**CHUYÊN ĐỀ: TOÁN TỔ HỢP (COMBINATORICS)   
TRONG LẬP TRÌNH THI ĐẤU**

**MỤC LỤC**

**[1. GIỚI THIỆU](#_Toc203723122)** [2](#_Toc203723122)

**[2. CƠ SỞ LÝ THUYẾT](#_Toc203723123)** [3](#_Toc203723123)

[2.1.](#_Toc203723124) **[Các quy tắc cơ bản của toán tổ hợp](#_Toc203723124)** [3](#_Toc203723124)

**[2.2. Các khái niệm cơ bản trong tổ hợp](#_Toc203723125)** [4](#_Toc203723125)

**[2.3. Tổ hợp:](#_Toc203723126)** [5](#_Toc203723126)

**[3. BÀI TẬP CƠ BẢN](#_Toc203723127)** [8](#_Toc203723127)

**[Bài 1: Tính tổ hợp chập k của N (Mã bài - CKN3)](#_Toc203723128)** [8](#_Toc203723128)

**[Bài 2. Hệ số nhị thức (Mã bài – HSNT)](#_Toc203723129)** [10](#_Toc203723129)

**[Bài 3: Tạo xâu (Mã bài – TAOXAU)](#_Toc203723130)** [11](#_Toc203723130)

**[Bài 4: Dãy ngoặc (Mã bài – BRACKET)](#_Toc203723131)** [12](#_Toc203723131)

**[Bài 5: Đếm vòng cổ (Mã bài – NECKLACES)](#_Toc203723132)** [14](#_Toc203723132)

**[Bài 6: Chia táo (Mã bài – CHIATAO)](#_Toc203723133)** [16](#_Toc203723133)

**[4. BÀI TẬP VẬN DỤNG](#_Toc203723134)** [17](#_Toc203723134)

**[Bài 1. HTDEN - Thiết kế hệ thống đèn (Đề thi chính thức DHBB 2022)](#_Toc203723135)** [17](#_Toc203723135)

**[Bài 2. LEUTHI - Lều thi (Đề thi chính thức DHBB 2019)](#_Toc203723136)** [19](#_Toc203723136)

**[Bài 3. MATRAN - Ma trận (Đề chọn ĐTQG Đà Nẵng năm 2020-2021)](#_Toc203723137)** [22](#_Toc203723137)

**[Bài 4. HOANVI - Hoán vị không bất động (Đề thi THTC Vòng Khu vực 2021)](#_Toc203723138)**  [25](#_Toc203723138)

**[Bài 5. ROUTES - Số đường đi (Nguồn bài của thầy Đỗ Đức Đông)](#_Toc203723139)** [29](#_Toc203723139)

**[Bài 6: PASS - Mật khẩu (Đề thi chính thức DHBB 2021)](#_Toc203723140)** [31](#_Toc203723140)

**[Bài 7: GIEOXUCXAC - Gieo xúc xắc](#_Toc203723141)** [34](#_Toc203723141)

**[Bài 8: SONGACH - Sơn gạch](#_Toc203723142)** [36](#_Toc203723142)

**[Bài 9: HOANVITRON - Hoán vị vòng tròn](#_Toc203723143)** [38](#_Toc203723143)

**[Bài 10: BALL - Chọn bóng](#_Toc203723144)** [41](#_Toc203723144)

**[5. BÀI TẬP THÊM](#_Toc203723145)** [43](#_Toc203723145)

**[Bài 1: BRACKET2 - Dãy ngoặc 2](#_Toc203723146)** [43](#_Toc203723146)

**[Bài 2: TOMAU - Tô màu](#_Toc203723147)** [45](#_Toc203723147)

**[Bài 3: HINHTHANG - Hình thang cân (Đề thi thử DHBB 2021)](#_Toc203723148)** [47](#_Toc203723148)

**[Bài 4: Hành trình](#_Toc203723149)** [50](#_Toc203723149)

**[6. KẾT LUẬN](#_Toc203723150)** [55](#_Toc203723150)

**[7. TÀI LIỆU THAM KHẢO](#_Toc203723151)** [55](#_Toc203723151)

**Chuyên đề: TOÁN TỔ HỢP (COMBINATORICS)**

**TRONG LẬP TRÌNH THI ĐẤU**

# **1. GIỚI THIỆU**

Toán tổ hợp (Combinatorics) là một nhánh quan trọng của toán học, chuyên nghiên cứu về cách sắp xếp, chọn lựa và tổ hợp các phần tử trong một tập hợp. Đây là lĩnh vực cung cấp nền tảng toán học mạnh mẽ để giải quyết nhiều bài toán liên quan đến đếm, tối ưu hóa và phân tích tổ hợp. Trong bối cảnh lập trình thi đấu, toán tổ hợp không chỉ là một công cụ hữu ích mà còn là chìa khóa để giải quyết các bài toán phức tạp, đòi hỏi tư duy logic và khả năng tính toán nhanh chóng.

Các khái niệm cơ bản như tổ hợp, chỉnh hợp, hoán vị và các nguyên lý đếm đóng vai trò quan trọng trong việc xác định số lượng cách sắp xếp hoặc chọn lựa các đối tượng từ một tập hợp. Bên cạnh đó, các kỹ thuật nâng cao như phân hoạch, tổ hợp với lặp lại, và đếm trong không gian đa chiều cũng thường xuyên xuất hiện trong các bài toán thực tế và bài toán thi đấu.

Việc thành thạo toán tổ hợp không chỉ giúp học sinh nhanh chóng giải quyết các bài toán về đếm mà còn hỗ trợ việc tối ưu hóa thuật toán, một yếu tố quan trọng trong các cuộc thi lập trình. Hiểu rõ cách áp dụng toán tổ hợp vào các bài toán cụ thể sẽ mang lại cho học sinh lợi thế lớn, từ việc giải quyết bài toán đường đi ngắn nhất đến việc tối ưu hóa quy hoạch tài nguyên hoặc phân tích dữ liệu.

Nội dung chuyên đề được chia thành các phần cụ thể sau:

1. **Cơ sở lý thuyết:**  
   Phần này cung cấp các quy tắc cơ bản và các khái niệm quan trọng trong toán tổ hợp. Bạn sẽ học cách áp dụng các nguyên tắc đếm, hiểu rõ các đặc điểm của tổ hợp và cách tính toán chúng một cách hiệu quả.
2. **Bài tập cơ bản:**  
   Bao gồm các bài tập minh họa đơn giản, giúp củng cố kiến thức lý thuyết. Đây là bước đầu tiên để làm quen với cách áp dụng toán tổ hợp vào các bài toán thực tế.
3. **Bài tập vận dụng:**  
   Phần này chứa các bài toán ở mức độ khó hơn, yêu cầu kết hợp kiến thức tổ hợp với tư duy logic và khả năng phân tích. Các bài tập này được lấy nguồn từ các cuộc thi lập trình để làm quen với các dạng bài phổ biến.
4. **Bài tập thêm:**  
   Bao gồm các bài toán mở rộng và nâng cao, mang tính thử thách. Phần này giúp học sinh phát triển khả năng giải quyết vấn đề sáng tạo và rèn luyện kỹ năng tối ưu hóa thuật toán.

Toàn bộ code và test của chuyên đề thầy cô có thể xem trong file đính kèm theo chuyên đề này.

# **2. CƠ SỞ LÝ THUYẾT**

## 2.1. **Các quy tắc cơ bản của toán tổ hợp**

**2.1.1 Quy tắc cộng:**

Nếu chúng ta có **A** số cách thực hiện Nhiệm vụ 1 và **B** số cách thực hiện Nhiệm vụ 2 thì tổng số cách để chọn một trong hai Nhiệm vụ bằng **A+B.**

Vì vậy, nhìn chung, nếu có **N** nhiệm vụ và nhiệm vụ thứ i có thể được thực hiện theo a[i] cách thì có **a1+ a2+ a3+…** cách để thực hiện một trong các nhiệm vụ.

**Ví dụ:**

*Hãy tưởng tượng bạn có 3 chiếc mũ khác nhau, 2 chiếc áo khác nhau và 4 chiếc quần khác nhau. Nếu bạn muốn tặng bất kỳ món đồ nào thì tổng số cách để làm là bao nhiêu?****Trả lời:*** *3 + 2 + 4 = 9.*

**2.1.2. Quy tắc nhân:**

Nếu chúng ta có A số cách thực hiện Nhiệm vụ 1 và B số cách thực hiện Nhiệm vụ 2 thì tổng số cách thực hiện cả hai nhiệm vụ bằng A\*B. Vì vậy, nói chung, nếu có N nhiệm vụ và nhiệm vụ thứ i có thể được thực hiện theo a[i] cách thì có a1\* a2\* a3\*… cách để thực hiện tất cả các nhiệm vụ.

**Ví dụ:** Hãy tưởng tượng bạn có 3 chiếc mũ khác nhau, 2 chiếc áo khác nhau và 4 chiếc quần khác nhau. Bạn muốn mặc một bộ quần áo và để mặc được bạn phải mặc 1 chiếc mũ, 1 chiếc áo và 1 chiếc quần. Bạn có thể làm như vậy theo bao nhiêu cách?

**Giải pháp:** 3 \* 2 \* 4 = 24

## **2.2. Các khái niệm cơ bản trong tổ hợp**

**2.2.1. [Giai thừa:](https://www.geeksforgeeks.org/permutation/)**

Công dụng chính của giai thừa là đếm số lượng hoán vị (số cách sắp xếp một số đối tượng). Nó có thể được sử dụng để trả lời các loại câu hỏi sau:

**Câu hỏi 1.** Một lớp học chỉ còn 3 chỗ trống. Ba người P, A và R đến cùng một lúc. Có bao nhiêu cách sắp xếp P, A và R vào 3 chỗ trống đó?

**Giải pháp:**

*Đối với ghế đầu tiên, chúng ta có* ***3 lựa chọn*** *là P, A và R. Chúng ta hãy chọn ngẫu nhiên A cho ghế đầu tiên.*

*Đối với ghế thứ 2, chúng ta có* ***2 lựa chọn*** *là P và R. Chúng ta hãy chọn ngẫu nhiên R cho ghế thứ hai.*

*Đối với ghế thứ ba, chúng ta có* ***1 lựa chọn*** *tức là P. Tóm lại, chúng tôi đã thực hiện như sau:*

*Đặt một người vào ghế số 1* ***,*** *đặt một người vào ghế số 2* ***và*** *đặt một người vào ghế số 3. Việc sử dụng và xuất phát từ thực tế là việc chiếm giữ cả 3 ghế là bắt buộc.*

*Trong toán học, và liên quan đến phép nhân, do đó ta có thể nói rằng tổng số lựa chọn = 3 × 2 × 1 = 3!*

*Nếu chúng ta thay đổi thứ tự chỗ ngồi thành P ở ghế đầu tiên, A ở ghế thứ hai và R ở ghế thứ ba, thì tổng số lựa chọn có thay đổi không? Không. Lý do là vì cả ba yếu tố P, A và R đều có tầm quan trọng như nhau.*

**Câu hỏi 2.** Tìm số cách sắp xếp 5 người sao cho luôn có 2 người ngồi cạnh nhau?

**Giải pháp:**

*Chúng ta hãy coi 2 người như một đơn vị và 3 người còn lại như 3 đơn vị riêng biệt, vậy chúng ta có tổng cộng 4 đơn vị. Số cách sắp xếp 4 đơn vị này là 4! (giống như cách chúng ta đã chứng minh ở bài toán trước)*

*Số cách sắp xếp 2 người ngồi với nhau là 2! Tóm lại, số cách sắp xếp 4 đơn vị và 2 người với nhau là 4! × 2!*

**Câu hỏi 3.** Tìm tất cả các từ ba chữ cái bắt đầu và kết thúc bằng một nguyên âm. Biết rằng việc lặp lại các chữ cái là không được phép.

