universal 3 stage 26 : China

Created: 4/29/2025 18:01
Updated: 4/29/2025 18:12
Exported: 4/29/2025 18:13

Dưới đây là bản dịch tiếng Việt và tóm tắt của đoạn mã trong file `**dapmocuoctinh.cpp**`:

📌 Bản dịch tiếng Việt:

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

#define ll long long
#define fi first
#define se second
#define re exit(0);
#define left id<<1
#define right id<<1|1
#define pb push_back

const int mod = 1e9+7;
const int INF = 1e9;

typedef vector<int> vi;
typedef pair<int,int> pii;
typedef vector<pii> vii;

void rf() {
    freopen("bai1.inp", "r", stdin); // Mở file đầu vào tên là "
```

🅰️ Bài A - Hitoshizuku

📘 Bản dịch tiếng Việt

Ngay cả một đám mây trôi cũng cứ tiếp tục đi
Dù chẳng biết điều gì sẽ đến chỉ sau một giây
Tiến vào ngày mai mờ ảo và phai nhòa vì
nỗi lo âu của chính bản thân.

Trong một trường học có tổng cộng $3n$ bạn gái muốn lập thành n ban nhạc, mỗi ban gồm đúng **3 người**.

Mỗi bạn gái thứ i có một chỉ số "sức hút" là a_i, chỉ số này tổng hợp nhiều yếu tố như kỹ năng chơi đàn, mức độ căng thẳng hay khả năng ngâm thơ,...

Để tránh ghen tị và căng thẳng, mỗi bạn gái khai báo một giá trị b_i — đây là **giới hạn sức hút tối đa của bạn khác mà cô ấy chấp nhận đứng chung ban**. Cụ thể, bạn gái thứ i không muốn vào cùng ban với ai có $a_j > b_i$.
(**Lưu ý:** đảm bảo luôn có $b_i \geq a_i$, vì không ai ghen tị với chính mình.)

Nhiệm vụ là kiểm tra xem có thể chia $3n$ bạn gái thành n ban nhạc thỏa điều kiện của từng người không. Chỉ quan tâm đến chỉ số sức hút — các kỹ năng khác có thể luyện tập.

Dữ liệu vào

- Dòng đầu tiên: số bộ test T $(1 \leq T \leq 10^5)$
- Mỗi test gồm:
  - Một dòng: số ban nhạc n $(1 \leq n \leq 10^5)$
  - Tiếp theo là $3n$ dòng, mỗi dòng gồm hai số a_i, b_i $(1 \leq a_i \leq b_i \leq 10^9)$

Tổng số n trong tất cả test không vượt quá 10^5

Dữ liệu ra

- Với mỗi test:
  - Nếu có cách chia hợp lệ: in ra `Yes`, sau đó n dòng, mỗi dòng gồm 3 chỉ số là các bạn gái thuộc cùng một ban (chỉ số từ 1 đến $3n$, không trùng nhau)
  - Nếu không chia được: in ra `No`

✍️ Tóm tắt đề

Cho $3n$ bạn gái, mỗi bạn có sức hút a_i và giới hạn chịu đựng sức hút bạn khác b_i. Chia họ thành n nhóm 3 người sao cho mỗi người đều không phải đứng cùng người có sức hút vượt quá mức họ chấp nhận.
Hỏi có chia được như vậy không? Nếu có, in ra nhóm cụ thể. Nếu không, in `No`.

🅱️ Bài B - Đoán hình đa giác 2

(Bài toán tương tác)

📘 Bản dịch tiếng Việt

Bạn được cung cấp một **đa giác đơn**, nhưng trước khi biết danh sách các đỉnh, các đỉnh này đã bị **xáo trộn thứ tự**.

Bạn có thể **đặt tối đa** $n - 2$ **truy vấn**, mỗi truy vấn gồm 3 số nguyên a, b và c (với $a^2 + b^2 > 0$). Khi đó bạn nhận được độ dài tổng cộng các đoạn mà **đường thẳng** $ax + by + c = 0$ cắt bên trong hoặc đi qua **đa giác**.

Kết quả truy vấn được trả về dưới dạng:

$$\frac{r\sqrt{a^2+b^2}}{s}$$

với $r \geq 0$, $s \geq 1$, và $\gcd(r, s) = 1$ (gcd là ước chung lớn nhất).

Mục tiêu là bạn phải **xác định đa giác ban đầu**, và in ra các đỉnh **theo thứ tự ngược chiều kim đồng hồ**. Bạn có thể bắt đầu từ bất kỳ đỉnh nào.

> *"Đa giác ban đầu và thứ tự xáo trộn được cố định từ đầu và **không thay đổi theo truy vấn của bạn** (không tương tác động)."*

Dữ liệu vào

- Số nguyên T — số test $(1 \leq T \leq 200)$
- Với mỗi test:
  - Một số nguyên n — số đỉnh của đa giác $(3 \leq n \leq 200)$
  - n dòng tiếp theo, mỗi dòng chứa tọa độ x, y của một đỉnh (bị xáo trộn). $0 \leq x, y \leq 1000$

Bảo đảm:

- Đa giác là **đơn** (không tự cắt, các đỉnh khác nhau, cạnh không trùng hoặc chạm nhau trừ tại đỉnh).
- Không có 2 cạnh liên tiếp nào **thẳng hàng**.
- Tổng tất cả n trên các test không vượt quá 200.

Giao tiếp tương tác

🔷 **Truy vấn**:
In ra:

```
? a b c
```

=> Nhận về hai số nguyên r, s tương ứng với độ dài:

$$\frac{r\sqrt{a^2+b^2}}{s}$$

🔷 **Đoán đa giác**:
In ra $n+1$ dòng:

```
!
x₁ y₁
x₂ y₂
...
xₙ yₙ
```

(với các đỉnh theo thứ tự ngược chiều kim đồng hồ)

Lưu ý:

- Lệnh đoán không tính là truy vấn.
- Nhớ `flush` output sau mỗi lệnh in:
  - C++: `cout.flush();`
  - Python: `sys.stdout.flush()`

✍️ Tóm tắt đề

Bạn có một đa giác đơn với các đỉnh bị xáo trộn.
Bạn được phép hỏi tối đa $n-2$ lần để kiểm tra xem một **đường thẳng bất kỳ** cắt

bao nhiêu **đoạn nằm trong đa giác**.

Từ đó, hãy **xác định lại thứ tự các đỉnh theo ngược chiều kim đồng hồ**, và in ra. Đây là **bài toán tương tác**, yêu cầu bạn vừa hỏi, vừa đoán.

🅒 Bài C - Norte da Universidade

📘 Bản dịch tiếng Việt

Một số trường đại học sử dụng **hệ thống tọa độ** để đặt tên cho các tòa nhà trong khuôn viên: chia khuôn viên thành 4 vùng **Bắc (N)**, **Nam (S)**, **Tây (W)**, **Đông (E)**, sau đó đánh số thứ tự cho các tòa nhà trong từng vùng. Tuy nhiên vì lý do lịch sử hay địa lý, một số vùng có thể không kết nối hoặc thậm chí không tồn tại.

Dù vậy, hệ thống đặt tên vẫn phải tuân theo các **quy tắc sau** để đảm bảo tính nhất quán:

- Với mỗi tòa nhà ở vùng **Bắc (N)**, **mọi ô phía trên (phía Bắc)** của nó cũng phải thuộc vùng Bắc.
- Với mỗi tòa nhà ở vùng **Nam (S)**, **mọi ô phía dưới (phía Nam)** của nó cũng phải thuộc vùng Nam.
- Với mỗi tòa nhà ở vùng **Đông (E)**, **mọi ô bên phải (phía Đông)** của nó cũng phải thuộc vùng Đông.
- Với mỗi tòa nhà ở vùng **Tây (W)**, **mọi ô bên trái (phía Tây)** của nó cũng phải thuộc vùng Tây.

Hôm nay, Aki đến thăm một trường đại học dùng hệ thống đặt tên như trên. Sau buổi chiều lang thang, Aki nhận thấy các tòa nhà được bố trí trong một **lưới** $n \times m$ (n hàng từ Bắc xuống Nam, m cột từ Tây sang Đông), mỗi ô là một tòa nhà.

Aki chỉ biết được một số ô (đã có ký hiệu N, S, W, E), những ô còn lại thì chưa rõ (ký hiệu `?`). Aki muốn biết **có bao nhiêu cách hợp lệ để điền các ô còn lại** sao cho tất cả các quy tắc được thỏa mãn.

Hai cách bố trí khác nhau nếu có ít nhất một ô có **ký hiệu vùng khác nhau**.

Dữ liệu vào

- Dòng đầu tiên: số bộ test T $(1 \leq T \leq 100)$
- Mỗi test gồm:
  - Một dòng chứa hai số nguyên n, m $(1 \leq n, m \leq 1000)$ — kích thước bản đồ (hàng và cột)
  - Sau đó là n dòng, mỗi dòng chứa chuỗi m ký tự gồm:
    - `'N'`: Bắc
    - `'S'`: Nam
    - `'E'`: Đông
    - `'W'`: Tây
    - `'?'`: chưa biết

Tổng số ô trên tất cả test không vượt quá 2×10^6

Dữ liệu ra

Với mỗi test, in ra **số cách điền hợp lệ modulo** 998244353.

📌 Tóm tắt đề

- Mỗi ô trong lưới $n \times m$ là một tòa nhà thuộc vùng N, S, W, E hoặc chưa biết (`?`).
- Các quy tắc yêu cầu **vùng ảnh hưởng phải nối dài về hướng cố định** (Bắc thì kéo lên trên, Nam kéo xuống dưới, Tây kéo sang trái, Đông kéo sang phải).
- Tính **số cách điền các ô `?`** sao cho thỏa các quy tắc trên.
- Kết quả lấy **modulo** 998244353.

Bạn cần xử lý nhiều test, và mỗi test là một bản đồ lưới ô vuông. Với mỗi lưới, xác định bao nhiêu cách điền các ô chưa biết sao cho quy tắc vùng vẫn được giữ nguyên.

🅳 Bài D - Keystone Correction

📘 Bản dịch tiếng Việt

Là một fan cuồng anime, Haru mới mua một chiếc máy chiếu để tận hưởng rạp chiếu phim tại gia. Tuy nhiên, chiếc máy chiếu này lại thuộc loại "giá rẻ", hoạt động theo kiểu "miễn là nó bật được". Nó chỉ hỗ trợ điều chỉnh rất hạn chế, lại được bắt cố định nên **hình ảnh chiếu thường bị lệch**.
May mắn thay, có chức năng **keystone correction** (chỉnh thẳng hình) thủ công, giúp căn chỉnh lại 4 góc sao cho hình ảnh chiếu ra giữ được **tỷ lệ khung hình gốc** và đúng vị trí.

Máy chiếu được mô hình hóa như **nguồn sáng tại điểm (0, 0, 0)**, và chiếu ánh sáng qua một hình chữ nhật (gọi là **ống kính**) có kích thước $w \times h$. Tia sáng đi qua ống kính này và được chiếu lên một **màn hình (screen)** — là một hình chữ nhật cố định nằm trên mặt phẳng tường.

🎯 Nhiệm vụ của bạn

