ankhangluonvuituoi@gmail.com | 0967 670 770 | https://github.com/GrootTheDeveloper

TOÁN HỌC

Tài liệu ôn tập Competitive Programming

Đặng Phúc An Khang

Sinh viên ngành CNTT (AI & DS) — Trường Đại học Quản lý & Công nghệ TP.HCM (UMT)

Ngày 30 tháng 8 năm 2025

Mục lục

CHƯƠNG 1

GIỚI THIỆU

Contents

Bài viết này được biên soạn với mục tiêu giúp tác giả hệ thống hoá và vận dụng các kiến thức thuộc chuyên đề *Quy hoạch động (Dynamic Programming)*, từ đó áp dụng hiệu quả trong *Competitive Programming* (Lập trình thi đấu).

1.1 Các nguồn tài nguyên

- C/C++: `https://github.com/GrootTheDeveloper/OLP-ICPC/tree/master/2025/C%2B%2B`
- [Kho23]. *CP10. Competitive Programming* `https://drive.google.com/drive/folders/1MTEVHT-7nBnMJ7C9LgyAR_pEVSE3FlKz?fbclid=IwAR3TovIj2rKCRe1a4oZxW-LQCoEoVkipVAvCzwrr0nJ6GzcAd47P6LO1Rwc`
- [CP-]. *Algorithms for Competitive Programming* `https://cp-algorithms.com`
- [VNO]. *Thư viện VNOI* `https://wiki.vnoi.info`

1.2 Tài khoản trên các Online Judge

- Codeforces: `https://codeforces.com/profile/vuivethoima`
- VNOI: `oj.vnoi.info/user/Groot`
- IUHCoder: `oj.iuhcoder.com/user/ankhang2111`
- MarisaOJ: `https://marisaoj.com/user/grootsiuvip/submissions`
- CSES: `https://cses.fi/user/212174`
- UMTOJ: `sot.umtoj.edu.vn/user/grootsiuvip`
- SPOJ: `www.spoj.com/users/grootsiuvip/`
- POJ: `http://poj.org/userstatus?user_id=vuivethoima`
- AtCoder: `https://atcoder.jp/users/grootsiuvip`
- OnlineJudge.org: `vuivethoima`

1.3 Một vài lưu ý

Chuyên đề này được viết bởi hai "tác giả":

- **vuivethoima** – tác giả chính, chịu trách nhiệm biên soạn nội dung.
- **Groot** – một thằng chuyên chọc ngoáy, đặt những câu hỏi nghe thì rất ngu ngơ nhưng lại gợi mở những góc khuất của bài toán mà thường ít ai để ý (chắc vậy?).

Nói cho sang thì là "cộng tác", nhưng thực chất đây là quá trình DPAK tự viết, rồi tự hỏi, rồi tự tranh luận. Hai "nhân vật" trong đầu thay phiên nhau đóng vai *tác giả* và *độc giả khó tính*. Và thế là hình thành nên chuyên đề này.

CHƯƠNG 2

TỔ HỢP (COMBINATORICS)

Contents

2.1 Lý thuyết

Tổ hợp là một ngành nghiên cứu toán học chuyên nghiên cứu về các trạng thái, các cấu hình của một sự vật, sự việc hoặc phương pháp nào đó.

2.1.1 Công cụ toán học cơ bản

Định nghĩa 1 (Giai thừa, chỉnh hợp, tổ hợp)**.** *Với* $n \in \mathbb{N}$*,* ***giai thừa*** $n! = 1 \cdot 2 \cdot \dots \cdot n$ *(quy ước* $0! = 1$*).*
Chỉnh hợp $P(n,k) = \dfrac{n!}{(n-k)!}$*: số cách chọn* có thứ tự *k phần tử từ n phần tử phân biệt.*
Tổ hợp $C(n,k) = \binom{n}{k} = \dfrac{n!}{k!(n-k)!}$*: số cách chọn* không thứ tự *k phần tử từ n phần tử phân biệt ($0 \le k \le n$).*

Tính chất 1 (Một số tính chất cơ bản)**.**

$$\binom{n}{0} = \binom{n}{n} = 1, \qquad \binom{n}{k} = \binom{n}{n-k}, \qquad \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}.$$

