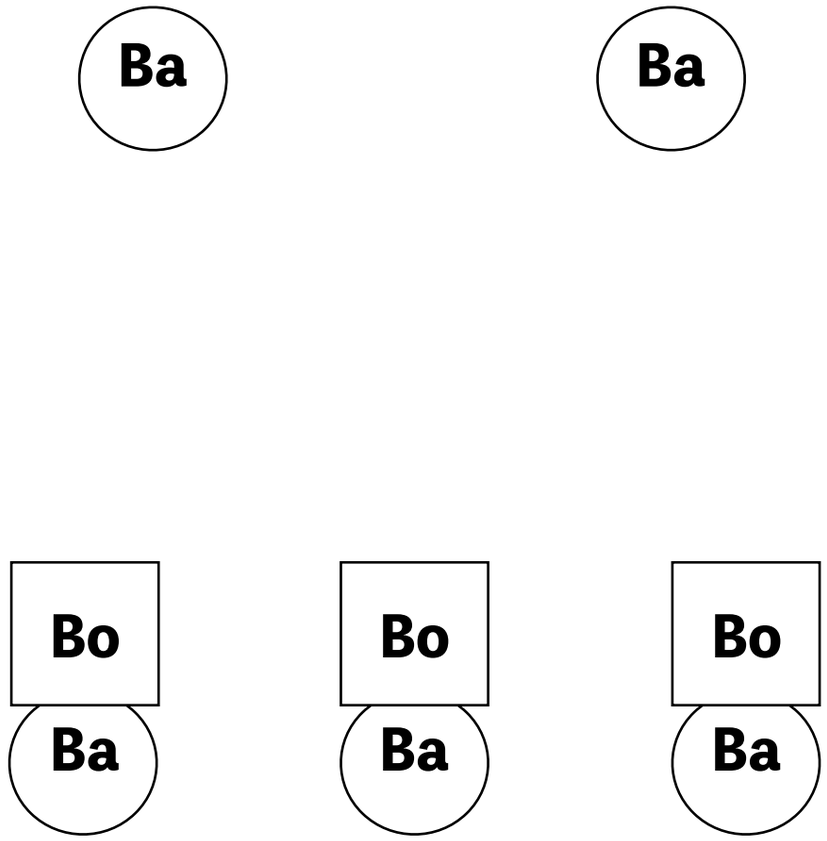
- ［問題］難易度: **茶**
  - n 個の区別できないボールを, k 個の区別できる箱に入れるとき、可能な入れ方の総数を求めよ.
  - どの箱にも、1 つ以上のボールを入れる必要がある.