Tìm cách chọn 4 điểm neo (anchor points) trên hình chữ nhật của ống kính sao cho:

- Hình chiếu lên màn hình là **hình chữ nhật** có cùng **tỷ lệ khung hình với ống kính**.
- Hình này phải **nằm hoàn toàn trong màn hình**.
- Cạnh dưới của hình chiếu phải **song song với mặt phẳng xOy** và là cạnh có **hoành độ z lớn nhất nhỏ nhất** (tức là cạnh gần người nhìn nhất).

Hãy viết chương trình để **tìm diện tích lớn nhất có thể đạt được** cho hình chữ nhật chiếu hợp lệ nói trên.

Dữ liệu vào

- Dòng đầu tiên: số bộ test T $(1 \leq T \leq 1000)$

Với mỗi test gồm **8 dòng**:

1. **4 dòng đầu**: mỗi dòng là một điểm (x_i, y_i, z_i) — các đỉnh của **hình chữ nhật ống kính**. Các đỉnh này theo thứ tự: trái dưới, phải dưới, phải trên, trái trên.
   - $|x_i|, |z_i| \leq 100$, $1 \leq y_i \leq 100$
   - $\max(x_1, x_4) < \min(x_2, x_3)$, $\max(z_1, z_2) < \min(z_3, z_4)$
2. **4 dòng tiếp theo**: mỗi dòng là một điểm (u_i, v_i, w_i) — các đỉnh của **màn hình**.
   - $|u_i|, |w_i| \leq 1000$, $\max(y_j) < v_i \leq 1000$
   - $u_1 = u_4 < u_2 = u_3$, $v_1 = v_4 < v_2 = v_3$, $w_1 = w_2 < w_3 = w_4$

✅ Ràng buộc thêm

- Hình chiếu có diện tích **không vượt quá** 10^6.
- Giao giữa chiếu thô ban đầu và màn hình là **ít nhất 1 đơn vị diện tích**.
- Góc nghiêng của cạnh dưới màn hình và cạnh dưới ống kính với trục x **không vượt quá** 10°.
- Máy chiếu và ống kính **nằm phía trước** mặt tường (theo trục y dương), **cách tường ít nhất 1 đơn vị**.
- Tỷ lệ w/h của ống kính: $1 \leq \frac{w}{h} \leq 4$

Dữ liệu ra

- Với mỗi test, in ra **diện tích lớn nhất** (dưới dạng số thực) có thể chiếu được, thỏa tất cả các điều kiện.
- Sai số cho phép: **tuyệt đối hoặc tương đối không vượt quá** 10^{-6}.

📌 Tóm tắt đề

Bạn được cung cấp 2 hình chữ nhật trong không gian 3D:

- Một là **hình chữ nhật ống kính** của máy chiếu (nằm gần điểm gốc).
- Một là **hình chữ nhật màn hình** (ở phía trước máy chiếu).

Yêu cầu: chọn cách chỉnh keystone (chiếu ánh sáng) sao cho:

- Hình ảnh chiếu lên màn hình là một **hình chữ nhật có cùng tỉ lệ khung hình với ống kính**, **nằm gọn trong màn hình**, và **nằm song song trục xOy ở cạnh**

dưới.

- Tìm **diện tích lớn nhất có thể đạt được** cho hình chiếu hợp lệ đó.

🅴 Bài E - Bảng điểm bị hỏng (Corrupted Scoreboard Log)

📘 Bản dịch tiếng Việt

Bạn đang tham gia công việc tình nguyện cho một cuộc thi lập trình cổ xưa. Thật không may, các giám khảo gặp rắc rối khi công bố bảng xếp hạng cuối cùng, bởi vì bảng điểm được khắc trên đá đã bị **mài mòn bởi bụi và cát**.
May thay, vẫn còn một bản sao chép (rubbing) lại từ phiến đá.

Thông tin cho biết có n đội và m bài trong cuộc thi. Mỗi dòng dữ liệu (ứng với một đội) **vẫn còn nguyên**, nhưng tất cả các **dấu cách đã bị mất**. Ngoài ra, **thứ tự các đội có thể đã bị xáo trộn**.

🎯 Yêu cầu

Khôi phục lại bảng điểm bằng cách **chèn dấu cách** vào mỗi dòng, sao cho:

- Mỗi dòng gồm:
  1. **Số bài giải được**
  2. **Tổng thời gian phạt**
  3. **m đoạn thông tin ứng với m bài toán**
- Mỗi bài có thể có:
  - **Không nộp bài gì**: để trống.

- **Đã giải bài**: biểu diễn dưới dạng `"y x try"` (nghĩa là giải đúng ở lần thử thứ x sau y phút). Nếu $x \geq 2$, thì `"try"` chuyển thành `"tries"`.
  - **Thời gian phạt**: $y + 20 \times (x - 1)$
- **Không giải đúng nhưng có nộp bài**: `"x try"` hoặc `"x tries"` tùy theo x

---

📌 Giả sử chắc chắn rằng:

- **Thời gian thi**: 300 phút (nộp bài từ phút 0 đến 299)
- **Không quá 13 bài**, không quá 500 đội
- **Mỗi đội không nộp quá 100 lần cho một bài**
- **Tổng chuỗi đầu vào hợp lệ** (luôn tồn tại cách tách hợp lệ)

📥 Dữ liệu vào

- Dòng đầu: hai số nguyên n, m — số đội và số bài ($1 \leq n \leq 500$, $1 \leq m \leq 13$)
- n dòng tiếp theo: mỗi dòng là chuỗi không dấu cách chứa dữ liệu của một đội

---

📤 Dữ liệu ra

- Với mỗi dòng đầu vào:
  - Xuất ra dòng tương ứng đã được chèn dấu cách đúng định dạng
  - Dòng gồm:
    `số_bài_đúng tổng_thời_gian_phạt [kết quả bài 1] [kết quả bài 2] ... [kết`

`quả bài m]`