**Giải pháp:**

*Tổng số nguyên âm trong tiếng Anh = 7 (a, e, i, o, u, y, w). Tổng số phụ âm trong tiếng Anh = 26 – 7 = 19. Lựa chọn cho chữ cái đầu tiên là 7. Có 6 lựa chọn cho chữ cái thứ ba (vì 1 nguyên âm được đặt làm chữ cái đầu tiên). Các lựa chọn cho chữ cái ở giữa là 19 + (7 – 2) = 24 (19 phụ âm + nguyên âm không được đặt). Do đó, tổng số hoán vị là 7 × 6 × 24 = 1008*

*Xin lưu ý rằng ở đây đầu tiên chúng ta thỏa mãn điều kiện nguyên âm cho chữ cái đầu tiên và chữ cái thứ ba, sau đó mới đặt chữ cái ở giữa.*

**2.2.2. [Hoán vị:](https://www.geeksforgeeks.org/permutation/)**

Trong hoán vị, chúng ta chủ yếu giải quyết bốn loại vấn đề.

**Hoán vị có lặp lại:**

**Câu hỏi:** Có bao nhiêu số có 3 chữ số lớn hơn 500 được tạo thành từ các chữ số 3, 4, 5 và 7?

***Giải pháp*** *: Vì một số có ba chữ số, lớn hơn 500 sẽ có 5 hoặc 7 ở vị trí hàng trăm, chúng ta có 2 lựa chọn cho vị trí này. Không có giới hạn về việc lặp lại các chữ số, do đó đối với 2 chữ số còn lại, chúng ta có 4 lựa chọn cho mỗi số. Vì vậy, tổng số hoán vị là: 2 × 4 × 4 = 32*

**Hoán vị không lặp lại:**

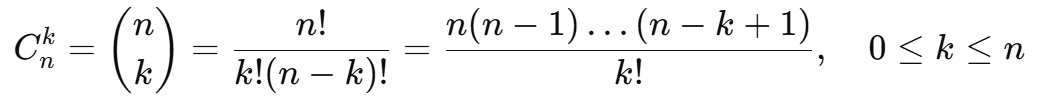
**Câu hỏi:** Có bao nhiêu số có 3 chữ số chia hết cho 3 được tạo thành từ các chữ số 2, 4, 6 và 8 mà không lặp lại?

***Giải pháp*** *: Để một số chia hết cho 3 thì tổng các chữ số của số đó phải chia hết cho 3. Từ tập hợp đã cho, có thể hình thành nhiều cách sắp xếp khác nhau như 444 nhưng vì không được phép lặp lại nên chúng ta sẽ không xét đến chúng. Chúng ta chỉ còn lại 2 trường hợp là 2, 4, 6 và 4, 6, 8. Số cách sắp xếp là 3! trong mỗi trường hợp. Do đó, tổng số hoán vị là: 3! + 3! = 12*

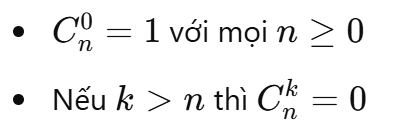
## **2.3. [Tổ hợp:](https://www.geeksforgeeks.org/permutations-and-combinations/" \t "_blank)**

Trong toán học, tổ hợp là cách chọn các phần tử từ một nhóm mà không phân biệt thứ tự chọn. Mỗi tập con gồm *k* phần tử khác nhau (không phân biệt thứ tự) của tập hợp *n* phần tử đã cho (0 ≤ *k* ≤ *n*) được gọi là một tổ hợp chập *k* của *n* phần tử.

Số các tổ hợp chập *k* của *n* phần tử khác nhau được kí hiệu là:

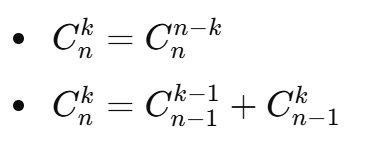


Ta quy ước:



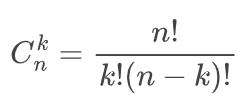
**Một số tính chất của tổ hợp**

Với mọi *n* ≥ 1 và 0 ≤ *k* ≤ *n*, ta có:



**Cách tính tổ hợp**

1. Sử dụng định nghĩa



Với công thức này, ta nghĩ ngay đến một thuật toán đơn giản: Tính *n*!, *k*! và (*n-k*)!. Từ đó tính được tổ hợp.

**long long res = 1;**

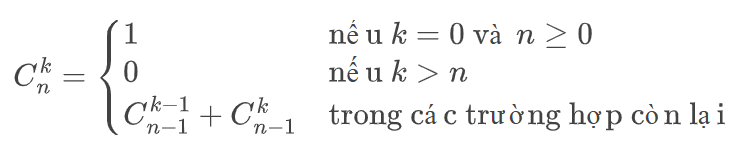
**for (int i = 1; i <= n; i++) res = res \* i;**

**for (int i = 1; i <= k; i++) res = res / i;**

**for (int i = 1; i <= n-k; i++) res = res / i;**

Cách tiếp cận trên rất tự nhiên, dễ nghĩ, dễ thực hiện nhưng lại có một trở ngại: giá trị của *n*! có thể rất lớn (khi *n*=20 thì *n*! = 2.42 x 1018)

1. Sử dụng công thức truy hồi



Với công thức truy hồi này, ta sẽ sử dụng một mảng hai chiều *C[n][k]* để tính tổ hợp.

**for (int i = 0; i <= n; i++){**

**C[i][0] = 1 % MOD;**

**for (int k = 1; k <= i; k++){**

**C[i][k] = (C[i - 1][k - 1] + C[i - 1][k]) % MOD;**

**}**

**}**

Độ phức tạp thuật toán: O(*n*2).

Đây là cách đơn giản, dễ nghĩ, dễ code đúng, nên sử dụng trong trường hợp *n* nhỏ để tiết kiệm thời gian.

1. Tính tổ hợp theo modulo M

Rào cản lớn nhất cho việc sử dụng định nghĩa để tính tổ hợp là *n*! quá lớn. Tuy nhiên khi ta cần lấy kết quả theo modulo *M*, đó lại là vấn đề khác.

**Điều kiện sử dụng:** *M* nguyên tố và *n* < *M*

**Kiến thức sử dụng:**

* Nghịch đảo modulo (Modular Inverse):
* Định lý Fermat nhỏ
* Lũy thừa nhanh

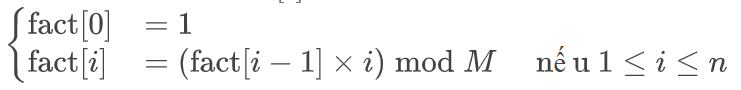
**Ý tưởng:**

* Ta viết lại: 
* Ta sử dụng hai mảng: mảng *fact[i]* để lưu *i*! mod *M* và mảng *ifact[i]* để lưu (*i*!)−1 mod *M*. Sau đó dùng công thức (sử dụng định lý Fermat nhỏ):

*ifact[i]*=(*fact[i]*)−1 mod *M* = (*fact[i]*)*M*−2 mod *M*

Chú ý rằng *fact[i]* ≡ 0 (mod *M*) ∀ *i* ≥ *M* nên ta chỉ tính *fact[i]* và *ifact[i]* với 0 ≤ *i* ≤ *M−1*.

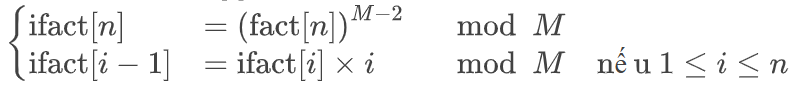
* Ta sẽ tính mảng *fact[i]* như sau:



* Tiếp theo ta sử dụng thuật toán lũy thừa nhanh để tính *ifact[n]*:

*ifact[n]*=(*fact[n]*)M−2 mod *M* với độ phức tạp O(logM).

* Còn mảng *ifact[i]* thì tính như sau:



* Cuối cùng, C(*n, k*)​ được tính như sau:



Độ phức tạp thuật toán O(*N*).

const int MOD = 1e9 + 7;

const int N = 1e6;

int fact[N + 5], ifact[N + 5];

// Hàm lũy thừa nhanh

long long binpow(long long a, long long b) {

long long ans = 1;

while (b > 0){

if (b % 2) ans = ans \* a % MOD;

a = a \* a % MOD;

b /= 2;

}

return ans;

}

// Chuẩn bị

void prepare(){

// Tính fact[]

fact[0] = 1;

for (int i = 1; i <= N; i++)

fact[i] = 1LL \* fact[i - 1] \* i % MOD;

// Tính ifact[]

ifact[N] = binpow(fact[N], MOD - 2);

for (int i = N - 1; i >= 1; i--)

ifact[i] = 1LL \* ifact[i + 1] \* (i + 1) % MOD;

}

// Hàm tính nCk

int C(int n, int k){

if (k > n) return 0;

return (1LL \* fact[n] \* ifact[k] % MOD) \* ifact[n - k] % MOD;

}

int main(){

prepare();

// Truy vấn

int q; cin >> q;

while (q--){

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

cout << C(n, k) << '\n';

}

}

# **3. BÀI TẬP CƠ BẢN**

## **Bài 1: Tính tổ hợp chập k của N (Mã bài - CKN3)**

**Đề bài**

Tính tổ hợp chập *k* của *n* phần tử C(*n, k*) theo modulo 109+7.

**Dữ liệu**

* + Một dòng duy nhất chứa hai số nguyên *n* và *k* (*k* < *n* ≤ 106).

**Kết quả**

* + Gồm một số nguyên không âm duy nhất là kết quả của bài toán

***Ví dụ***

|  |  |
| --- | --- |
| **CKN3.INP** | **CKN3.OUT** |
| 1000000 500000 | 996692777 |

**Hướng dẫn**

Sử dụng cách tính tổ hợp theo modulo của *M* (*M*=109+7)

## **Bài 2. Hệ số nhị thức (Mã bài – HSNT)**

Tính toán **n** hệ số nhị thức chia lấy dư cho *109+7*. Một hệ số nhị thức  có thể được tính toán bằng công thức . Chúng ta giả định rằng **a** và **b** là những số nguyên và 0 ≤ *b* ≤ *a*.

**Dữ liệu**

* Dòng đầu tiên chứa một số nguyên **n**: số lượng test cases.
* **n** dòng sau, mỗi dòng chứa hai số nguyên **a** và **b**.

**Kết quả**

* In mỗi hệ số nhị thức chia lấy dư cho *109+7*.

**Ràng buộc**

* 1 ≤ *n* ≤105
* 0 ≤ *b* ≤ *a* ≤ 106

***Ví dụ***

|  |  |
| --- | --- |
| **HSNT.INP** | **HSNT.OUT** |
| 3  5 3  8 1  9 5 | 10  8  126 |

**Hướng dẫn**

Sử dụng cách tính tổ hợp theo modulo của *M* (*M*=109+7)

## **Bài 3: Tạo xâu (Mã bài – TAOXAU)**

**Đề bài**

Cho một xâu, hãy tính toán số lượng các xâu khác nhau có thể được tạo thành bằng các kí tự của nó.