- e.g.
  - ［入力］$n = 5, k = 3$
  - ［出力］6



Balls and Boxes 6

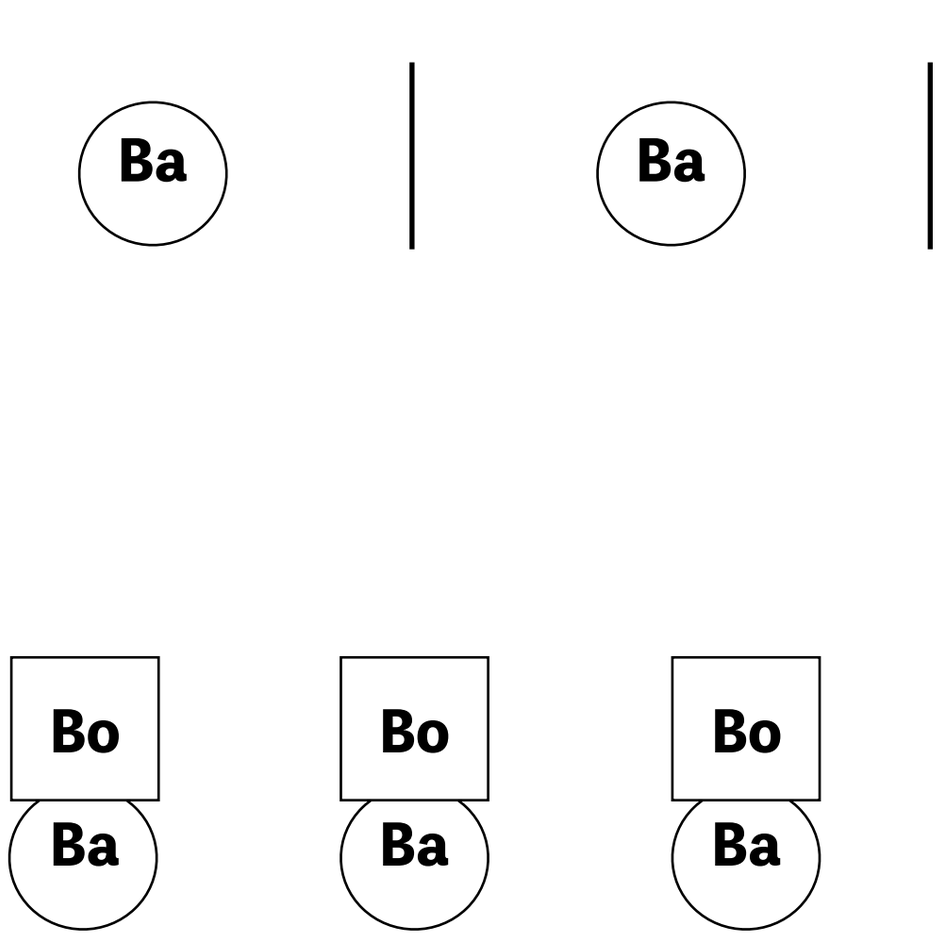
- とりあえずボールを 1 ずつ入れちゃう



Balls and Boxes 6

- とりあえずボールを 1 ずつ入れちゃう

- 棒のボールを並べる



Balls and Boxes 6

- $_{n-1}C_{n-k}$ が答え

```
int main() {

  comb_init();
  int n, k; cin >> n >> k;
  cout << mod_comb(n - 1, n - k) << endl;
  return 0;
}
```

- 時間計算量 $\Theta(n)$

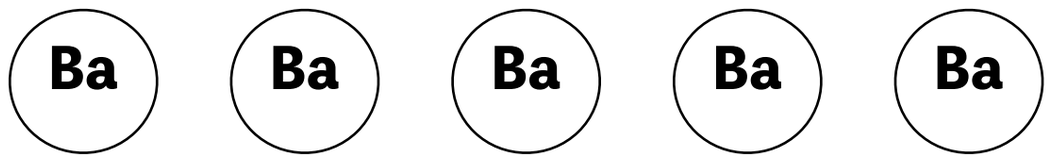
目次

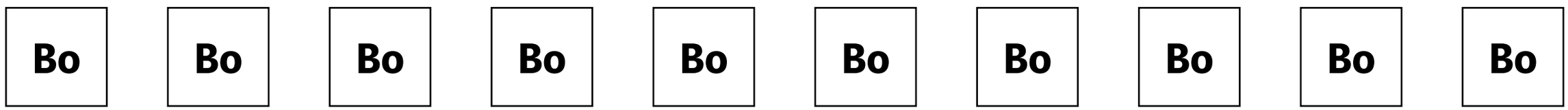
玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)^i{}_kC_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	${}_{n-1}C_{n-k}$
区別する	区別しない	???	ここ	
区別しない	区別しない			

Balls and Boxes 8

- ［問題］難易度: 灰
  - n 個の区別できるボールを, k 個の区別できない箱に入れるとき、可能な入れ方の総数を求めよ.
  - 各箱に入れることのできるボールの数は高々 1 個とする.

- e.g.
  - ［入力］$n = 5, k = 10$
  - ［出力］1





Balls and Boxes 8

- 0 か 1 が答え
- 時間計算量 $O(1)$

```cpp
// ボール：区別あり，箱：区別なし，入れ方：1 つ以下
#include <iostream>
using namespace std;

int main() {

  int n, k; cin >> n >> k;

  // 箱が足りるなら 1
  // 足りないなら 0
  if(n <= k) cout << 1 << endl;
  else cout << 0 << endl;

  return 0;
}
```