- Các phần cách nhau đúng **một dấu cách**
- Không được có dấu cách dư
- Sau khi xóa hết dấu cách, kết quả phải khớp **chính xác với chuỗi đầu vào**

- Nếu có nhiều cách đúng, in **bất kỳ cách nào** là hợp lệ

🧠 Tóm tắt đề

Cho danh sách chuỗi bị mất dấu cách ghi lại bảng điểm cuối kỳ (mỗi dòng ứng với một đội).
Nhiệm vụ là khôi phục lại đúng cấu trúc dữ liệu bằng cách **thêm dấu cách phù hợp**, tuân theo luật tính điểm ACM (giống ICPC).
Chỉ cần đảm bảo chuỗi sau khi bỏ dấu cách **khớp với đầu vào** và cách tách là hợp lệ.

🅵 Bài F - Dựng lại hàm Boolean (Boolean Function Reconstruction)

📘 Bản dịch tiếng Việt

Bạn được cung cấp bảng chân trị (truth table) của một **hàm Boolean** có n biến đầu vào. Nhiệm vụ là xây dựng **biểu thức Boolean** đúng với bảng chân trị này, chỉ sử dụng **hai phép toán logic**:

- **Phép và**: `&`
- **Phép hoặc**: `|`

➤ Biểu thức Boolean hợp lệ có dạng:

- `T`, `F`: biểu diễn `True`, `False`

- `a`, `b`, ..., `z`: biểu diễn các biến boolean (biến thứ i là chữ cái thứ i trong bảng chữ cái)
- `(<expr>&<expr>)`, `(<expr>|<expr>)`: biểu diễn phép toán logic giữa 2 biểu thức con

🎯 Yêu cầu

Từ bảng chân trị, xây dựng biểu thức thỏa mãn:

- Kết quả đúng với mọi giá trị đầu vào trong bảng chân trị
- Sử dụng không quá $2^{n-1} + 10$ phép toán nhị phân
- Mức độ lồng ngoặc không quá 100 tầng

Luôn tồn tại cách dựng biểu thức thỏa mãn điều kiện nếu kết quả là "Yes".

📥 Dữ liệu vào

- Dòng đầu tiên: số lượng test T $(1 \leq T \leq 2^{16})$
- Với mỗi test:
  - Dòng 1: số nguyên n $(1 \leq n \leq 15)$ — số biến Boolean
  - Dòng 2: chuỗi nhị phân độ dài 2^n, biểu diễn bảng chân trị của hàm

Cách hiểu chuỗi nhị phân:

- Với mỗi tổ hợp giá trị của n biến $x_1, x_2, ..., x_n$, hãy đánh số thứ tự (giống nhị phân) và lấy bit tương ứng trong chuỗi để biết giá trị hàm.

- Ví dụ: nếu $x_1 = 1, x_2 = 0, x_3 = 1$, thì vị trí tương ứng trong chuỗi là: $index = x_1 \cdot 2^0 + x_2 \cdot 2^1 + x_3 \cdot 2^2$

📤 Dữ liệu ra

Với mỗi test:

- In `Yes` nếu tồn tại biểu thức thỏa mãn
- Nếu có, in thêm dòng biểu thức Boolean dạng chuẩn
- Nếu không thể dựng, in `No`

Yêu cầu biểu thức:

- Tuân thủ **đúng định dạng cú pháp**
- Không thêm hoặc bớt dấu ngoặc
- Không có dấu cách dư thừa

🧠 Tóm tắt đề

Bạn được cho bảng chân trị (truth table) của một hàm Boolean gồm n biến. Hãy kiểm tra xem có thể dựng biểu thức chỉ dùng `&`, `|` biểu diễn hàm này không. Nếu có, in biểu thức dạng chuẩn hóa theo quy tắc (và không vượt quá giới hạn toán tử và độ sâu). Nếu không có cách dựng đúng, in `No`.

Bạn nói đúng — mình sẽ dịch và **tóm tắt đầy đủ**, bao gồm cả phần **Input** và **Output**.

🅶 Bài G - Giả thuyết Collatz

📘 Bản dịch tiếng Việt

Lynk đang nghiên cứu về **giả thuyết Collatz**. Giả thuyết này phát biểu như sau:

- Bắt đầu với một số nguyên dương n.
  - Nếu n là **số chẵn**, ta chia đôi: $n \to \frac{n}{2}$
  - Nếu n là **số lẻ**, ta nhân 3 rồi cộng 1: $n \to 3n + 1$
- Lặp lại quá trình trên vô hạn lần. Giả thuyết cho rằng **dù bắt đầu từ số nào, cuối cùng cũng sẽ về 1**.

Lynk đã giải quyết giả thuyết trên và chuyển sang nghiên cứu **một biến thể mới**. Cụ thể:

> *"Với hai tham số nguyên dương **A** và **B** (nguyên tố cùng nhau), quá trình mới như sau:"*

- Bắt đầu với số nguyên dương n
  - Nếu n chia hết cho A: $n \to \frac{n}{A}$
  - Nếu không chia hết: $n \to n + B$

🧠 Mục tiêu

Cho trước A, B, và n, xác định xem **sau một số hữu hạn bước**, liệu n có **trở về lại chính nó** không.

Lưu ý: không yêu cầu về 1 như Collatz gốc, chỉ cần **trở lại chính nó sau hữu hạn bước** (tức là **vào chu trình với chính mình**).

📥 Input

- Dòng đầu tiên: một số nguyên T $(1 \leq T \leq 10^5)$ — số lượng test.
- Với mỗi test:
  - Một dòng gồm 3 số nguyên A, B, n:

$$2 \leq A \leq 10^9, \quad 1 \leq B \leq 10^9, \quad 1 \leq n \leq 10^{18}$$

  - Đảm bảo $\gcd(A, B) = 1$

📤 Output

- Với mỗi test, in:
  - `Yes` nếu n sẽ **quay trở lại chính nó** sau hữu hạn bước.
  - `No` nếu không bao giờ quay lại được.

📌 Ví dụ

Input