**Dữ liệu**

* Một dòng duy nhất chứa một xâu độ dài **n**. Mỗi kí tự nằm giữa **a - z**.

**Kết quả**

* In số lượng xâu khác nhau chia lấy dư cho *109+7*.

**Ràng buộc**

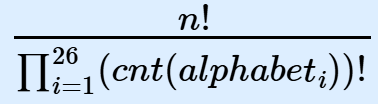
* 1 ≤ n ≤ 106

***Ví dụ***

|  |  |
| --- | --- |
| **TAOXAU.INP** | **TAOXAU.OUT** |
| aabac | 20 |

**Hướng dẫn thuật toán**

Đây là bài toán tổ hợp cơ bản. Số cách sắp xếp các kí tự này là:



## **Bài 4: Dãy ngoặc (Mã bài – BRACKET)**

**Đề bài**

Tính toán số dãy ngoặc hợp lệ có độ dài *n*. Ví dụ: khi *n* = 6, có 5 dãy:

* ()()()
* ()(())
* (())()
* ((()))
* (()())

**Dữ liệu**

* Một dòng duy nhất có số nguyên *n*.

**Kết quả**

* In số lượng dãy chia lấy dư cho 109+7.

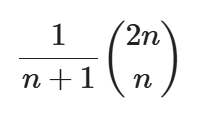
**Giới hạn:** 1 ≤ *n* ≤ 106

***Ví dụ***

|  |  |
| --- | --- |
| **BRACKET.INP** | **BRACKET.OUT** |
| 6 | 5 |

**Hướng dẫn**:

Số lượng dãy ngoặc đúng chỉ với một loại ngoặc () có thể được tính bằng các số Catalan. Số lượng dãy ngoặc đúng có độ dài 2*n* (với *n* cặp ngoặc) là:



## **Bài 5: Đếm vòng cổ (Mã bài – NECKLACES)**

**Đề bài**

Đếm số lượng vòng cổ khác nhau bao gồm *n* hạt, và mỗi hạt có *m* màu sắc. Hai vòng cổ được coi là khác nhau nếu không thể xoay một vòng để nó giống hệt với vòng còn lại.

**Dữ liệu**

* Một dòng duy nhất chưa hai số *n* và *m*: Số lượng hạt và số màu.

**Kết quả**

* In ra một số nguyên là số lượng vòng cổ khác nhau modulo 109+7.

**Giới hạn:** 1 ≤ *n*, *m* ≤ 106

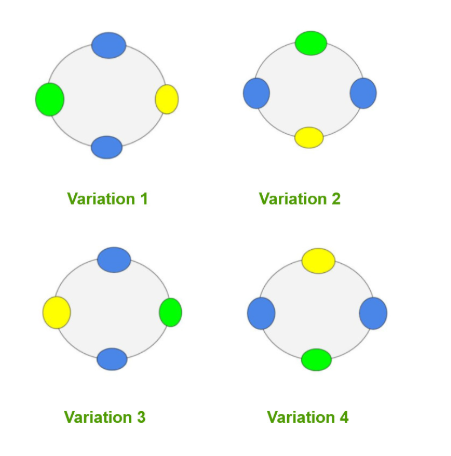
***Ví dụ***

|  |  |
| --- | --- |
| **NECKLACES.INP** | **NECKLACES.OUT** |
| 4 3 | 24 |

**Hướng dẫn**

Bây giờ giả sử chúng ta có **N = 4** viên đá với **M = 3** màu, thì:

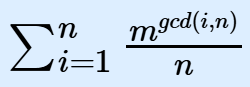
Vì chúng ta có **N** viên đá, nên chúng ta có **N** biến thể có thể có của mỗi chiếc vòng cổ bằng cách xoay:



**Nhận xét:** Có N cách để thay đổi vị trí của chiếc vòng cổ vì chúng ta có thể xoay nó từ **0** đến **N – 1** lần.

1. Có **MN** cách để tô màu cho một chiếc vòng cổ. Nếu số vòng quay là 0, thì tất cả **MN** các cách đều khác nhau, tức là tạo ra được **MN** vòng cổ khác nhau.
2. Nếu số vòng quay là 1 thì chỉ có **M** vòng cổ sẽ khác nhau hoàn toàn từ **MN** cách tô màu.
3. Nhìn chung, nếu số vòng quay là **K** thì **Mgcd(K,N**) vòng cổ sẽ vẫn giữ nguyên như vậy trong mọi **Mgcd(K,N**) trường hợp.

Do đó, đối với tổng số chuỗi hạt riêng biệt gồm **N** hạt sau khi tô màu bằng **M** màu là tổng của tất cả các chuỗi hạt riêng biệt tại mỗi lần quay. Kết quả đưa ra là:



## **Bài 6: Chia táo (Mã bài – CHIATAO)**

Có **n** đứa trẻ và **m** quả táo sẽ được chia cho chúng. Đếm số cách chia có thể được thực hiện. Ví dụ: nếu **n**=3 và **m**=2, có 6 cách: [0,0,2], [0,1,1], [0,2,0], [1,0,1], [1,1,0] và [2,0,0].

**Dữ liệu**

* Một dòng duy nhất có hai số nguyên **n** và **m**.

**Kết quả**

* In số lượng cách chia lấy cho *109+7*.

**Ràng buộc**

* 1 ≤ *n, m* ≤ 106

***Ví dụ***

|  |  |
| --- | --- |
| **CHIATAO.INP** | **CHIATAO.OUT** |
| 3 2 | 6 |

**Hướng dẫn thuật toán**

Bài toán này tương đương với bài toán: Đếm số nghiệm nguyên của phương trình tuyến tính có N biến như sau: **x1 + x2 +…. + xn = m,** trong đó **x1, x2, …, xn** là các số nguyên không âm.

Bài toán này có thể giải bằng khái niệm Hoán vị và Tổ hợp. Sau đây là các công thức trực tiếp để tìm nghiệm không âm của phương trình: .

# **4. BÀI TẬP VẬN DỤNG**

## **Bài 1. HTDEN - Thiết kế hệ thống đèn (Đề thi chính thức DHBB 2022)**

**Đề bài**

Để trang trí hội trường cho buổi khai mạc Duyên Hải năm 2022, Ban tổ chức đã sử dụng ***n*** đèn, nếu coi mỗi đèn là một đỉnh của đồ thị và dây nối giữa các đèn là cạnh thì hệ thống đèn tương ứng như một cây. Ban tổ chức đã ghi chép lại dãy các thông tin ***b*1​, *b*2​, …, *bn*​**, trong đó ***bi***(1 ≤ *i* ≤ *n*) tương ứng là số dây nối với đèn thứ ***i***, hay nói một cách khác ***bi***​ là bậc của đỉnh ***i***. Vì một lí do nào đó, Ban tổ chức đã vô tình làm mất thông tin của một số phần tử trong dãy thông tin ***b*1​, *b*2​, …, *bn*​**. Để khôi phục lại các thông tin, Ban tổ chức đã nhờ đến Đào Quang Thái Dương (là cựu học sinh Chuyên Trần Phú, Huy chương Đồng APIO 2020). Rất nhanh chóng, Dương đã đếm được số lượng cách khác nhau điền thông tin vào các phần tử bị khuyết để nhận được dãy vẫn là dãy bậc của một cây nào đó.

Yêu cầu: Gọi **s** là số lượng cách điền thỏa mãn, hãy tính ***s*** % (109+7), trong đó % là phép chia lấy dư để kiểm tra kết quả của Dương.

**Dữ liệu**

* Dòng đầu chứa số nguyên dương ***n***;
* Dòng thứ hai chứa **n** số nguyên ***b*1​, *b*2​, …, *bn*​**, trong đó 1 ≤ *bi* ​≤ *n* −1 hoặc *bi*​ = −1 cho biết thông tin đỉnh ***i*** bị mất.

**Kết quả**

* In ra một số nguyên là giá trị *s* % (109+7).

**Giới hạn**

* Subtask 1 (20% số điểm): *n* ≤ 6;
* Subtask 2 (20% số điểm): *n* ≤ 10;
* Subtask 3 (20% số điểm): *n* ≤ 100;
* Subtask 4 (20% số điểm): *n* ≤ 104;
* Subtask 5 (20% số điểm): *n* ≤ 106.

***Ví dụ***

|  |  |  |
| --- | --- | --- |
| **HTDEN.INP** | **HTDEN.OUT** | **Giải thích** |
| 3  -1 -1 1 | 2 | Có hai cách điền thông tin:   * Cách 1: 1 2 1 * Cách 2: 2 1 1 |

**Hướng dẫn giải thuật**

1. **Tính tổng bậc**:
   * Tổng số bậc tối đa của cây (all) là (n - 1) \* 2 (vì cây có n - 1 cạnh và mỗi cạnh đóng góp 2 bậc).
   * Duyệt qua danh sách a[i] để tính số bậc đã biết và đếm số lượng đèn có bậc bị thiếu (cnt).
2. **Tính tổng bậc cần bổ sung**:
   * Tổng số bậc cần bổ sung để hoàn thành cây là rev = all - sum(a[i] != -1).
3. **Bài toán tổ hợp**:
   * Số cách điền giá trị cho các bậc bị thiếu tương đương với việc chia rev tổng số bậc cần bổ sung vào cnt đèn.
   * Đây là bài toán chọn tổ hợp có lặp, được tính bằng công thức: C(rev−1,cnt−1) (vì cần phân phối rev giá trị cho cnt phần tử ).

## **Bài 2. LEUTHI - Lều thi (Đề thi chính thức DHBB 2019)**

**Đề bài**

Trong một kỳ thi Olympic Tin học đồng đội có ***n*** Đội học sinh tham gia. Ban Tổ chức bố trí mỗi đội làm việc trong một lều riêng biệt. Các đội và các lều được đánh số từ 1 đến ***n***. Ngày đầu tiên thử nghiệm làm quen với hệ thống chấm điểm tự động, Đội thứ ***i*** được phân vào làm việc ở lều thứ ***i***. Ở buổi thi chính thức, các đội tiến hành bốc thăm xác định lều mình sẽ làm việc. Dĩ nhiên, việc bốc thăm cũng được tin học hoá: Trước sự chứng kiến của các Đội trưởng, Ban Tổ chức kích hoạt chương trình tạo một hoán vị ***P* = (*p*1​, *p*2​, ..., *pn*​)** các số từ 1 đến ***n***. Hoán vị ***P*** được hiển thị công khai trên màn hình lớn trong hội trường và các đội theo đó đi vào lều của mình – đội ***i*** sẽ vào lều ***pi***​. Không ai nghi ngờ về tính trung thực và khách quan của kết quả bốc thăm. Nhưng tâm lý chung ai cũng thầm mong ước may mắn được về lại chính lều nơi ban đầu mình thử nghiệm hệ thống.

**Yêu cầu** : Hãy xác định trong số ***n*!** khả năng xuất hiện ***P*** có bao nhiêu khả năng có đúng ***k*** đội may mắn.

**Dữ liệu**

* Gồm một dòng chứa 2 số nguyên ***n*** và ***k***, các số cách nhau ít nhất một dấu cách (1 ≤ *n* ≤ 105, 0 ≤ *k* ≤ *n*).

