universal cup 3 stage 21 : Ookayama

Created: 5/4/2025 22:27
Updated: 5/4/2025 22:32
Exported: 5/4/2025 22:30

🇻🇳 Bài A. Đừng Phát Hiện Chu Trình

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho một đồ thị vô hướng **G** gồm **N** đỉnh được đánh số từ **1 đến N**. Ban đầu, đồ thị không có cạnh nào.
Bạn sẽ thêm **M cạnh** vào đồ thị **G**, với hình dạng cuối cùng của đồ thị đã được xác định. Cạnh thứ **i** (với $1 \leq i \leq M$) là cạnh nối hai đỉnh u_i và v_i.

> *"Chúng ta sẽ gọi đây là **cạnh i**."*

Yêu cầu:
Hãy xác định xem **có tồn tại một hoán vị** $(P_1, P_2, ..., P_M)$ của dãy $(1, 2, ..., M)$ sao cho khi thêm các cạnh theo thứ tự này vào đồ thị **G**, bạn **không bao giờ thêm một cạnh mà trong đồ thị hiện tại đã tồn tại chu trình chứa** u_{P_i} **hoặc** v_{P_i}. Nếu tồn tại, hãy đưa ra một ví dụ về hoán vị đó.

🧠 Điều kiện chính

Với mỗi bước $i = 1, 2, ..., M$, ta làm:

1. Nếu **đã tồn tại chu trình** trong đồ thị hiện tại **chứa** u_{P_i} **hoặc** v_{P_i}, thì **không hợp lệ** → dừng lại.

2. Ngược lại, **thêm cạnh** P_i vào đồ thị.

📥 Input

Gồm nhiều bộ test, với định dạng như sau:

```
T
case1
case2
...
caseT
```

Trong đó, mỗi test case có dạng:

```
N M
u1 v1
u2 v2
...
uM vM
```

Ràng buộc:

- $1 \leq T \leq 2000$
- Với mỗi test:
  - $2 \leq N \leq 4000$
  - $1 \leq M \leq 4000$
  - $1 \leq u_i, v_i \leq N$
  - Tổng N trên tất cả test ≤ 4000
  - Tổng M trên tất cả test ≤ 4000
  - Đồ thị cuối cùng **không có cạnh trùng hoặc cạnh tự nối (simple graph)**

📤 Output

Với mỗi test case:

- Nếu **tồn tại hoán vị** $(P_1, ..., P_M)$ thỏa mãn điều kiện → in ra dãy hoán vị đó (gồm M số, cách nhau bởi dấu cách)
- Nếu **không tồn tại**, in ra `-1`

📌 Ví dụ

Input:

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

Output:

```
1 4 3 2
```

📝 Tóm tắt đề bài

- Bạn cần thêm các cạnh vào một đồ thị đang rỗng.
- Mỗi lần thêm cạnh u, v, nếu **có chu trình chứa u hoặc v** trong đồ thị hiện tại → không được phép thêm.

- Cần tìm một **thứ tự thêm cạnh hợp lệ** sao cho **toàn bộ M cạnh được thêm vào** mà **không vi phạm điều kiện**.
- Nếu không tồn tại thứ tự nào → in `-1`.

Bạn có muốn mình viết luôn hướng giải chi tiết không?

🇻🇳 Bài B. Tự Thanh Toán

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho một **dãy số S** có độ dài **N**, bao gồm các số **1, 2, 3**.
Nhiệm vụ của bạn là:
Tìm **số lượng dãy A** (gồm các số **1 và 2**) sao cho nếu thực hiện **quy trình đặc biệt** bên dưới lên dãy A, thì kết quả thu được là **chính xác dãy S**.