```
7
2 1 1
2 1 2
2 1 3
2 1 100
314 159 265
314 159 2653
314 159 26535
```

Output

```
Yes
Yes
No
No
Yes
Yes
No
```

✍️ Tóm tắt đề

Bạn được cho biến thể của giả thuyết Collatz với 2 tham số A, B.
Tại mỗi bước:

- Nếu $n \mod A = 0$: chia n cho A
- Ngược lại: cộng B

Hỏi rằng bắt đầu từ n, quá trình này có **quay lại được chính** n sau số hữu hạn bước không?
Với mỗi test, in `Yes` hoặc `No` tương ứng.

Bạn nói đúng, lần này mình sẽ **dịch đầy đủ và tóm tắt rõ ràng cả phần Input và Output**.

🅷 Bài H - Bảo tàng hình bậc thang (Staircase Museum)

📘 Bản dịch tiếng Việt

Một bảo tàng trưng bày nghệ thuật trừu tượng được xây dựng với hình dạng giống **cầu thang**.

Mặt đất được mô hình hóa như một **mặt phẳng vô hạn gồm các ô vuông đơn vị**, với các cạnh song song trục tọa độ x, y.
Mỗi ô vuông có tọa độ là một cặp số nguyên (x, y), trong đó ô có góc dưới bên trái là $(x-1, y-1)$, góc trên bên phải là (x, y).

Cấu trúc bảo tàng được định nghĩa bằng hai dãy số:

- $l_1, l_2, ..., l_n$
- $r_1, r_2, ..., r_n$

Với mỗi cột x từ 1 đến n, các ô vuông từ hàng l_x đến hàng r_x (tức y chạy từ l_x đến r_x) **là phần bên trong của bảo tàng**.

> *"Dãy này mô tả hình dạng "bậc thang":*
> *Với mọi $i = 1$ đến $n - 1$, ta có:"*
>
> - *"$l_i \leq l_{i+1}$"*
> - *"$r_i \leq r_{i+1}$"*

🎯 Yêu cầu

Có một nhóm nghệ sĩ đến tham quan bảo tàng. Mỗi người sẽ **đứng tại một ô vuông** trong bảo tàng (nội bộ hoặc biên).

Quy tắc quan trọng:
Hai nghệ sĩ **có thể nhìn thấy nhau** nếu **đoạn thẳng nối vị trí của họ** nằm hoàn toàn bên trong hoặc trên biên của bảo tàng.

> *"Do nghệ thuật trừu tượng rất khó hiểu, nên **không nghệ sĩ nào muốn nhìn thấy người khác** để tránh phân tâm."*

✅ Nhiệm vụ của bạn

Tìm **số lượng tối đa nghệ sĩ** có thể được đặt vào các ô trong bảo tàng sao cho **không cặp nào nhìn thấy nhau**.

📥 Input

- Dòng đầu tiên: số bộ test T $(1 \leq T \leq 10^5)$

Với mỗi test:

- Dòng đầu: số nguyên n — số cột của bảo tàng $(1 \leq n \leq 5 \times 10^5)$
- Tiếp theo n dòng: mỗi dòng chứa hai số nguyên l_i, r_i $(1 \leq l_i \leq r_i \leq 10^9)$ — xác định phạm vi hàng ở cột i

Bảo đảm:
Tổng tất cả n trên mọi test **không vượt quá** 5×10^5

📤 Output

- Với mỗi test, in ra **một dòng chứa một số nguyên duy nhất** — là số lượng nghệ sĩ tối đa sao cho **không ai nhìn thấy ai**.

📌 Ví dụ minh họa

Input:

```
4
3
1 2
1 3
1 3
3
1 2
2 3
3 3
3
1 1
1 3
3 3
4
1 2
2 3
3 4
4 5
```

Output:

```
2
3
3
4
```

✍️ Tóm tắt đề

- Mỗi test mô tả hình dạng **bảo tàng hình bậc thang**, gồm nhiều cột với chiều cao từng cột xác định bởi đoạn từ l_i đến r_i.
- Mỗi nghệ sĩ được đặt tại 1 ô trong bảo tàng.

- Hai nghệ sĩ nhìn thấy nhau nếu đoạn nối thẳng giữa họ nằm hoàn toàn trong bảo tàng.
- Hãy tìm **tối đa bao nhiêu nghệ sĩ có thể đặt mà không ai nhìn thấy ai**.

Với mỗi test, in ra kết quả là **một số nguyên duy nhất**.

Dưới đây là bản dịch và tóm tắt tiếng Việt đầy đủ của **Problem I - Color-Balanced Tree**:

🅸 Bài I - Cây Cân Bằng Màu (Color-Balanced Tree)

📘 Bản dịch tiếng Việt

Có nhiều loại cây nhị phân cân bằng, như cây cân bằng chiều cao (height-balanced tree), cân bằng trọng số (weight-balanced tree), hoặc cây với các nút có màu như cây đỏ-đen (red-black tree).
Vậy nếu ta định nghĩa một loại mới là **cây cân bằng màu** thì sao?

Một **cây có số đỉnh chẵn** (gồm $2n$ đỉnh) được gọi là **cây cân bằng màu** nếu thỏa mãn 3 điều kiện sau:

1. Mỗi đỉnh được gán **một màu**: **xanh lam (cyan)** hoặc **trắng (white)**.
2. **Số lượng đỉnh màu xanh và trắng bằng nhau.**
3. Trên **mọi đường đi đơn** trong cây, **độ lệch tuyệt đối** giữa số đỉnh màu xanh và trắng **không vượt quá 3**.

> *"Đường đi đơn là đường không lặp lại đỉnh."*

🎯 Nhiệm vụ

Cho một cây gồm $2n$ đỉnh và các cạnh, bạn cần **gán màu (cyan hoặc trắng) cho từng đỉnh** sao cho thỏa mãn các điều kiện trên.

- Nếu làm được, in ra dãy nhị phân độ dài $2n$:
  - `0` nghĩa là đỉnh đó màu **cyan**
  - `1` nghĩa là đỉnh đó màu **trắng**
- Nếu **không thể tô màu hợp lệ**, in ra `-1`.

📥 Input

- Dòng đầu tiên: một số nguyên T $(1 \leq T \leq 10^4)$ — số lượng test case.
- Với mỗi test:
  - Dòng đầu: số nguyên n $(1 \leq n \leq 10^5)$ — cây có tổng $2n$ đỉnh.
  - Sau đó là $2n - 1$ dòng, mỗi dòng chứa hai số nguyên u, v — một cạnh nối giữa đỉnh u và v.

Bảo đảm:

- Dữ liệu luôn tạo thành **một cây hợp lệ**
- Tổng tất cả n trên mọi test không vượt quá 10^5

📤 Output

- Với mỗi test:

- Nếu tồn tại cách tô màu hợp lệ: in ra $2n$ số nguyên $c_1, c_2, ..., c_{2n}$ (0 hoặc 1), cách nhau bởi dấu cách.
- Nếu không tồn tại: in ra `-1`.

📌 Ví dụ

Input

```
4
3
1 2
2 3
2 4
4 5
4 6
3
1 2
1 3
1 4
1 5
1 6
4
1 2
2 3
3 4
4 5
5 6
6 7
7 8
5
1 2
1 4
4 3
4 5
5 6
5 8
```