**Kết quả**

* Đưa ra một số nguyên là kết quả tìm được lấy số dư cho 109+7.

**Giới hạn**

* Subtask 1 (50% số điểm): *n* ≤ 10.
* Subtask 2 (50% số điểm): không có điều kiện gì thêm

***Ví dụ***

|  |  |
| --- | --- |
| LEUTHI.INP | LEUTHI.OUT |
| 4 2 | 6 |

**Giải thuật**

Sử dụng quy hoạch động (Dynamic Programming - DP) để tính số cách sắp xếp ***n*** đội sao cho không có đội nào được khớp với vị trí ban đầu:

dp[i]=(dp[i−1]+dp[i−2])\*(i−1)mod MOD

Với:

* dp[0]=0, dp[1]=0, dp[2]=1.
* dp[i] đại diện cho số cách hoán vị ***i*** đội sao cho không có đội nào được khớp với vị trí ban đầu.

**Tính số cách thỏa mãn bài toán**

* Với bài toán trên, có chính xác ***k*** đội được đặt đúng vị trí ban đầu, điều đó có nghĩa:
  + Chọn ***k*** đội từ ***n*** đội để đặt đúng vị trí: C(n, k)
  + Sắp xếp ***n−k*** đội còn lại sao cho không có đội nào khớp vị trí: dp[n−k]
* Kết hợp hai yếu tố trên: Kết quả = C(n, k) \* dp[n−k] mod (109+7)

## **Bài 3. MATRAN - Ma trận (Đề chọn ĐTQG Đà Nẵng năm 2020-2021)**

**Đề bài**

Ma trận kích thước ***m*∗*n***, các hàng được đánh số từ **1** đến ***m*** và các cột được đánh số từ **1** đến ***n***. Ở trong ma trận có một cái hồ, cái hồ sẽ trải dài trong hình chữ nhật con của trận có góc trái trên là (***h1​, c1***​) và góc phải dưới là (***h*2​, *c2***​). Ngọc sẽ di chuyển từ ô (1, 1), vì bản chất ngại di chuyển, Ngọc sẽ chỉ đi xuống ô bên dưới hoặc đi qua ô bên phải (từ ô (***i*, *j***) Ngọc sẽ chọn đi ô đến ô (***i*+1, *j***) hoặc (***i*, *j*+1**). Dù ngại di chuyển nhưng lại tò mò, Ngọc muốn tính xem có bao nhiêu lộ trình khác nhau để di chuyển từ ô (1,1) đến ô (***m*, *n***) thông qua cách đi đã nêu trên.

Đảm bảo rằng cái hồ sẽ không chứa 2 ô (1, 1) và (***m*, *n***). Vì số lộ trình là quá lớn, các bạn hãy chia dư 109+7 trước khi đưa cho Ngọc nhé.

**Yêu cầu**: In ra số lộ trình sau khi mod 109+7.

**Dữ liệu**

* Dòng đầu tiên là số tự nhiên *m* và *n* (1 ≤ *n*, *m* ≤ 105) là kích thước của ma trận
* Dòng thứ hai chứa 4 nguyên dương *h1​, c1​, h2​, c2*​ (1 ≤ *h*1 ​≤ *h*2 ​≤ *m*, 1 ≤ *c*1 ​≤ *c*2 ​≤ *n*) là góc trái trên và phải dưới của cái hồ.

**Kết quả**

* Ghi ra một số nguyên duy nhất - số cách chia.

**Giới hạn**

* Subtask 1 (20% số điểm): 1 ≤ *m*, *n* ≤ 103
* Subtask 2 (30% số điểm): *h*1​ = *h*2​, *c*1 ​= *c*2​ và 1 ≤ *m*, *n* ≤ 105
* Subtask 3 (50% số điểm): không có thêm dữ kiện nào cả.

Ví dụ

|  |  |  |
| --- | --- | --- |
| **MATRAN.INP** | **MATRAN.OUT** | **Giải thích** |
| 3 4  2 2 2 3 | 2 | Ở ví dụ 1, ma trận có dạng sau  0000  0110  0000  Có 2 lộ trình, đi từ (1,1) → (1,2) → (1,3) → (1,4) → (2,4) → (3,4)  hoặc (1,1) → (2,1) → (3,1) → (3,2) → (3,3) → (3,4). |
| 1 5  2 2 2 2 | 0 | Ở ví dụ 2, ma trận có dạng sau  01000  Rõ ràng không có cách đi nào mà không đi qua hồ. |

**Giải thuật**

Bài toán yêu cầu tính số lượng đường đi từ ô (*1, 1*) đến ô (*m, n*) trong một ma trận *m×n*, với điều kiện không đi qua vùng hồ hình chữ nhật được chỉ định. Kết quả cần trả về modulo 109+7.

**1. Xác định số đường đi trong ma trận**

Số đường đi từ ô (*1, 1*) đến ô (*m, n*) mà không có chướng ngại vật được tính bằng công thức tổ hợp: *C(n + m − 2, n − 1)*

Công thức này dựa trên việc chọn *n* − 1 bước đi xuống trong tổng số *n + m* − 2 bước.

**2. Xử lý vùng hồ cấm đi**

Vùng hồ được định nghĩa bằng các toạ độ (h1​,c1​) là góc trái trên và (h2​,c2​) là góc phải dưới.

* Nếu vùng hồ bao phủ toàn bộ (1,1) hoặc (m,n), in ra 0.
* Nếu không, cần loại bỏ các đường đi đi qua vùng hồ.

Để loại bỏ, thực hiện như sau:

1. **Loại bỏ đường đi chặn bởi cạnh trên của hồ:**
   * Tính số đường đi từ (1,1) đến các ô bên ngoài hồ nhưng bị chặn bởi cạnh trên (h1​−1).
   * Công thức: cur=C(i,c1​−1)\*C(n−i+1,m−c1​+1)
   * Cộng dồn và trừ khỏi tổng số đường đi.
2. **Loại bỏ đường đi chặn bởi cạnh trái của hồ:**
   * Tương tự, tính số đường đi chặn bởi cạnh trái:

cur=C(h1​−1,j)\*C(m−h1​+1,n−j+1)

* + Trừ khỏi tổng số đường đi.

**3. Kết quả**

Số đường đi hợp lệ là tổng số đường đi ban đầu trừ đi các đường đi không hợp lệ qua hồ.

## **Bài 4. HOANVI - Hoán vị không bất động (Đề thi THT C Vòng Khu vực 2021)**

**Đề bài**

Huấn luyện viên của đội hành tinh ***Z*** biết rằng, đội Trái Đất đã nắm rõ các chỉ số thể lực và chỉ số kĩ thuật của các vận động viên đội mình, vì vậy ông ta quyết định thay đổi số áo nhằm làm sai lệch những tính toán của đội Trái đất.

Đội hành tinh ***Z*** có ***m*** vận động viên đánh số từ 1 tới ***m***, ban đầu vận động viên thứ ***i*** mang số áo là ***i***(1 ≤ *i* ≤ *m*).

Huấn luyện viên chọn ***T*** đoạn, đoạn thứ ***s***(1 ≤ *s* ≤ *T*) mô tả bằng cặp số *Ls*​, *Rs* ​(1 ≤ *Ls*​ ≤ *Rs* ​≤ *m*), rồi hoán vị số áo của các vận động viên (có thể cả ***m*** vận động viên) sao cho tất cả các vận động viên có số áo nằm trong một trong ***T*** đoạn phải mang số áo khác với số áo ban đầu của mình.

Cụ thể, với một vận động viên mang số áo ***i*** mà tồn tại *s* (1 ≤ *s* ≤ *T*) để *Ls* ​≤ *i* ≤ *Rs*​  
thì sau khi hoán vị vận động viên này phải mang số áo khác với số áo ban đầu của mình.

**Yêu cầu:** Hãy cho biết huấn luyện viên của đội hành tinh ***Z*** có bao nhiêu cách khác nhau để hoán vị số áo cho các vận động viên theo quy tắc trên, hai cách hoán vị số áo được gọi là khác nhau nếu có một vận động viên mang hai số áo khác nhau trong hai cách hoán vị.

**Dữ liệu**

Vào từ thiết bị vào chuẩn theo khuôn dạng:

* Dòng đầu chứa số nguyên dương ***m***, ***T***.
* Dòng thứ ***s***(1 ≤ *s* ≤ *T*) trong ***T*** dòng tiếp theo chứa hai số nguyên dương ***Ls*​, *Rs***​(1 ≤ *Ls* ​≤ *Rs* ​≤ *m*).

**Kết quả**

* Ghi ra thiết bị ra chuẩn gồm một dòng chứa một số nguyên duy nhất là số dư của phép chia: số cách hoán vị số áo cho (109+7).

**Giới hạn**

* Subtask 1 (20% số điểm): *m* ≤ 10 và *T* = 0;
* Subtask 2 (20% số điểm): *m* ≤ 10 và *T* = 1;
* Subtask 3 (20% số điểm): *m* ≤ 103 và *T* = 1; *R1​−L1*​≤ 10;
* Subtask 4 (20% số điểm): *m* ≤ 103 và *T* ≤ 103;
* Subtask 5 (20% số điểm): *m* ≤ 105 và *T* ≤ 103.

***Ví dụ***

|  |  |  |
| --- | --- | --- |
| **HOANVI.INP** | **HOANVI.OUT** | **Giải thích** |
| 3 1  1 2 | 3 | Có 3 cách hoán vị là: 1) 2 1 3 2) 2 3 1 3) 3 1 2 |

**Giải thuật**

**Phát biểu lại đề bài**

Cho 2 số nguyên **n** và **q.**

Cho **q** đoạn [**li, ri**].

Xét tập **S** gồm các điểm 1, 2, …**n** mà thuộc vào ít nhất 1 trong **q** đoạn.

Gọi điểm cố định là điểm ở vị trí **p** cũng có giá trị **p** của một hoán vị đang xét.

Đếm số hoán vị bậc **n** sao cho không tồn tại điểm cố định trong tập S

Không mất tính tổng quát, đưa các điểm trong tập **S** thành các điểm 1, 2, …, **k** với *k = |S|*. Bài toán trở thành đếm số hoán vị bậc **n** có **k** điểm đầu không cố định.

**Hướng dẫn Subtask 1-2**

Ý tưởng đơn giản là thử từng hoán vị, và kiểm tra xem có tồn tại điểm cố định trong

k phần tử đầu hay không

Để thử từng hoán vị thì có thể xài C++ std::next\_permutation

ĐPT **O(n!)**

**Hướng dẫn Subtask 3-4**

Đặt hàm quy hoạch động f[n][k] là số hoán vị bậc **n** không tồn tại điểm cố định trong **k** phần tử đầu.

Trường hợp **n**=0, thì không có hoán vị nào, nên f[0][k]=0

Trường hợp **k**=0, thì cần đếm số hoán vị bậc **n**, nên f[n][0]=n!

Ngược lại, ta có:

f[n][k]=f[n][k−1] − f[n−1][k−1]

* f[n][k−1] là số cách chọn giá trị tại vị trí **k**
* f[n−1][k−1] là số cách chọn giá trị tại vị trí **k** mà có thêm một điểm cố định

Kết quả bài toán là f[n][k].