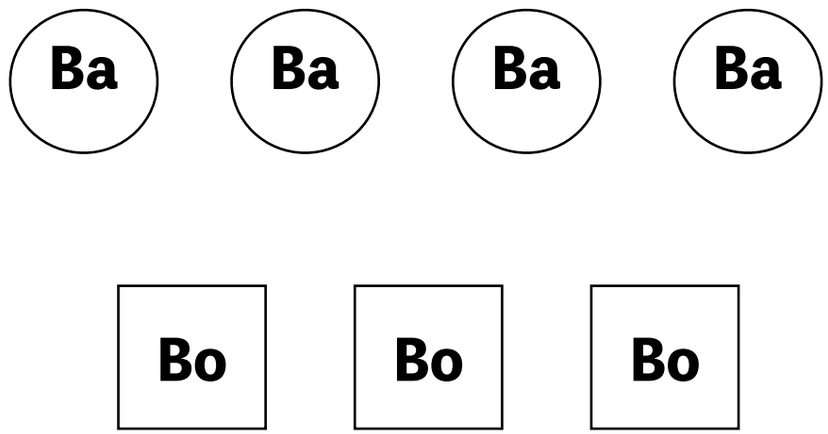
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)^i{}_kC_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	${}_{n-1}C_{n-k}$
区別する	区別しない	???	0 or 1	ここ
区別しない	区別しない			

Balls and Boxes 9

- ［問題］難易度: 青
  - n 個の区別できるボールを, k 個の区別できない箱に入れるとき、可能な入れ方の総数を求めよ.
  - どの箱にも、1 つ以上のボールを入れる必要がある.

- e.g.
  - ［入力］$n = 4, k = 3$
  - ［出力］6

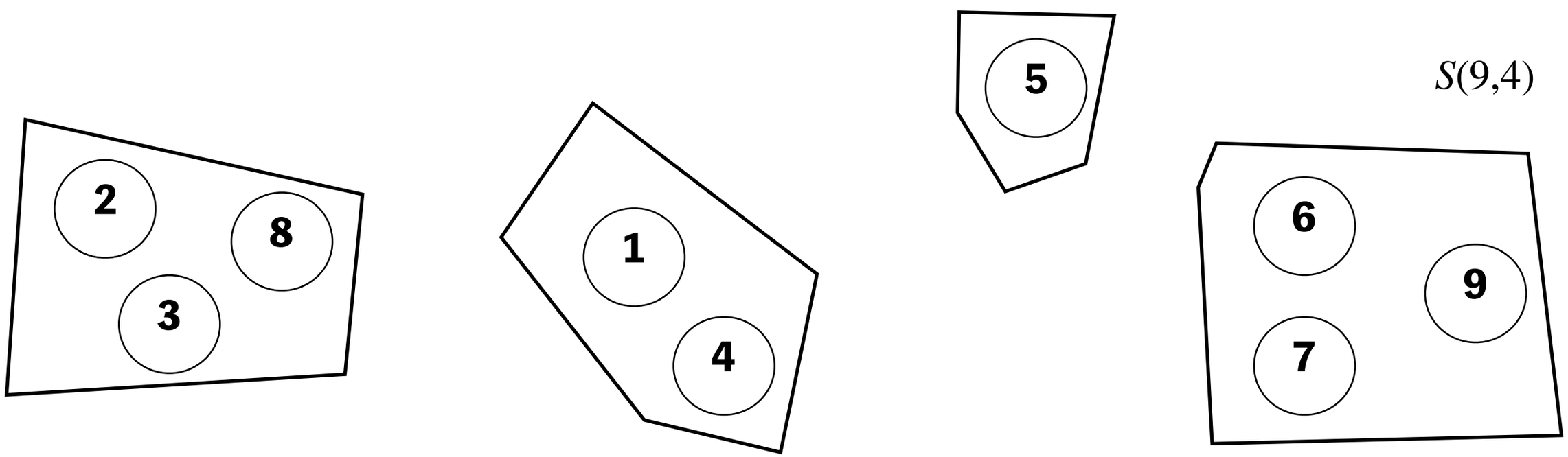


Balls and Boxes 9

- n 個の区別できるボールをちょうど k 個グループに分ける方法を考えれば良い.