> *"Do số lượng dãy A có thể rất lớn, bạn cần in ra kết quả **modulo 998244353**."*

⚙️ Quy trình sinh dãy T từ dãy A

Khởi tạo dãy rỗng `T`, và thực hiện các bước sau **trên dãy A**:

1. Khởi tạo biến đếm `C = 0`.

2. Nếu dãy A **có ít nhất một số 1**, loại bỏ **số 1 đầu tiên**, và `c += 1`.
3. Nếu dãy A **vẫn còn phần tử**, loại bỏ **phần tử đầu tiên** x, và `c += x`.
4. Thêm giá trị `c` vào cuối dãy T.
5. Nếu A đã rỗng, kết thúc. Nếu chưa, lặp lại từ bước 1.

📥 Input

```
N
S1 S2 ... SN
```

- $1 \leq N \leq 10^6$
- $1 \leq S_i \leq 3$

📤 Output

- In ra **số lượng dãy A** hợp lệ sao cho **khi áp dụng quy trình trên, kết quả là dãy T = S**.
- Kết quả lấy **modulo 998244353**.

📌 Ví dụ

Input

```
2
```

```
3 2
```

Output

```
5
```

Các dãy A hợp lệ trong ví dụ:

- (1, 2, 2)
- (2, 1, 2)
- (2, 2, 1)
- (2, 1, 1, 1)
- (1, 2, 1, 1)

📝 Tóm tắt đề bài

- Ta cần xây dựng bao nhiêu dãy A (chỉ gồm 1 và 2) sao cho khi thực hiện một quy trình đặc biệt sẽ tạo ra đúng dãy đầu ra S.
- Quy trình này bao gồm:
  - Xoá 1 đầu tiên nếu có, tăng C lên 1
  - Xoá phần tử đầu tiên còn lại, tăng C theo giá trị đó (1 hoặc 2)
  - Thêm C vào T
  - Lặp lại cho đến khi A rỗng
- Trả về số dãy A hợp lệ sinh ra T = S, **modulo 998244353**.

Bạn có muốn mình giải thích ý tưởng và hướng tiếp cận thuật toán cho bài này không?

🇻🇳 Bài C. Cây đoạn (Segment Tree)

⏱️ **Giới hạn thời gian:** 6 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho một đồ thị vô hướng **G** với $2^N + 1$ đỉnh và $2^{N+1} - 1$ cạnh.

- Đỉnh được đánh số từ **0 đến** 2^N.
- Mỗi cạnh mang một **loại** từ **0 đến N** (tổng cộng có $N + 1$ loại).

Với mỗi loại i (từ 0 đến N), tồn tại đúng 2^i cạnh thuộc loại này:

- Các cạnh có số thứ tự từ 2^i đến $2^i + 2^i - 1$
- Cạnh số $2^i + j$ (với $0 \leq j < 2^i$) là cạnh nối hai đỉnh:
  - $j \times 2^{N-i}$ và $(j + 1) \times 2^{N-i}$
  - Chiều dài của cạnh đó là C_{2^i+j}

📌 *Gợi ý:* Cấu trúc đồ thị này giống như cây segment tree hoàn chỉnh theo cách gán cạnh.

⚙️ Bạn cần xử lý Q truy vấn, gồm 2 loại:

- **1 j x**: Gán lại chiều dài của cạnh thứ **j** thành **x**.
- **2 s t**: Tính **độ dài ngắn nhất** giữa hai đỉnh **s và t**.

📥 Input

```
N
C1 C2 ... C_(2^{N+1}-1)
Q
query1
query2
...
queryQ
```

- $1 \leq N \leq 18$
- $1 \leq C_j \leq 10^7$
- $1 \leq Q \leq 2 \times 10^5$
- Truy vấn 1: `1 j x` với $1 \leq j \leq 2^{N+1} - 1$, $1 \leq x \leq 10^7$
- Truy vấn 2: `2 s t` với $0 \leq s < t \leq 2^N$
- Có ít nhất một truy vấn loại 2

📤 Output

Với mỗi truy vấn loại **2 s t**, in ra **độ dài đường đi ngắn nhất** từ đỉnh **s đến t**.