ĐPT **O**(N x K)

**Hướng dẫn Subtask 5**

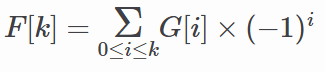
Gọi G[x] là số hoán vị bậc **n** có ít nhất **x** điểm cố định trong **k** vị trí đầu tiên

Ta có G[x]= C(k, x) × (n−x)!

* Với C(k, x) là số cách chọn một hoán vị có **x** điểm cố định trong **k** phần tử đầu.
* Và (n−x)! là số cách chọn **n−x** phần tử còn lại chưa được chọn (có thể có điểm cố định)

Gọi F[k] là số hoán vị bậc **n** không có bất cứ điểm cố định nào xuất hiện tại **k** vị trí đầu tiên

* Dễ thấy được F[k] = G[0]−G[1]+G[2]−G[3]+G[4]−...±G[k]
* Một cách tổng quát



Kết quả bài toán là F[k].

ĐPT **O(N)**

## **Bài 5. ROUTES - Số đường đi (Nguồn bài của thầy Đỗ Đức Đông)**

**Đề bài**

Cho lưới ô vuông kích thước *n*×*m*, ô (1, 1) ở góc dưới trái, ô (*n*, *m*) – trên phải. Có *k*  
ô chứa chướng ngại vật, ô thứ *i* ở tọa độ (*xi*​, *yi*​), 1 ≤ *xi* ​≤ *n*, 1 ≤ *yi* ​≤ *m*, *i* = 1÷*k*. Không có chướng ngại vật ở ô (1, 1) và (*n*, *m*).

Rô bốt xuất phát từ ô (1, 1), ở mỗi bước được chuyển sang ô kề cạnh bên phải hoặc  
bên trên nếu ô tới không chứa chướng ngại vật.

Hãy xác định số lượng đường rô bốt có thể đi từ ô (1, 1) đến ô (*n*, *m*) và đưa ra số  
lượng theo mô đun *p*, trong đó *p* – số nguyên tố.

**Dữ liệu**

* Dòng đầu tiên chứa 4 số nguyên *n*, *m*, *k*, *p* (*n*, *m* ≤ 105, 0 ≤ *k* ≤ 100, 2×*max(m*, *n)* < *p* < 2×109),
* Nếu *k* > 0, dòng thứ *i* trong *k* dòng tiếp theo chứa 2 số nguyên *xi*​, *yi*​ (1 ≤ *xi* ​≤ *n*, 1 ≤ *yi* ​≤ *m*).

**Kết quả**

* Đưa ra một số nguyên không âm – số lượng đường tìm được theo mô đun *p*.

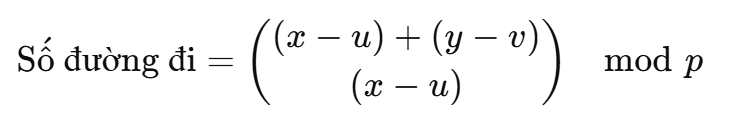
***Ví dụ***

|  |  |
| --- | --- |
| ROUTES.INP | ROUTES.OUT |
| 5 6 3 101  2 2  3 5  4 2 | 25 |

**Giải thuật**

**Tính số đường đi giữa hai ô bất kỳ**:

* Để đi từ ô (u, v) đến (x, y), cần thực hiện:
  + x−u bước xuống.
  + y−v bước sang phải.
* Công thức:

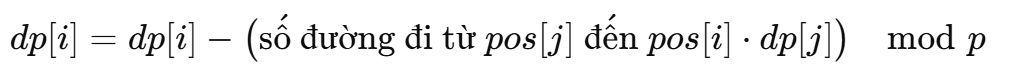


**Sắp xếp danh sách các ô cần xử lý**:

* Danh sách các ô bao gồm các chướng ngại vật và đích đến (n, m).
* Sắp xếp danh sách theo tọa độ (trước theo hàng, sau theo cột).

**Quy hoạch động để tính số đường đi hợp lệ**:

* dp[i]: Số đường đi từ (1,1) đến ô pos[i], tránh các ô chướng ngại trước đó.
* Công thức: dp[i] = số đường đi từ (1,1) đến pos[i]
* Trừ đi số đường đi qua bất kỳ ô pos[j] trước đó (nằm bên trái hoặc phía trên) như sau:



Kết quả cuối cùng là giá trị dp[k], tức là số đường đi hợp lệ từ (1, 1) đến (*n, m*) tránh ***k*** ô chướng ngại vật.

## **Bài 6: PASS - Mật khẩu (Đề thi chính thức DHBB 2021)**

**Đề bài**

Do dịch Covid-19, hai bạn Hồng và Chi không được đi học và gặp nhau nhưng hai bạn vẫn thường xuyên nhắn tin cho nhau. Một lần, Hồng muốn gửi mật khẩu tham gia lớp học online cho Chi nhưng không muốn em Phúc tò mò và biết được. Theo ý tưởng giấu tin trong ảnh, Hồng quyết định sẽ giấu mật khẩu vào trong đoạn văn bản gửi cho Chi. Cụ thể, với một văn bản mà Hồng gửi cho Chi được biểu diễn bằng xâu ký tự **T=t1t2 … tn** (gồm **n** ký tự, mỗi ký tự thuộc ′a′ đến ′z′) và dãy số nguyên **a1,a2,…,am** (1 ≤ a1<a2< … <am ≤ n) là dãy số mà hai bạn đã thống nhất thì mật khẩu là một xâu P=ta\_1ta\_2 … ta\_m, là xâu độ dài **m** nhận được bằng cách ghép lần lượt các ký tự ở các vị trí **a1,a2,…,am**. Ví dụ, T= ′missyouuu′ và dãy số a1=2,a2=3,a3=5,a4=6,a5=8 thì mật khẩu là P= “isyou”.

Hồng nhanh chóng nhận ra rằng, với một xâu T và mật khẩu P sẽ tồn tại nhiều dãy số để xác định mật khẩu. Ví dụ, một dãy số khác a1=2,a2=4,a3=5,a4=6,a5=7 cũng xác định được mật khẩu P= “isyou” trong xâu T= ′missyouuu′.

Trong quá trình gửi, xâu **T** sẽ được mã hóa theo phương thức **RLE** (Run Length Encoding). Nghĩa là, một xâu **T** chỉ gồm các ký tự ‘a’ đến ‘z’ được mã hóa thành xâu TE (chỉ gồm các ký tự ‘a’ đến ‘z’ và ký tự ‘0’ đến ‘9’) bằng cách đi từ trái sang phải, mã hoá dãy các ký tự liên tiếp giống nhau trong **T** thành ký tự đại diện và số lượng.

Ví dụ, xâu T= ′missyouuuuuuuuuu′ thì TE= ′m1i1s2y1o1u10′.

**Yêu cầu**: Cho xâu **TE** (là mã hóa của xâu **T**) và xâu mật khẩu **P**, gọi **R** là số lượng dãy số khác nhau có thể xác định được mật khẩu **P** trong xâu **T**. Hãy tính **R** chia dư cho **109+7**.

**Dữ liệu**

* + Dòng đầu chứa hai số nguyên dương **n, m**;
  + Dòng thứ hai chứa một xâu là mã hóa của xâu **T**
  + Dòng thứ ba chứa một xâu là xâu **P**.

**Kết quả**

* + Ghi ra một số nguyên duy nhất là số **R** chia dư cho **109+7**

**Ràng buộc**

* + Subtask 1 (20% số điểm): *n ≤ 20, m=1*;
  + Subtask 2 (20% số điểm): *n ≤ 20, m < n*;
  + Subtask 3 (20% số điểm): *n ≤ 105, m=3*;
  + Subtask 4 (20% số điểm): *n ≤ 105, m ≤ 30*;
  + Subtask 5 (20% số điểm): *n ≤ 109, m ≤ 30* và xâu mã hóa của xâu **T** có độ dài không vượt quá 105.

***Ví dụ***

|  |  |
| --- | --- |
| **PASS.INP** | **PASS.OUT** |
| 9 5  m1i1s2y1o1u3  isyou | 6 |
| 11 3  m1i1s2i1s2i1p2i1  isi | 14 |

**Hướng dẫn thuật toán**

Ý tưởng QHĐ: Ta chia xâu *T* thành các khối có ký tự giống nhau (RLE đã tự làm việc chia này), gọi *f*(*i*,*j*) là số cách tạo đc *j* ký tự đầu tiên của xâu *P* bằng *i* khối đầu tiên của xâu *T*. Ta có *f*(0,0)=1.

Từ trạng thái *f*(*i*,*j*); ta tiến hành cập nhật cho các trạng thái QHĐ khác theo ý tưởng như sau: Giả sử khối *i*+1 sẽ phủ tiếp *k* ký tự tiếp theo, điều kiện để có được điều này là các ký tự từ vị trí *j*+1 tới *j*+*k* của *P* phải bằng ký tự của khối *i*+1 của xâu *T*. Khi đó ta có *f*(*i*+1, *j*+*k*)+=*f*(*i*,*j*)∗*C*(*k*, *w*(*i*+1)) với *w*(*i*+1) là số ký tự của khối *i*+1 và *C*(*k*, *w*) là tổ hợp chập *k* của *w* phần tử. Tính *C*(*k*,*w*) trong *O*(1) một cách đơn giản: Nhận thấy khi code ta sẽ for biến *k* tăng dần từ 1 và cập nhật dần *f*(*i*+1, *j*+*k*)+=*f*(*i*, *j*)∗*C*(*k*, *w*(*i*+1)). Như thế, bản chất là ta cần tính các giá trị *C*(1, *w*),*C*(2, *w*),*C*(3, *w*), ... Ta có công thức: *C*(*k*, *w*)=*C*(*k*−1, *w*) ∗ (*w*−*k*+1) / *k*. Sử dụng công thức này, ta dùng một biến lưu lại giá trị *C*(*k*, *w*); khi *k* tăng 1, ta có thể cập nhật lại giá trị này trong *O*(1) mà không phải lưu thêm bất kỳ mảng phụ nào.

## **Bài 7: GIEOXUCXAC - Gieo xúc xắc**

**Đề bài:**

Có một cục xúc xắc có **6** mặt. Gieo **n** lần. Tính xem có bao nhiêu cách gieo mà tích các lần gieo bằng **m**.

**Dữ liệu vào:**

* Gồm một dòng duy nhất chứa hai số nguyên dương **n** và **m**.

**Kết quả:**

* Chứa một số nguyên duy nhất là số cách tìm được chia lấy phần dư cho **109 + 7**.

**Giới hạn**: **n** ≤ 100, **m** ≤ 1018

Ví dụ:

|  |  |  |
| --- | --- | --- |
| **GIEOXUCXAC.INP** | **GIEOXUCXAC.OUT** | **Giải thích** |
| 2 12 | 4 | Có bốn cách để đạt được tích bằng 12 tương ứng với các kết quả gieo (2,6), (3,4), (4,3) và (6,2). |
| 8 450 | 2520 |  |

**Hướng dẫn thuật toán**

Nhận xét:

Ví dụ: m = 11, 17, 21, ... 🡪 đáp số = 0

Nhận xét 1: Nếu m có ước nguyên tố khác 2, 3, 5 🡪 đáp số = 0

2 = 2, 3 = 3, 4 = 22, 5 = 5, 6 = 2 \* 3

Tích của **n** lần gieo có dạng là **2a \* 3b \* 5c =** **m**

Ví dụ 4 \* 6 \* 5 = 22 \* (2 \* 3) \* 5 = 23 \* 31 \* 51.