- これは以下の漸化式で求めることができる.

$$S(n,k) = S(n-1,k-1) + kS(n-1,k)$$



$S(9,4)$

Balls and Boxes 9

- 玉 1 が単独でグループを形成するとき

  残り $n-1$ 個の玉を $k-1$ グループに分ける方法は

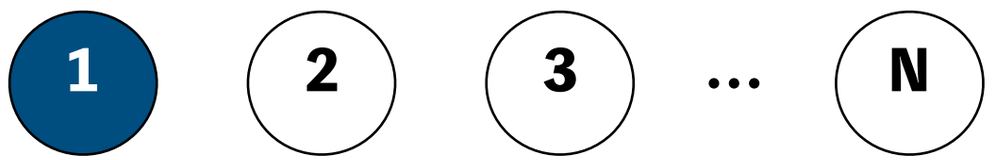
  $S(n-1,k-1)$ 通り

- 玉 1 が単独でない場合

  1 以外の $n-1$ 個について k グループに分ける方法が

  $S(n-1,k)$ 通りあり,

  1 を入れるグループが k 個あるので $kS(n-1,k)$ 通り



Balls and Boxes 9

- $S(n, k)$ が答え
- 時間計算量 $O(nk)$

```cpp
// ボール：区別あり，箱：区別なし，入れ方：1 個以上
// スターリング数
// S(n, k) = S(n - 1, k - 1) + k * S(n - 1, k)
#include <iostream>
using namespace std;
const int MOD = 1000000007;
const long long MAXN = 1010;
long long S[MAXN][MAXN] = {};

void star_init() {
  S[0][0] = 1;
  for(int n = 1; n < MAXN; n++) {
    for(int k = 1; k < MAXN; k++) {
      S[n][k] = S[n - 1][k - 1] + k * S[n - 1][k];
      S[n][k] %= MOD;
    }
  }
}

int main() {

  star_init();
  int n, k; cin >> n >> k;
  cout << S[n][k] << endl;
  return 0;
}
```

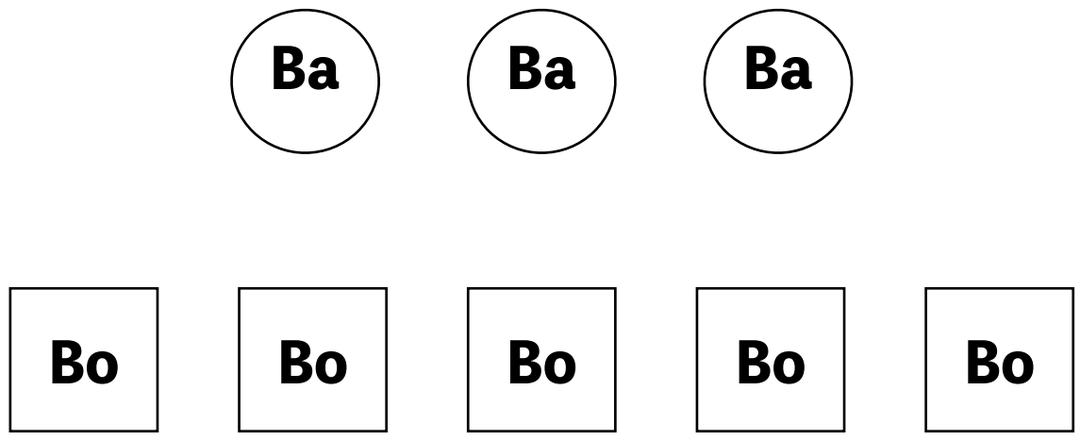
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)^i{}_kC_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	${}_{n-1}C_{n-k}$
区別する	区別しない	ここ	0 or 1	$S(n,k)$
区別しない	区別しない			

Balls and Boxes 7

- ［問題］難易度: 青
  - n 個の区別できるボールを, k 個の区別できない箱に入れるとき、可能な入れ方の総数を求めよ.
  - ただし、箱に対する制限はないものとする.