🧠 Tóm tắt ý tưởng đề bài

- Đồ thị có cấu trúc đặc biệt, giống **cây segment tree**, các cạnh được gán theo cấp độ từ gốc đến lá.

- Mỗi cạnh nối hai đỉnh cách nhau một lượng đều đặn 2^{N-i}
- Bạn cần hỗ trợ:
  - **Cập nhật trọng số 1 cạnh**
  - **Truy vấn đường đi ngắn nhất giữa 2 đỉnh**

📌 Ví dụ Input

```
3
7 1 14 3 9 4 8 2 6 5 5 13 8 2 3
10
2 0 1
2 0 4
2 4 6
2 4 8
2 3 5
1 6 30
2 3 5
2 4 6
1 1 10000000
2 0 8
```

📌 Ví dụ Output

```
2
1
4
8
17
18
```

```
13
15
```

Bạn có muốn mình giải thích cách xây dựng cây và thuật toán để xử lý hiệu quả các truy vấn không?

🇻🇳 Bài D. A xor B cộng C

⏱️ **Giới hạn thời gian:** 8 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Cho ba số nguyên không âm **A, B, C**. Ta định nghĩa một **dãy số nguyên không âm** $X = (X_1, X_2, X_3, \ldots)$ như sau:

- $X_1 = A$
- $X_2 = B$
- Với mọi $i \geq 1$:

$$X_{i+2} = (X_i \oplus X_{i+1}) + C$$

(Trong đó, $\oplus$ là phép XOR bit)

Bạn được cho thêm một số nguyên dương N, nhiệm vụ của bạn là **tính** $X_N \mod 998244353$.

📥 Input

Dòng duy nhất gồm 4 số nguyên:

```
A B C N
```

- $0 \leq A, B, C < 2^{20}$
- $1 \leq N \leq 10^{18}$

📤 Output

- In ra giá trị $X_N \mod 998244353$

📌 Ví dụ

Input:

```
1 2 3 4
```

Output:

```
7
```

👉 Diễn giải:
Dãy X bắt đầu là:

```
X = [1, 2]
X3 = (1 xor 2) + 3 = 3 + 3 = 6
X4 = (2 xor 6) + 3 = 4 + 3 = 7
```

→ $X_4 = 7$

🧠 Tóm tắt đề bài

- Cho công thức truy hồi:

$$X_{i+2} = (X_i \oplus X_{i+1}) + C$$

- Nhiệm vụ là tính nhanh phần tử thứ N trong dãy, modulo 998244353
- Dãy được xây dựng bằng phép XOR và cộng thêm hằng số C

Bạn có muốn mình trình bày hướng giải nhanh cho bài này, đặc biệt với N lớn đến 10^{18} không?

🇻🇳 Bài E. Du Hành Lặp Lại (ReTravel)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Trên mặt phẳng tọa độ xy, có **N điểm** được đánh số từ **1 đến N**. Tọa độ của điểm thứ **i** là (X_i, Y_i).

Một con robot đang ở gốc tọa độ $(0, 0)$. Nhiệm vụ của bạn là điều khiển robot đi qua **tất cả các điểm theo thứ tự 1 → 2 → ... → N**.

⚙️ Robot có một chuỗi ký tự `s`, ban đầu là chuỗi rỗng.

Robot có thể thực hiện 4 loại thao tác sau:

1. ➡️ **Tăng hoành độ (x += 1)**
   - Ghi thêm ký tự `'X'` vào cuối chuỗi `s`
   - Chi phí: **1**
2. ⬆️ **Tăng tung độ (y += 1)**
   - Ghi thêm ký tự `'Y'` vào cuối chuỗi `s`
   - Chi phí: **1**
3. ⬅️ **Giảm hoành độ (x -= 1)**
   - Chỉ thực hiện nếu **ký tự cuối chuỗi `s` là 'X'**, sau đó xóa ký tự đó
   - Chi phí: **0**
4. ⬇️ **Giảm tung độ (y -= 1)**
   - Chỉ thực hiện nếu **ký tự cuối chuỗi `s` là 'Y'**, sau đó xóa ký tự đó
   - Chi phí: **0**

🎯 Mục tiêu

Tìm **tổng chi phí nhỏ nhất** để robot đi qua **tất cả các điểm từ 1 đến N theo đúng thứ tự**.