Ta quy về bài toán tính tổng: theo bậc của 2, 3, 5.

Đầu tiên, phân tích **m = 2a \* 3b \* 5c** trong O(log m).

Có 2 cách tính:

**Cách 1: Sử dụng quy hoạch động với ĐPT O(n4)**

Định nghĩa: dp[i][a][b][c] = số cách gieo i lần và tích = 2a \* 3b \* 5c

Chuyển trạng thái: từ i -> i + 1

(i, a, b, c) -- gieo 1 🡪 (i + 1, a , b, c)

(i, a, b, c) -- gieo 2 🡪 (i + 1, a + 1, b, c)

(i, a, b, c) -- gieo 3 🡪 (i + 1, a , b + 1, c)

(i, a, b, c) – gieo 4 🡪 (i + 1, a + 2, b, c);

(i, a, b, c) – gieo 5 🡪 (i + 1, a, b, c + 1);

(i, a, b, c) – gieo 6 🡪 (i + 1, a+1, b+1, c); …

**Cách 2: Sử dụng toán tổ hợp với ĐPT O(n)**

Dùng a1 số 1, a2 số 2, a3 số 3, a4 số 4, a5 số 5 và a6 số 6 cho **n** lần gieo.

Tức là**: a1 + a2 + a3 + a4 + a5 + a6 = n**

Điều kiện: Bậc 2: a2 + 2 \* a4 + a6 = a

Bậc 3: a3 + a6 = b

Bậc 5: a5 = c

Chọn vị trí cho a1 số 1: C(n, a1) 🡪 còn lại n – a1 vị trí trống

Chọn vị trí cho a2 số 2: C(n - a1, a2) 🡪 còn lại n - a1 - a2 vị trí trống

....

Công thức tổng quát tính số cách gieo: **C(n, a1) \* C(n - a1, a2) \* C(n - a1 - a2, a3) \* .... \* C(n - a1 - a2- a3 - a4 - a5, a6)**

Rút gọn thành: **n! / (a1! \* a2! \* ... \* a6!)**

* Lưu ý 1: a5 = c
* Lưu ý 2: a1 + a2 + ... + a6 = n. Nếu mà có a2, a3, a4, a6 thì xác định được a1 = n - a2 - a3 - ... - a6.
* Lưu ý 3: a3 + a6 = b

Nếu có a6 thì xác định được a3 = b - a6

* Lưu ý 4: a2 = a - 2 \* a4 - a6

Từ các lưu ý trên, ta chỉ cần hai vòng for để thử các giá trị của **a4** và **a6**, từ đó có thể tính được các **a** còn lại.

## **Bài 8: SONGACH - Sơn gạch**

**Đề bài**

Có **n** viên gạch xếp thành hàng trên mặt đất. Minh có **m** thùng sơn khác màu nhau. Minh sẽ sơn **n** viên gạch với **m** màu đó. Minh muốn sơn sao cho có **k** viên gạch đặc biệt mà màu của viên gạch đó khác với màu của viên gạch ở phía bên trái của nó (không tính viên gạch đầu tiên).

**Yêu cầu**: Đếm xem có bao nhiêu cách sơn gạch thỏa mãn yêu cầu của Minh? Hai cách sơn được gọi là khác nhau nếu có ít nhất một viên gạch được sơn khác màu với cách còn lại. Đáp án có thể là một số khá lớn nên bạn có thể ghi ra kết quả là số cách chia dư cho **998244353**.

**Dữ liệu**:

* Một dòng duy nhất chứa ba số nguyên **n, m** và **k** (1 ≤ *n, m* ≤ 2000; 0 ≤ *k* ≤ *n*-1) – số viên gạch, số màu và số viên gạch đặc biệt.

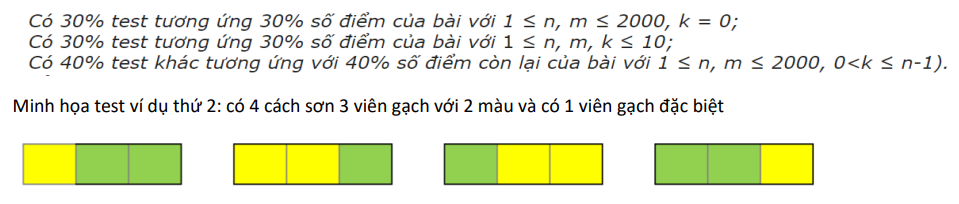
**Kết quả:**

* Gồm một số duy nhất là số cách thỏa mãn chia dư cho **998244353.**

Ví dụ:

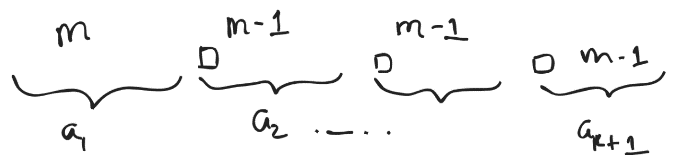
|  |  |
| --- | --- |
| SONGACH.INP | SONGACH.OUT |
| 3 3 0 | 3 |
| 3 2 1 | 4 |

**Ràng buộc:**



**Hướng dẫn thuật toán**

Ta có thể mô tả việc sơn **n** viên gạch theo yêu cầu như sau:



**k** viên gạch đặc biệt sẽ tạo ra **k**+1 đoạn.

**a1 + a2 + … + ak+1 = n (ai ≥ 1)**

Đoạn **a1** có **m** cách chọn màu sơn, các đoạn còn lại **a2, a3, …, ak+1** có **m**-1 cách chọn màu sơn (trừ 1 màu bên trái)

Số cách chia đoạn: có n-1 vị trí để đặt k viên gạch đặc biệt 🡪 **C(n-1, k)**

Đáp số bài toán: **C(n-1,k)\*m\*(m-1)k**

## **Bài 9: HOANVITRON - Hoán vị vòng tròn**

**Đề bài**

Một **hoán vị** có độ dài **n** là một dãy gồm **n** số nguyên khác nhau từ **1** đến **n** theo một thứ tự bất kỳ. Ví dụ, [2,3,1,5,4] là một hoán vị, nhưng [1,2,2] không phải là hoán vị (vì số 2 xuất hiện hai lần trong dãy) và [1,3,4] cũng không phải là hoán vị (vì n=3 nhưng có số 4 trong dãy).

Xét một hoán vị **p** có độ dài **n**. Ta xây dựng một đồ thị có kích thước **n** bằng cách sử dụng **p** như sau:

* Với mỗi chỉ số **1** ≤ **i** ≤ **n**, tìm chỉ số lớn nhất **j** sao cho 1 ≤ *j* < *i* và *pj* > *pi*​; sau đó thêm một cạnh không có hướng giữa nút **i** và nút **j**.
* Với mỗi chỉ số **1** ≤ **i** ≤ **n**, tìm chỉ số nhỏ nhất **j** sao cho *i* < *j* ≤ *n* và *pj*>*pi*​; sau đó thêm một cạnh không có hướng giữa nút **i** và nút **j**.

Trong trường hợp không tồn tại chỉ số **j** nào như vậy, không có cạnh nào được thêm. Lưu ý rằng chúng ta chỉ tạo các cạnh giữa các chỉ số tương ứng, chứ không phải giá trị tại các chỉ số đó.

Để làm rõ hơn, hãy xét một ví dụ với **n** = 4, và **p** = [3,1,4,2]; khi đó, các cạnh của đồ thị là (1,3), (2,1), (2,3), (4,3).

Một hoán vị **p** là **có chu trình** nếu đồ thị được tạo từ **p** có ít nhất một **chu trình đơn**.

Cho **n**, hãy tìm số lượng hoán vị có chu trình có độ dài **n**. Vì kết quả có thể rất lớn, hãy đưa ra kết quả theo modulo **109+7**.

**Dữ liệu**

* Dòng đầu tiên và duy nhất chứa một số nguyên **n** (3 ≤ *n* ≤ 106).

**Kết quả**

* In ra một số nguyên duy nhất 0 ≤ **x** < 109+7, là số lượng hoán vị có chu trình có độ dài **n** theo modulo 109+7.

Ví dụ:

|  |  |
| --- | --- |
| HOANVITRON.INP | HOANVITRON.OUT |
| 4 | 16 |
| 583291 | 135712853 |

**Ghi chú**

Có 16 hoán vị có chu trình với **n** = 4. [4,2,1,3] là một trong những hoán vị đó, có một chu trình độ dài 4: 4→3→2→1→4.

Các đỉnh **v1, v2,…, vk**​ tạo thành một **chu trình đơn** nếu các điều kiện sau thỏa mãn:

* **k** ≥ 3.
* **vi ≠ vj**​ với bất kỳ cặp chỉ số **i** và **j** nào (*1 ≤ i < j ≤ k*).
* **vi​** và **vi+1**​ có chung một cạnh đối với mọi *1 ≤ i < k*, và *v1*​ và *vk*​ có chung một cạnh.

**Hướng dẫn thuật toán**

**Cách 1**: Cố định **pk** = **n**

Mỗi số còn lại thì có 2 cách đặt: đặt vào trái **n** và đặt vào phải **n** vậy thì có 2n-1 cách.

**Cách 2**: Tổng C(n - 1, k - 1) cách chọn các số p1, p2, .., p(k - 1)

Đáp số = n! – (C(n - 1, 0) + C(n - 1, 1) + ... ) = n! - 2(n - 1)

## **Bài 10: BALL - Chọn bóng**

**Đề bài**

An có một hộp gồm **N** quả bóng, trên mỗi quả bóng được viết một số nguyên dương (các số này không nhất thiết phải khác nhau). An chọn 1 tập hợp **K** quả bóng, đánh dấu tập hợp này, đặt chúng lại vào hộp và tiếp tục quá trình này cho đến khi tất cả các tập hợp như trên được đánh dấu. Với mỗi tập hợp như vậy, An ghi lại hiệu giữa số lớn nhất và số bé nhất trong nó.

Lưu ý: 2 tập khác nhau nếu có ít nhất 1 quả bóng thuộc tập này và không thuộc tập kia.

Yêu cầu: Bạn hãy tính tổng tất cả các số được An ghi lại. In ra số dư của kết quả khi chia cho **109+7**.

**Dữ liệu**

* Dòng đầu chứa 2 số nguyên dương **N** và **K**.
* Dòng tiếp theo chứa **N** số được ghi trên bóng.
* Trong đó **1 ≤ K ≤ N**, các số trên bóng ≤ 109

**Kết quả**

* Ghi một số nguyên duy nhất là kết quả bài toán.

**Ràng buộc**

* Subtask 1 (20% số điểm): *N* ≤ 20.
* Subtask 2 (30% số điểm): *N* ≤ 5000.
* Subtask 3 (50% số điểm): *N* ≤ 105.