- e.g.
  - ［入力］$n = 5, k = 3$
  - ［出力］5

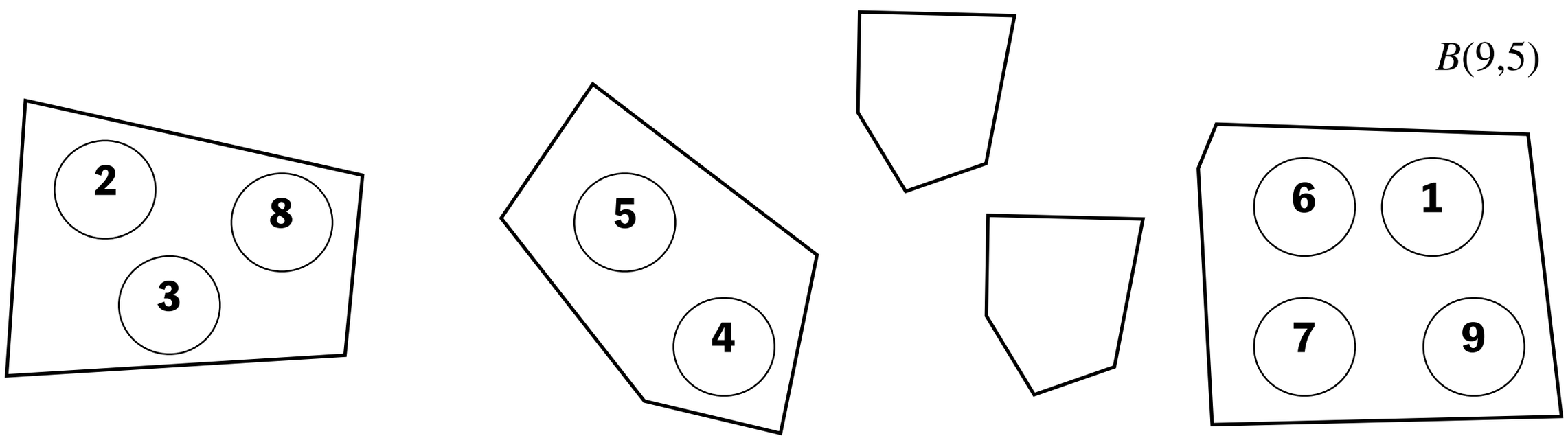


Balls and Boxes 7

- n 個の区別できるボールをちょうど k 個以下グループに分ける方法を考えれば良い.

- これは以下の漸化式で求めることができる.

$$B(n,k) = S(n,0) + S(n,1) + \ldots + S(n,k)$$



Balls and Boxes 7

- $B(n, k)$ が答え
- 時間計算量 $O(nk)$

```cpp
// ボール: 区別あり, 箱 : 区別なし, 入れ方: 制限なし
// ベル数
// B(n, k) = S(n, 0) + S(n, 1) + ... + S(n, k)
#include <iostream>
using namespace std;
const int MOD = 1000000007;
const long long MAXN = 1010;
long long S[MAXN][MAXN] = {};

void star_init() {
  S[0][0] = 1;
  for(int n = 1; n < MAXN; n++) {
    for(int k = 1; k < MAXN; k++) {
      S[n][k] = S[n - 1][k - 1] + k * S[n - 1][k];
      S[n][k] %= MOD;
    }
  }
}


int main() {

  star_init();
  int n, k; cin >> n >> k;
  long long B = 0;
  for(int i = 0; i <= k; i++) {
    B += S[n][i];
    B %= MOD;
  }
  cout << B << endl;
  return 0;
}
```

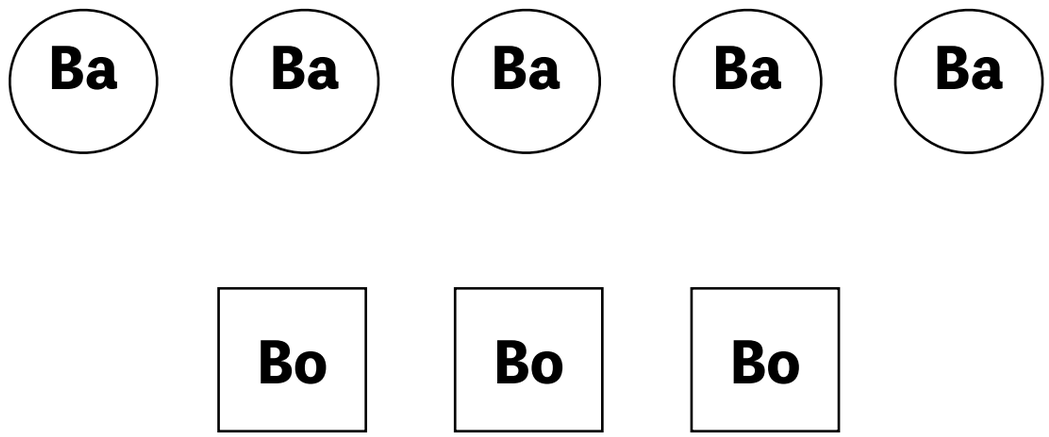
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)_k^i C_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	${}_{n-1}C_{n-k}$
区別する	区別しない	$B(n,k)$	0 or 1	$S(n,k)$
区別しない	区別しない	もう少し		

Balls and Boxes 10

- ［問題］難易度:青
  - n 個の区別できないボールを, k 個の区別できない箱に入れるとき、可能な入れ方の総数を求めよ.
  - ただし、箱に対する制限はないものとする.