```
8 7
8 9
9 10
```

Output

```
0 0 0 1 1 1
1 1 1 0 0 0
1 1 1 0 0 0 1 0
1 1 1 0 0 0 1 0 1 0
```

✍️ Tóm tắt đề

- Cho một cây có **số đỉnh chẵn**, yêu cầu bạn gán **màu xanh (0)** hoặc **trắng (1)** cho từng đỉnh sao cho:
  1. **Số đỉnh mỗi màu bằng nhau**
  2. **Trên mọi đường đi trong cây, độ lệch số màu không vượt quá 3**
- Nếu làm được thì in ra dãy màu cho từng đỉnh. Nếu không thì in `-1`.

Dưới đây là bản dịch đầy đủ và tóm tắt bằng tiếng Việt của **Problem J - The Mysterious Shop**:

🅹 Bài J - Cửa hàng bí ẩn

📘 Bản dịch tiếng Việt

Ở bên ngoài lục địa Valoran có một **cửa hàng bí ẩn**. Không ai biết ai là người điều hành, nhưng vào **mỗi đầu ngày**, hàng hóa trong cửa hàng sẽ được **tái tạo lại một cách ngẫu nhiên**:

- Có **n loại vật phẩm**, với trọng lượng từ **1 đến n**.
- Với mỗi trọng lượng i, sẽ có **xác suất** $\frac{1}{2}$ là:
  - Có đúng **một** vật phẩm trọng lượng i, hoặc
  - Không có vật phẩm nào cả.

> *"Nghĩa là với mỗi trọng lượng, việc có vật phẩm hay không là độc lập và ngẫu nhiên với xác suất 1/2."*

🎯 Chiến lược của Kevin

Kevin có một chiếc túi có sức chứa tối đa **trọng lượng** n. Mỗi ngày, anh là **người đầu tiên đến cửa hàng** và chọn vật phẩm theo cách sau:

- **Duyệt từ nặng nhất đến nhẹ nhất**.
- Nếu vật phẩm tồn tại và **vừa vào túi** (không vượt quá trọng lượng tối đa), thì anh lấy.
- Tiếp tục đến khi xét hết tất cả vật phẩm.

> *"Kevin có thể mua mọi thứ anh bỏ vào túi — tiền không phải là vấn đề."*

✅ Nhiệm vụ của bạn

Kevin muốn đánh giá vận may của mình: mỗi ngày anh mang về khối lượng khác nhau.

Bạn cần tính xác suất mà **tổng trọng lượng Kevin mang về là** i với mọi $i \in [0, n]$.

📥 Input

- Một dòng duy nhất chứa số nguyên n $(1 \leq n \leq 2 \times 10^5)$ — số loại vật phẩm, cũng là sức chứa túi của Kevin.

📤 Output

- In ra $n + 1$ số nguyên cách nhau bởi dấu cách.
- Số thứ i (tính từ 1) là **xác suất** Kevin mang về tổng trọng lượng $i - 1$, được tính như sau:
  - Nhân với 2^n
  - Rồi **modulo 998244353**

> *"Đảm bảo rằng sau khi nhân với 2^n, kết quả là số nguyên."*

📌 Ví dụ minh họa

Input

```
2
```

Xác suất các tổ hợp:

- `{}`: không lấy gì → trọng lượng = 0

- `{1}`: lấy 1 → trọng lượng = 1
- `{2}`: lấy 2 → trọng lượng = 2
- `{1, 2}`: duyệt từ 2 xuống → chọn 2 đầu tiên (vừa túi), bỏ qua 1 → trọng lượng = 2

Tổng trọng lượng	Xác suất	Nhân $2^2 = 4$
0	1/4	1
1	1/4	1
2	1/2	2

Output

```
1 1 2
```

✍️ Tóm tắt đề

- Có n vật phẩm với trọng lượng từ 1 đến n.
- Mỗi vật phẩm tồn tại với xác suất 50%.
- Kevin chọn vật phẩm từ nặng đến nhẹ, bỏ vào túi sao cho không vượt quá tổng trọng lượng n.
- Tính **xác suất (nhân 2^n, lấy mod)** rằng Kevin mang về tổng trọng lượng đúng bằng i, với $i \in [0..n]$.

Dưới đây là **bản dịch tiếng Việt đầy đủ** và phần **tóm tắt** của **Problem K - Exploration Boundary**:

🅺 Bài K - Biên khám phá trong Dijkstra (Exploration Boundary)

📘 Bản dịch tiếng Việt

Gần đây, bài báo “Universal Optimality of Dijkstra via Beyond-Worst-Case Heaps” trên hội nghị FOCS đã thu hút sự chú ý của Fuyu. Trong đó, có một khái niệm gọi là **“biên khám phá” (exploration boundary)** trong thuật toán Dijkstra:

- Khi ta **tạm dừng thuật toán Dijkstra tại một thời điểm bất kỳ**, **biên khám phá** là tập các **đỉnh đã được chèn vào hàng đợi (priority queue)** nhưng **chưa bị xóa khỏi hàng đợi** (tức là khoảng cách chính thức chưa được xác định).

🧠 Mô tả cụ thể:

Thuật toán Dijkstra hoạt động như sau:

```
1. S ← ∅ (tập đỉnh đã khám phá)
2. Q ← ∅ (priority queue chứa các cặp (khoảng cách, đỉnh))
3. P ← mảng con trỏ vào Q
4. D ← mảng lưu khoảng cách tốt nhất từ đỉnh xuất phát
5. Đưa đỉnh xuất phát s vào Q với khoảng cách 0
6. While Q không rỗng:
    - Lấy đỉnh u có khoảng cách nhỏ nhất khỏi Q
    - Đưa u vào S
    - Với mỗi đỉnh kề v chưa được khám phá:
        - Nếu v chưa có trong Q, chèn giá trị giả
```