Ví dụ

|  |  |  |
| --- | --- | --- |
| **BALL.INP** | **BALL.OUT** | **Giải thích** |
| 4 2  10 20 30 40 | 100 | Có tổng cộng 6 cách chọn:  10 20  20 30  30 40  10 30  20 40  10 40  Tổng của chúng :  10+10+10+20+20+30=100 |

**Hướng dẫn thuật toán**

* SubTask 1: (N ≤ 20) 🡪 ĐPT O(2N)

Liệt kê tất cả tập con, rồi tính giá trị

* SubTask 2: (N ≤ 5000) 🡪 ĐPT O(N2)
  + Sắp xếp các quả bóng theo giá trị tăng dần
  + Cố định vị trí của **min** và **max** là **i** và **j** (*i* < *j*)
  + Số tập con nhận **ai** là **min** và **aj** là **max**: K - 2 số còn lại phải nằm trong khoảng [i + 1, i + 2, ..., j - 1]: C(j - i - 1, K - 2)

for i = 1 to N:

for j = i + 1 to N:

res += (a[j] - a[i]) \* C(j - i - 1, K - 2)

* SubTask 3: (N ≤ 105) 🡪 ĐPT O(NlogN)

Tổng (max - min) = Tổng max - Tổng min

Tổng max = với mỗi cách chọn một tập K phần tử, tính a\_max rồi cộng lại.

Cố định max là **aj**, thì K - 1 phần tử còn lại phải chọn từ [1, 2, .., j - 1]. Số cách chọn = C(j - 1, K - 1)

Tổng max = Tổng (a[j] \* C(j - 1, k - 1))

Tương tự, tổng min = Tổng (a[j] \* C(n - j, k - 1))

# **5. BÀI TẬP THÊM**

## **Bài 1: BRACKET2 - Dãy ngoặc 2**

**Đề bài**

Tính toán số lượng dãy ngoặc hợp lệ có độ dài **n** khi một *tiền tố* của dãy được cho trước.

**Dữ liệu**

* Dòng đầu vào đầu tiên có một số nguyên **n**.
* Dòng thứ hai có một xâu gồm **k** kí tự: tiền tố của chuỗi.

**Kết quả**

* In số lượng dãy chia lấy dư cho *109+7*.

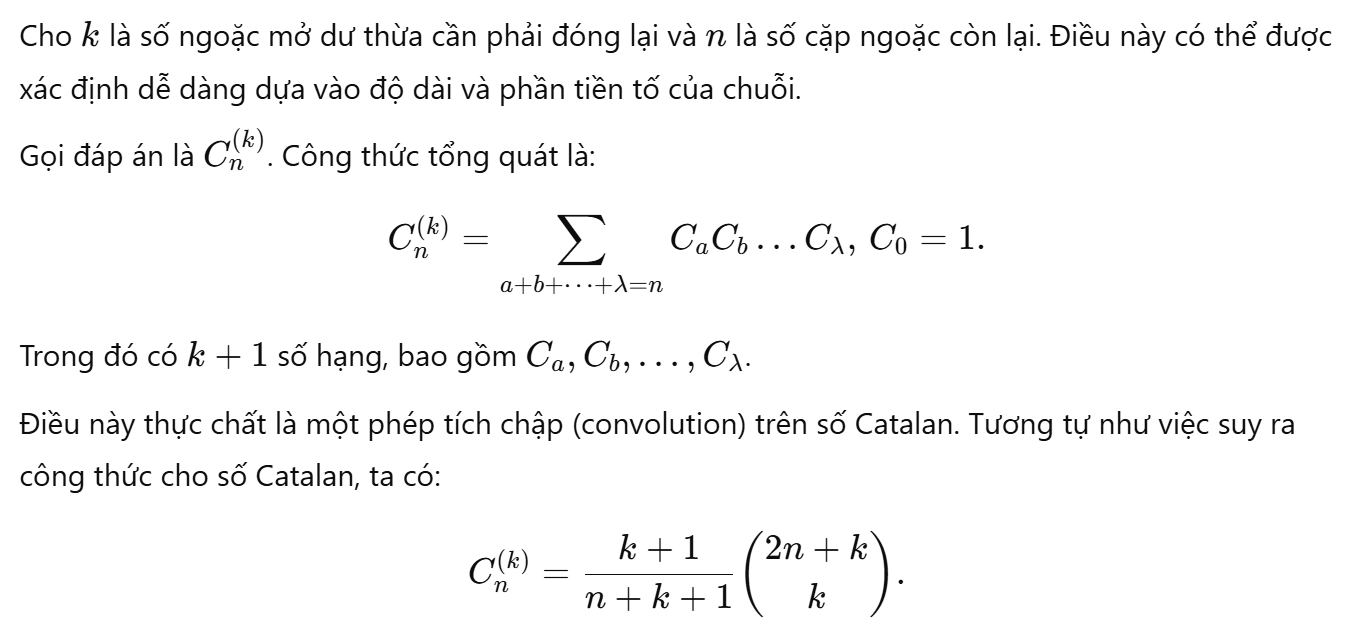
**Ràng buộc**

* 1 ≤ *k* ≤ *n* ≤ 106

***Ví dụ***

|  |  |  |
| --- | --- | --- |
| **BRACKET2.INP** | **BRACKET2.OUT** | **Giải thích** |
| 6  (() | 2 | Có hai dãy có thể: (())() và (()()). |

**Hướng dẫn**



## **Bài 2: TOMAU - Tô màu**

**Đề bài**

An có một cái tháp **N** tầng. An sẽ tô màu mỗi tầng của tháp bằng một trong 3 màu: đỏ, xanh dương, xanh lá hoặc không tô. Đồng thời, An cũng định nghĩa độ đẹp của tháp như sau:

Độ đẹp của tháp bằng tổng độ đẹp của **N** tầng. Trong đó, độ đẹp của tầng màu đỏ là **A**, xanh lá là **A + B**, xanh dương là **B**, và **0** nếu không được tô. Ở đây, **A, B** là các số được cho trước.

An tự hỏi rằng, có bao nhiêu cách tô màu sao cho độ đẹp của tháp đúng bằng **K**? Hai cách tô được xem là khác nhau nếu có một tầng được tô màu khác nhau trong hai cách.

**Dữ liệu**

Gồm một dòng chứa bốn số nguyên **N, A, B, K** ( 1 ≤ N ≤ 3.105, 1 ≤ A,B ≤ 3 .105, 0 ≤ K ≤ 18.1010)

**Kết quả**

In ra số cách tô mod **998244353**.

**Ví dụ**

|  |  |
| --- | --- |
| TOMAU.INP | TOMAU.OUT |
| 4 1 2 5 | 40 |
| 2 5 6 0 | 1 |
| 90081 33447 90629 6391049189 | 577742975 |

**Hướng dẫn thuật toán**

Xanh lá = Đỏ + Xanh dương

Nhận xét 1: Mỗi tầng có thể tô đỏ, xanh, đỏ + xanh, không tô

Nhận xét 2: Tô đỏ riêng, rồi tô xanh riêng

**Duyệt trâu:**

Gọi **x** là số tầng đỏ, **y** là số tầng xanh dương

|  |
| --- |
| for x = 0 to N  for y = 0 to N  if (A\*x + B\*y == K):res += Combi(n, x)\*Combi(n, y) |

**Tối ưu**

Gọi x là số tầng đỏ, y là số tầng xanh dương

|  |
| --- |
| for x = 0 to N  y = (K - A \* x) / B  res += combi(n, x) \* combi(n, y) |

## **Bài 3: HINHTHANG - Hình thang cân (Đề thi thử DHBB 2021)**

**Đề bài**

Cho **n** độ dài đoạn thẳng. Hãy đếm số cách chọn ra **4** trong số **n** độ dài trên để dựng ra một hình thang cân có diện tích khác 0.

Hai cách được cho là khác nhau, nếu có 1 cạnh trong cách này không là cạnh trong cách kia.

**Dữ liệu**

* + Dòng đầu tiên chứa số nguyên **t** là số lượng test. Sau đó là **t** test.
  + Mỗi test bắt đầu bằng một dòng chứa số nguyên **n** (1 ≤ *n* ≤ 5000).
  + Dòng thứ hai chứa **n** số nguyên dương không vượt quá 100000000 là độ dài các đoạn thẳng.
  + Tổng **n** trong tất cả các test không vượt quá 5000.

**Kết quả**

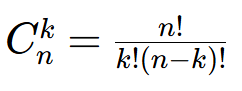
* + Với mỗi test, ghi ra đáp số trên một dòng

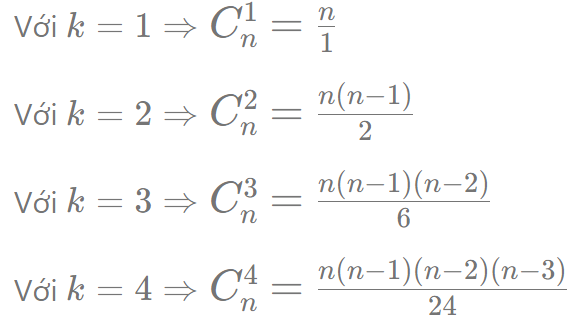
***Ví dụ***

|  |  |
| --- | --- |
| **HINHTHANG.INP** | **HINHTHANG.OUT** |
| 2  4  3 5 5 9  6  1 1 1 1 1 1 | 1  15 |

**Hướng dẫn thuật toán**

Số cách để chọn *k* cạnh bất kì trong *n* cạnh bằng nhau là:



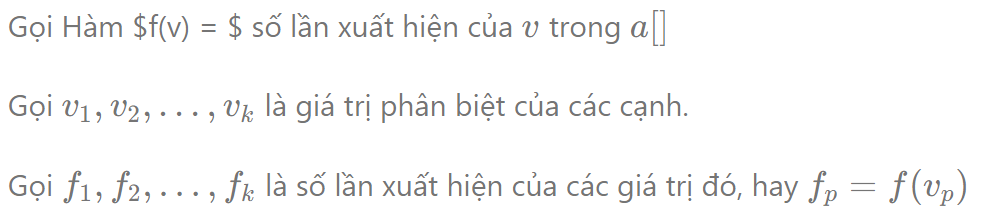


* **Nhận xét:** Thay vì thử các bộ 4 cạnh, ta chỉ cần thử các giá trị phân biệt

Các giá trị giống nhau ta dùng tổ hợp để đếm số cách chọn thì đơn giản hơn

Và số cách chọn từ một bộ bao gồm các giá trị phân biệt đó là tích các cách chọn mỗi giá trị đó với nhau.