- e.g.
  - ［入力］$n = 5, k = 3$
  - ［出力］5



Balls and Boxes 10

- 自然数 n を k 個 の 0 以上の整数の和で表す方法の通り数を考えれば良い. (ただし, 足す順番を入れ替えて同じものは同一視する)

- これは以下の漸化式で求めることができる.

- $P(n,k) = P(n,k-1) + P(n-k,k)$

$P(5,3) = 5$

$5 = 0+0+5 = 0+1+4 = 0+2+3 = 1+1+3 = 1+2+2$

Balls and Boxes 10

- 分解に 0 を含むもの場合

  一つの 0 を取り除いて, n を $k-1$ 個の 0 以上の

  整数和に分解する問題に帰着することができるので

  $P(n, k-1)$ 通り

  $$5 = 0 + 0 + 5$$

- 分解に 0 を含まない場合

  k 個の整数が全て 1 以上なので, それぞれ 1 引くことで

  $n-k$ を k 個の 0 以上の整数和に分解する問題に帰着できるので

  $P(n-k, k)$ 通り

  $$5 = 1 + 1 + 3$$

Balls and Boxes 10

- $P(n, k)$ が答え
- 時間計算量 $O(nk)$

```cpp
// ボール: 区別なし, 箱 : 区別なし, 入れ方: 制限なし
// 分割数
// P(n, k) = P(n, k - 1) + P(n - k, k)
#include <iostream>
using namespace std;
const int MOD = 1000000007;
const long long MAXN = 1010;
long long P[MAXN][MAXN] = {};

void pertition_init() {
  for(int k = 0; k < MAXN; k++) P[0][k] = 1;
  for(int n = 1; n < MAXN; n++) {
    for(int k = 1; k < MAXN; k++) {
      P[n][k] += P[n][k - 1] + (n - k >= 0 ? P[n - k][k] : 0);
      P[n][k] %= MOD;
    }
  }
}

int main() {

  pertition_init();
  int n, k; cin >> n >> k;
  cout << P[n][k] << endl;
  return 0;
}
```

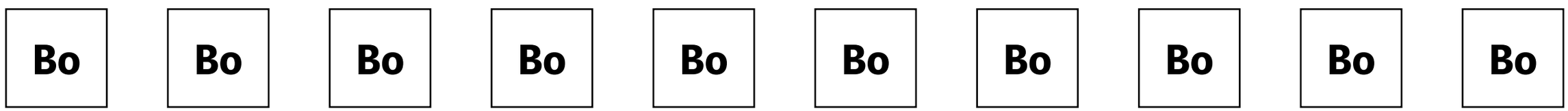
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)_k^i C_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	${}_{n-1}C_{n-k}$
区別する	区別しない	$B(n,k)$	0 or 1	$S(n,k)$
区別しない	区別しない	$P(n,k)$	もう少し	

Balls and Boxes 11

- ［問題］難易度: 灰
  - n 個の区別できないボールを, k 個の区別できない箱に入れるとき、可能な入れ方の総数を求めよ.
  - 各箱に入れることのできるボールの数は高々 1 個とする.

- e.g.
  - ［入力］$n = 5, k = 10$
  - ［出力］1



Balls and Boxes 11

- 0 か 1 が答え
- 時間計算量 $O(1)$

```cpp
// ボール：区別なし，箱：区別なし，入れ方：1 つ以下
#include <iostream>
using namespace std;

int main() {

  int n, k; cin >> n >> k;

  // 箱が足りるなら 1
  // 足りないなら 0
  cout << (n <= k) << endl;

  return 0;
}
```