Định lý 1 (Nhị thức Newton)**.** *Với* $n \in \mathbb{N}$*,*

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}.$$

Định nghĩa 2 (Nguyên lý bao hàm–loại trừ (PIE))**.** *Với các tập $A_1, \dots, A_m$ trong không gian hữu hạn U,*

$$\left| U \setminus \bigcup_{i=1}^{m} A_i \right| = \sum_{J \subseteq [m]} (-1)^{|J|} \left| \bigcap_{j \in J} A_j \right|.$$

2.1.2 Kỹ thuật tính toán trong lập trình thi đấu

Số học modulo (MOD nguyên tố) Trong đa số bài, MOD là số nguyên tố (ví dụ 10^9+7 hoặc $998\,244\,353$). Với p nguyên tố và $a \not\equiv 0 \pmod p$:

$$a^{-1} \bmod p \equiv a^{p-2} \text{ (Fermat nhỏ)}.$$

Chuẩn hoá số âm: `((x % MOD) + MOD) % MOD`.

Listing 2.1: Lũy thừa nhanh và nghịch đảo modulo (C++)

```
const long long MOD = 1000000007LL;

long long modpow(long long a, long long e){
    long long r = 1 % MOD;
    a %= MOD;
```

```
    while(e){
        if(e & 1) r = (r * a) % MOD;
        a = (a * a) % MOD;
        e >>= 1;
    }
    return r;
}

long long modinv(long long a){
    a %= MOD; if(a < 0) a += MOD;
    return modpow(a, MOD - 2);
}
```

Tiền xử lý giai thừa & nghịch đảo giai thừa Sau $O(N)$ chuẩn bị, tính $\binom{n}{k}$ trong $O(1)$.

Listing 2.2: Precompute factorial / invfactorial; tính C(n,k)

```
const int MOD = 1000000007;
const int MAXN = 2000000;

int fact[MAXN+1], invfact[MAXN+1];

int modpow_int(int a, long long e){
    long long r = 1, b = (a % MOD + MOD) % MOD;
    while(e){
        if(e & 1) r = (r * b) % MOD;
        b = (b * b) % MOD;
        e >>= 1;
    }
    return (int)r;
}
void init_fact(){
    fact[0] = 1;
    for(int i=1;i<=MAXN;i++) fact[i] = (long long)fact[i-1]*i % MOD;
    invfact[MAXN] = modpow_int(fact[MAXN], MOD-2);
    for(int i=MAXN;i>=1;i--) invfact[i-1] = (long long)invfact[i]*i % MOD;
}
int C(int n, int k){
    if(k < 0 || k > n) return 0;
    return (long long)fact[n]*invfact[k]%MOD*invfact[n-k]%MOD;
}
```

2.2 Bài tập

Bài tập 1. Ball in Berland **link:** https://codeforces.com/problemset/problem/1475/C

Ở trường của Vasya đang chuẩn bị cho lễ tốt nghiệp. Một trong những tiết mục là buổi dạ hội, nơi các cặp nam–nữ sẽ khiêu vũ. Mỗi lớp phải cử *hai* cặp tham dự. Ở lớp của Vasya có a bạn nam và b bạn nữ muốn tham gia, nhưng không phải tất cả đều sẵn sàng nhảy cặp.

Cụ thể, bạn biết k cặp nam–nữ có thể ghép được. Hãy chọn **hai** cặp trong số đó sao cho **không người nào xuất hiện trong quá một cặp**.

Ví dụ, nếu $a = 3$, $b = 4$, $k = 4$ và các cặp $(1,2),(1,3),(2,2),(3,4)$ sẵn sàng nhảy (trong mỗi cặp, số của bạn nam đứng trước, rồi đến số của bạn nữ) thì các cách chọn hợp lệ gồm, chẳng hạn: $(1,3)$ và $(2,2)$; $(3,4)$ và $(1,3)$. Những cách không hợp lệ: $(1,3)$ và $(1,2)$ — nam số 1 xuất hiện hai lần; $(1,2)$ và $(2,2)$ — nữ số 2 xuất hiện hai lần.

Hãy đếm số cách chọn hai cặp thỏa điều kiện trên. Hai cách được coi là khác nhau nếu chúng gồm các cặp khác nhau.

Input

- Dòng đầu chứa một số nguyên t $(1 \le t \le 10^4)$ — số bộ test.
- Với mỗi bộ test: dòng đầu chứa ba số nguyên a, b, k $(1 \le a, b, k \le 2 \cdot 10^5)$ — số bạn nam, số bạn nữ và số cặp có thể ghép.
- Dòng thứ hai chứa k số $a_1, a_2, \ldots, a_k$ $(1 \le a_i \le a)$, trong đó a_i là số của bạn nam ở cặp thứ i.
- Dòng thứ ba chứa k số $b_1, b_2, \ldots, b_k$ $(1 \le b_i \le b)$, trong đó b_i là số của bạn nữ ở cặp thứ i.
- Bảo đảm tổng các giá trị a, b và k qua tất cả các test không vượt quá $2 \cdot 10^5$.
- Bảo đảm mỗi cặp (a_i, b_i) xuất hiện nhiều nhất một lần trong một test.