* Để thuận tiện, ta định nghĩa việc nén mảng như sau

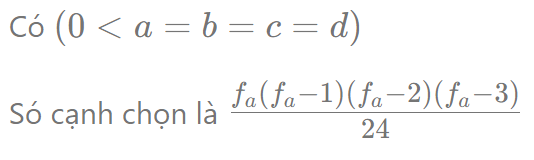


* **Nhận xét:** Vì hình thang cân, nên hiển nhiên có ít nhất một cặp cạnh bằng nhau, nên ta không cần duyệt giá trị thứ 4

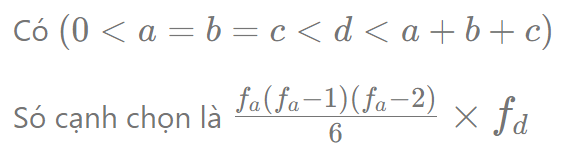
Với mỗi bộ ta thử xét xem giá trị thứ 4 là giá trị nào trong 3 số đã chọn

Để kiểm tra thay vì duyệt hoán vị, thì vì giờ đã nén mảng, và đếm bằng tổ hợp thì tiện hơn, nên ta sẽ xét các trương hợp riêng biệt kể trên

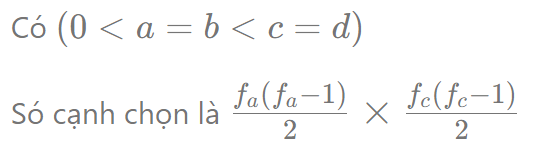
* **Hình vuông:**



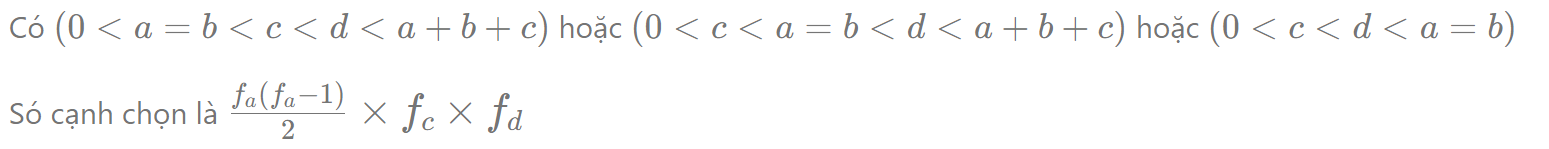
* **Hình thang cân phải:**



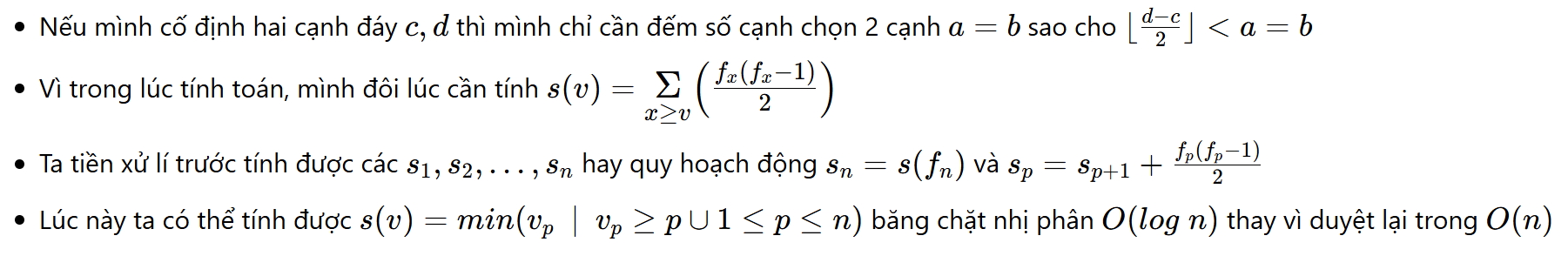
* **Hình chữ nhật:**

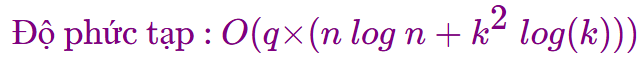


* **Hình thang cân thường:**



**Tối ưu bằng tiền xử lí quy hoạch động**





## **Bài 4: Hành trình**

**Đề bài**

Vào thời điểm t = 0, Bessie ở vị trí x=0 trên một đường số vô hạn. Cô ấy hoảng loạn tìm kiếm lối thoát bằng cách di chuyển qua trái hoặc phải mỗi giây một đơn vị. Tuy nhiên, thực tế không có lối thoát nào và sau T giây, Bessie lại ở vị trí x=0, mệt mỏi và từ bỏ.

Farmer cố gắng theo dõi Bessie nhưng chỉ biết số lần Bessie băng qua các điểm 0.5, 1.5, 2.5, …, (N−1).5, được cho bởi một mảng A0​, A1​, …, AN−1​ (1 ≤ N ≤ 105,1 ≤ Ai ​≤ 106). Bessie không bao giờ đi ra ngoài phạm vi x > N hoặc x < 0.

Cụ thể, hành trình của Bessie có thể được biểu diễn bởi một chuỗi các ký tự L (trái) và R (phải), trong đó ký tự thứ i biểu thị hướng Bessie di chuyển trong đoạn tiếp theo. Số lần đổi hướng được định nghĩa là số lần xuất hiện của các cặp LR cộng với số lần xuất hiện của các cặp RL.

Hãy giúp Farmer đếm số hành trình mà Bessie có thể đã đi, phù hợp với A, đồng thời tối thiểu hóa số lần đổi hướng. Đảm bảo rằng luôn có ít nhất một hành trình hợp lệ.

**Dữ liệu**

* Dòng đầu tiên chứa N.
* Dòng thứ hai chứa A0​, A1​, …, AN−1​.

**Kết quả**

* Số lượng hành trình Bessie có thể thực hiện, modulo **109+7**.

Ví dụ

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| 2  4 6 | 2 |

Bessie phải đổi hướng ít nhất 5 lần. Có 2 hành trình tương ứng với việc Bessie đổi hướng đúng 5 lần:

* RRLRLLRRLL
* RRLLRRLRLL

**Ràng buộc:**

* Test 2 − 4: N ≤ 2 và max(Ai​) ≤ 103.
* Test 5 − 7: N ≤ 2.
* Test 8−11: max(Ai​) ≤ 103.
* Test 12 − 21: Không có ràng buộc bổ sung.

**Hướng dẫn thuật toán**

Hãy gán chú thích cho từng ký tự trong hành trình với một chỉ số phụ **i**, biểu thị điểm **i.5** mà nó đi qua. Cụ thể:

* Nếu Bi ​≥ Bi+1​, thì bất kỳ Li​ nào phải được theo sau bởi một Li+1​. Ngoài ra, chính xác Bi+1​ ký tự Ri​ phải được theo sau bởi Ri+1​, và các Bi​ − Bi+1​ ký tự Li+1​ còn lại sẽ được theo sau bởi Li​.
* Nếu Bi ​≤ Bi+1​, thì bất kỳ Ri​ nào phải được theo sau bởi một Ri+1​. Ngoài ra, chính xác Bi​ ký tự Li+1​ phải được theo sau bởi Li​, và các Bi+1​ − Bi​ ký tự Ri+1​ còn lại sẽ được theo sau bởi Ri​.

Ngoài ra, chúng ta lưu ý rằng trong bất kỳ hành trình nào, ký tự cuối cùng Li+1​ phải được theo sau bởi một Li​ (và không phải là Ri+1​), nếu không, Bessie sẽ không thể quay lại vị trí 0.

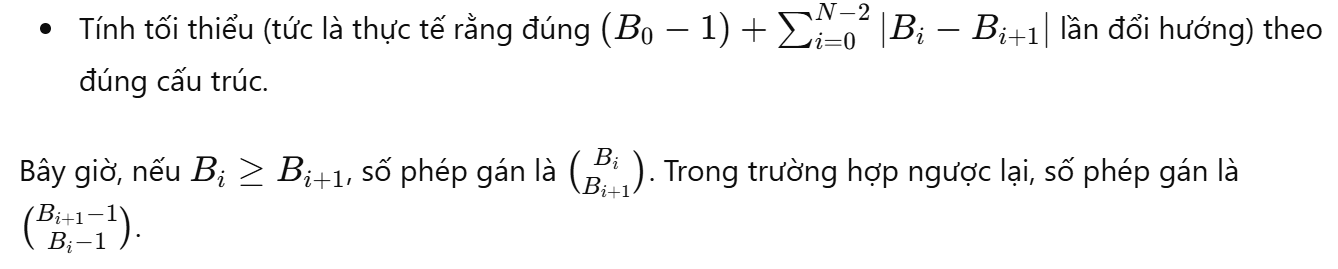
**Phân tích**

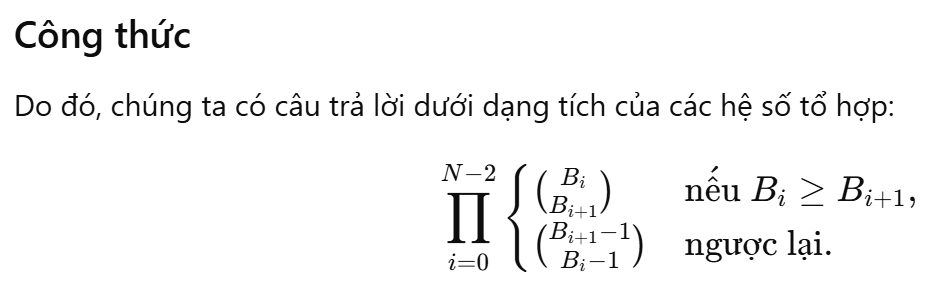
Chúng ta khẳng định rằng đây là ràng buộc duy nhất. Nghĩa là, để đếm số đường đi khả thi, với mỗi i = 0, 1, …, N−2, chỉ cần đếm số cách chọn các Ri​ được theo sau bởi Ri+1​ nếu Bi ​≥ Bi+1​, hoặc các Li​ được theo sau bởi Li+1​ nếu Bi ≤ Bi+1​ (đảm bảo rằng Li+1​ cuối cùng được theo sau bởi Li​). Sau đó, bất kỳ phép gán nào cũng sẽ tạo ra một đường đi hợp lệ.

Tính duy nhất là rõ ràng, nhưng để chứng minh tính hợp lệ, giả sử chúng ta xây dựng đường đi theo các phép gán, trong đó đường đi kết thúc khi L0​ cuối cùng được theo sau bởi tất cả R0​. Chúng ta cần kiểm tra rằng tất cả các Ri​ và Li​ thực sự được sử dụng; vì số lượng Ri​ và Li​ đã xác định.

Chúng ta sẽ làm điều này bằng phương pháp quy nạp:

* L0​ là đúng theo giả thiết. Với bước quy nạp, giả sử Bi​ xuất hiện trong hành trình. Nếu Bi ​≥ Bi+1​, thì Bi+1​ xuất hiện ngay sau đó. Ngược lại, nếu Bi ​≤ Bi+1​, thì tất cả Li+1​ phải xuất hiện.





**Tính toán**

Cho T=maxi​Ai​. Chúng ta có thể tiền tính giai thừa trong O(T). Tính nghịch đảo giai thừa bằng cách tính nghịch đảo modulo của T! (ví dụ: nâng lên MOD−2 bằng lũy thừa nhị phân). Sau đó, chúng ta có thể tính các hệ số tổ hợp trong O(1) thời gian, với tổng thời gian là O(log(MOD)+T+N).

# **6. KẾT LUẬN**

Chuyên đề này được xây dựng nhằm cung cấp một cái nhìn toàn diện và có hệ thống về toán tổ hợp, từ lý thuyết nền tảng đến các bài tập vận dụng và nâng cao. Qua từng phần nội dung, học sinh không chỉ được trang bị kiến thức cơ bản mà còn có cơ hội rèn luyện kỹ năng tư duy logic, phân tích và giải quyết vấn đề. Việc lồng ghép các dạng bài thi lập trình vào chuyên đề giúp tăng tính thực tiễn và ứng dụng của kiến thức toán học trong lĩnh vực công nghệ thông tin. Hy vọng rằng chuyên đề sẽ là tài liệu hữu ích, hỗ trợ hiệu quả cho quá trình học tập, luyện thi và phát triển tư duy thuật toán của học sinh.

# **7. TÀI LIỆU THAM KHẢO**

[1]. Bài tập của Thầy Đỗ Đức Đông.

[3]. Tài liệu tham khảo của các đồng nghiệp khác.

[4]. Các trang web :

[https://wiki.vnoi.info](https://wiki.vnoi.info/translate/he/Number-Theory-5)

<https://codeforces.com>

<https://lqdoj.edu.vn>

[https://usaco.guide](https://usaco.guide/gold/combo?lang=cpp)