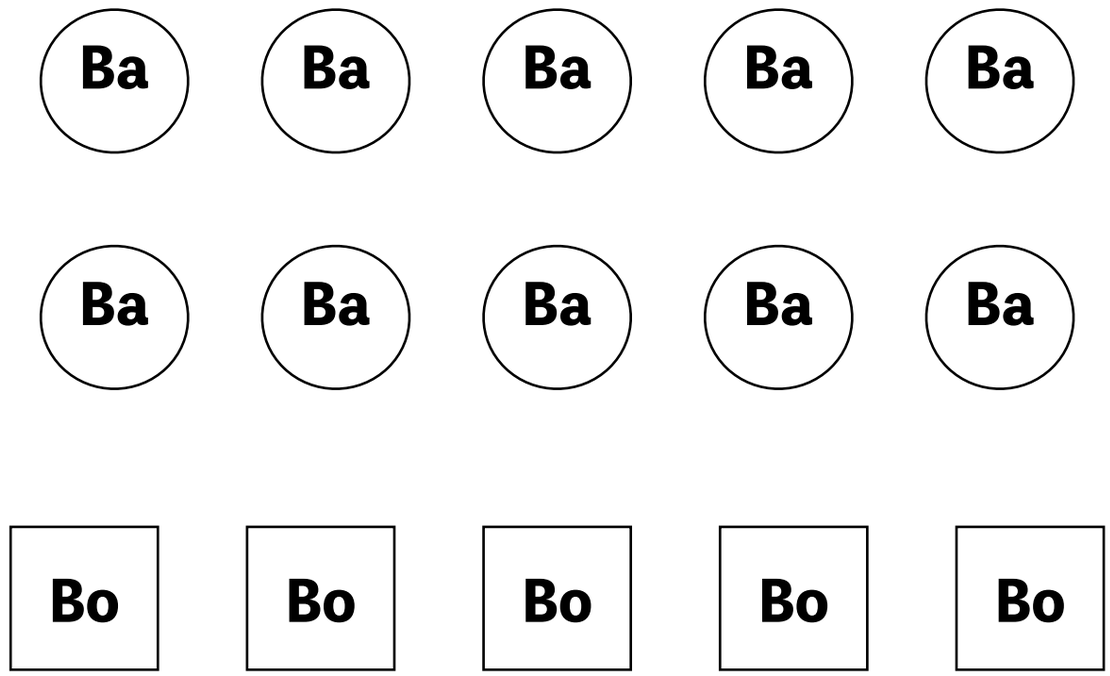
目次

玉	箱	制限なし	1 個以内	1 個以上
区別する	区別する	k^n	${}_kP_n$	$\sum_{i=0}^{k}(-1)_k^i C_i(k-i)^n$
区別しない	区別する	${}_{n+k-1}C_{k-1}$	${}_kC_n$	${}_{n-1}C_{n-k}$
区別する	区別しない	$B(n,k)$	0 or 1	$S(n,k)$
区別しない	区別しない	$P(n,k)$	0 or 1	ラスト!!!

Balls and Boxes 12

- ［問題］難易度:青
  - n 個の区別できないボールを, k 個の区別できない箱に入れるとき、可能な入れ方の総数を求めよ.
  - どの箱にも、1 つ以上のボールを入れる必要がある.

- e.g.
  - ［入力］$n = 10, k = 5$
  - ［出力］7



Balls and Boxes 12

- $P(n-k,k)$ が答え
- 時間計算量 $O(nk)$

```cpp
// ボール: 区別なし, 箱 : 区別なし, 入れ方: 1 個以上
// 分割数
// P(n, k) = P(n, k - 1) + P(n - k, k)
#include <iostream>
using namespace std;
const int MOD = 1000000007;
const long long MAXN = 1010;
long long P[MAXN][MAXN] = {};

void pertition_init() {
  for(int k = 0; k < MAXN; k++) P[0][k] = 1;
  for(int n = 1; n < MAXN; n++) {
    for(int k = 1; k < MAXN; k++) {
      P[n][k] += P[n][k - 1] + (n - k >= 0 ? P[n - k][k] : 0);
      P[n][k] %= MOD;
    }
  }
}

int main() {

  pertition_init();
  int n, k; cin >> n >> k;
  if(n - k < 0) cout << 0 << endl;
  else cout << P[n - k][k] << endl;
  return 0;
}
```

参考資料

- AOJ 写像 12 相問題集
- 「写像12相」を総整理！ ～ 数え上げ問題の学びの宝庫 ～
- 高校数学の美しい物語