Output

Với mỗi bộ test, in ra một số nguyên — số cách chọn hai cặp thỏa điều kiện.

Ví dụ

Sample Input	Sample Output
3	4
3 4 4	0
1 1 2 3	2
2 3 2 4	
1 1 1	
1	
1	
2 2 4	
1 1 2 2	
1 2 1 2	

Phân tích bài toán

Gọi `boy[i]` và `girl[i]` lần lượt là chỉ số của bạn nam và bạn nữ trong cặp thứ i.

Bài toán yêu cầu đếm số cặp (i, j) với $i \neq j$ sao cho `boy[i]` $\neq$ `boy[j]` và `girl[i]` $\neq$ `girl[j]`. Nếu làm trực tiếp bằng cách duyệt tất cả các cặp (i, j) thì độ phức tạp $O(k^2)$, quá lớn khi $k \leq 2 \cdot 10^5$.

Ý tưởng tối ưu như sau:

- Khi đang xét cặp thứ i, rõ ràng có $i - 1$ cặp trước đó có thể kết hợp với nó.
- Ta phải loại bỏ những cặp trùng bạn nam hoặc bạn nữ:
  - Có `mb[boy[i]]` cặp trước đó có cùng bạn nam.
  - Có `mg[girl[i]]` cặp trước đó có cùng bạn nữ.
- Như vậy sau khi loại bỏ, ta còn M cặp thỏa mãn. Với mỗi cặp trong M cặp, ta đều ghép được với cặp thứ i. Vậy số cặp hợp lệ có thể ghép với i là: $(i - 1) -$ `mb[boy[i]]` $-$ `mg[girl[i]]`

Độ phức tạp mỗi test là $O(k \log k)$, phù hợp với ràng buộc đề bài.

Cài đặt

```cpp
#include <bits/stdc++.h>
#define int long long
using namespace std;

signed main() {
    int t; cin >> t;
    while (t--) {
        int a, b, k; cin >> a >> b >> k;
        vector<int> boy(k + 1), girl(k + 1);
        for (int i = 1; i <= k; i++) {
            cin >> boy[i];
        }
        for (int i = 1; i <= k; i++) {
            cin >> girl[i];
        }
        map<int,int> mb, mg;
        mb[boy[1]]++, mg[girl[1]]++;
        int ans = 0;
        for (int i = 2; i <= k; i++) {
            int group = i - 1;
            group -= mb[boy[i]];
            group -= mg[girl[i]];
            ans += group;
            mb[boy[i]]++;
            mg[girl[i]]++;
        }
        cout << ans << endl;
    }
}
```

CHƯƠNG 3

QUY HOẠCH ĐỘNG + TỔ HỢP

CHƯƠNG 4

XÁC SUẤT (PROBABILITIES)

CHƯƠNG 5

KỲ VỌNG (EXPECTED VALUE)

CHƯƠNG 6

LÝ THUYẾT TRÒ CHƠI (GAME THEORY)

CHƯƠNG 7

MỘT SỐ ĐỊNH LÝ VÀ PHƯƠNG PHÁP CHỨNG MINH TOÁN HỌC

CHƯƠNG 8

INCLUSIVE - EXCLUSIVE

CHƯƠNG 9

GAME THEORY SOLVING BY GRUNDY NUMBER

CHƯƠNG 10

EULER TOTIENT FUNCTION

CHƯƠNG 11

MOBIUS FUNCTION

CHƯƠNG 12

GEOMETRY

CHƯƠNG 13

CONVEX HULL + OPTIMIZE DP BY CONVEX HULL

CHƯƠNG 14

FFT

BIBLIOGRAPHY

[CP-] CP-Algorithms. *CP-Algorithms*. URL: `https://cp-algorithms.com/` (visited on 08/26/2025).

[Kho23] Dinh Nguyen Khoi. *Competitive Programming 10*. drive, 2023.

[VNO] VNOI. *VNOI Wiki*. URL: `https://wiki.vnoi.info/` (visited on 08/26/2025).