```
        - Cập nhật khoảng cách D[v] = min(D[v], D[u] + trọng số)
        - Giảm khóa Q[v]
```

Tập các đỉnh đang nằm trong Q tại bất kỳ thời điểm nào **trước khi bước 6 chạy** được gọi là **biên khám phá**.

🧩 Vấn đề đặt ra

Cho một đồ thị vô hướng G và một số tập đỉnh **được kỳ vọng là các biên khám phá**, với đỉnh xuất phát $s = 1$, bạn cần xác định:

> *"Có thể gán trọng số nguyên dương (≤ 10^9) cho các cạnh sao cho:"*
>
> - *"**Không có hai đỉnh nào có cùng khoảng cách từ đỉnh 1**"*
> - *"**Mỗi tập đỉnh được cung cấp phải là biên khám phá** trong quá trình chạy Dijkstra's Algorithm từ đỉnh 1."*

📥 Input

- Dòng đầu: số bộ test T $(1 \leq T \leq 10^5)$
- Mỗi test gồm:
  - Dòng đầu: hai số nguyên n, m $(2 \leq n \leq 2 \times 10^5, 1 \leq m \leq 2 \times 10^5)$ — số đỉnh và cạnh.
  - m dòng tiếp theo: mỗi dòng chứa hai số u, v (cạnh vô hướng)
  - Một dòng: số tập biên khám phá k $(1 \leq k \leq 2 \times 10^5)$
  - Tiếp theo k dòng: mỗi dòng bắt đầu bằng số b_i, theo sau là b_i số nguyên đại diện cho tập đỉnh của biên khám phá

Đảm bảo:

- Đồ thị liên thông, không có cạnh trùng hay khuyên
- Tổng tất cả n, m, và $\sum b_i$ trên mọi test ≤ 10^6

📤 Output

- Với mỗi test:
  - Nếu tồn tại cách gán trọng số thỏa mãn:
    - In `Yes`
    - In tiếp một dòng gồm m số — trọng số ứng với các cạnh theo thứ tự đầu vào
  - Nếu không thể: in `No`

📌 Ví dụ

Input

```
2
8 10
1 2
1 3
1 4
1 5
2 6
3 6
3 7
4 8
5 8
```

```
7 8
2
4 3 4 5 6
4 4 5 6 7
5 4
1 2
1 3
2 4
2 5
2
2 3 4
2 2 5
```

Output

```
Yes
1 2 3 5 3 5 4 5 6 3
No
```

✍️ Tóm tắt đề

- Cho đồ thị vô hướng và một số tập đỉnh — được cho là **biên khám phá**.
- Bạn cần kiểm tra xem có thể gán trọng số (số nguyên dương ≤ 10^9) cho các cạnh sao cho:
  1. Mỗi đỉnh có khoảng cách khác nhau từ đỉnh 1
  2. Mỗi tập đỉnh được cho thực sự là biên khám phá trong một lần chạy Dijkstra’s từ đỉnh 1
- In `Yes` và dãy trọng số nếu được, còn không thì in `No`.

Dưới đây là bản dịch đầy đủ và tóm tắt tiếng Việt của **Problem L - Shiori**:

🅻 Bài L - Shiori

📘 Bản dịch tiếng Việt

"Bình thường" có nghĩa là gì vậy nhỉ?
Tomorin, một chú chim cánh cụt, có một bộ sưu tập đá khổng lồ! Để ngắm chúng, Tomorin đặt đá thành **n đống** trên mặt đất. Ban đầu, đống thứ i có a_i viên đá (có thể là 0).

Tomorin sẽ thực hiện một số thao tác để **di chuyển và chỉnh sửa các đống đá**, và bạn cần giúp cô ấy thực hiện các thao tác này.

Có 3 loại thao tác:

🔧 Các thao tác:

`1 l r v` — Gán lại giá trị

Tomorin xây lại các đống đá từ vị trí l đến r, mỗi đống có đúng v viên đá:
→ Với mỗi $i \in [l, r]$, gán $a_i = v$

`2 l r` — Tăng theo `mex`

Tomorin quan sát các đống đá từ vị trí l đến r. Là một chú chim thích số tự nhiên, cô ấy **không muốn thiếu số nào nhỏ nhất**.
→ Gọi $w = \text{mex}(a_l, a_{l+1}, ..., a_r)$: là **số tự nhiên nhỏ nhất chưa xuất hiện**
→ Với mỗi $i \in [l, r]$, tăng $a_i += w$

`3 l r` — Tính tổng

In ra tổng số viên đá từ vị trí l đến r:
→ Tính $\sum_{i=l}^{r} a_i$

📥 Input

- Dòng đầu: hai số nguyên n, m — số lượng đống đá và số thao tác $(1 \leq n, m \leq 5 \times 10^5)$
- Dòng tiếp: n số nguyên $a_1, a_2, ..., a_n$ — số viên đá ban đầu $(0 \leq a_i \leq 5 \times 10^5)$
- m dòng tiếp: mỗi dòng là một thao tác:
  - Dạng `1 l r v`
  - Hoặc `2 l r`
  - Hoặc `3 l r`

📤 Output

- Với mỗi thao tác loại `3`, in ra một dòng chứa tổng $a_l + a_{l+1} + ... + a_r$

📌 Ví dụ

Input

```
5 8
0 7 2 1 0
1 2 4 0
```

```
2 1 3
2 3 4
3 1 3
1 2 3 4
3 1 4
2 1 5
3 2 5
```

Output

```
5
11
22
```

✍️ Tóm tắt đề

Bạn quản lý một mảng số nguyên a có thể lên đến 500,000 phần tử, với 3 loại thao tác:

1. Gán đoạn [l, r] thành cùng một giá trị
2. Cộng thêm `mex` của đoạn [l, r] vào tất cả các phần tử trong đoạn đó
3. Truy vấn tổng đoạn [l, r]

Yêu cầu xử lý **hiệu quả** cho tất cả các thao tác.