📥 Input

```
N
X1 Y1
X2 Y2
...
XN YN
```

- $1 \leq N \leq 500$
- $0 \leq X_i, Y_i \leq 10^9$

📤 Output

In ra **chi phí tối thiểu** để đi qua tất cả các điểm theo thứ tự đã cho.

📌 Ví dụ

Input:

```
2
3 3
1 2
```

Output:

```
6
```

🧾 Diễn giải:

- Đến điểm (3,3): dùng thao tác 1 (x+1) **1 lần**, thao tác 2 (y+1) **3 lần**, thao tác 1 **2 lần** → chi phí = 6

- Quay về (1,2): dùng thao tác 3 (x-1) **2 lần**, thao tác 4 (y-1) **1 lần** → chi phí = 0
→ Tổng chi phí: **6**

🧠 Tóm tắt đề bài

- Di chuyển từ điểm này đến điểm kia với 2 loại thao tác tốn phí (đi tới) và 2 thao tác miễn phí (quay lại).
- Có thể "xóa lùi" nếu quay lại bằng cùng hướng (X hoặc Y tương ứng).
- Bài toán yêu cầu tận dụng cấu trúc của chuỗi `s` như **một ngăn xếp (stack)** để tiết kiệm chi phí.
- Cần tìm cách tối ưu thứ tự thao tác **đi tới** và **xóa ngược** sao cho robot luôn đi đúng hướng và ít tốn phí nhất.

Bạn có muốn mình trình bày thuật toán chi tiết và cách triển khai hiệu quả cho bài này không?

🇻🇳 Bài F. Origami Warp — Biến Hình Gấp Giấy

⏱️ **Giới hạn thời gian:** 3 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho **N đường thẳng** trong mặt phẳng tọa độ xy.

- Đường thẳng thứ **i** (với $1 \leq i \leq N$) đi qua hai điểm phân biệt (a_i, b_i) và (c_i, d_i).

- Hai đường thẳng đầu tiên luôn là:
  - $i = 1$: Trục hoành (x-axis) — đi qua $(0, 0)$ và $(1, 0)$
  - $i = 2$: Trục tung (y-axis) — đi qua $(0, 0)$ và $(0, 1)$

🤖 Alice đang đứng trên mặt phẳng tại điểm $S = (x, y)$.

Cô ấy có thể thực hiện **nhiều lần (kể cả 0 lần)** phép biến đổi sau:

> *"Chọn **một đường thẳng** trong N đường, rồi **lật gương đối xứng** qua đường thẳng đó."*

📍 Khái niệm Reachable

Một điểm P được gọi là **reachable** từ điểm S nếu:

> *"Với mọi số thực $\varepsilon > 0$, tồn tại một điểm Q sao cho:"*
>
> - *"Khoảng cách từ Q đến P **≤ ε**, và"*
> - *"Alice **có thể đến** Q từ S bằng một số hữu hạn phép đối xứng"*

👉 Tức là Alice không cần đến chính xác P, nhưng có thể đến **rất gần** với bất kỳ độ chính xác nào.

📥 Input

```
T
<test case 1>
```

```
...
<test case T>
```

Mỗi test case có dạng:

```
N
a1 b1 c1 d1
a2 b2 c2 d2
...
aN bN cN dN
Q
X1 Y1 Z1 W1
...
XQ YQ ZQ WQ
```

Giới hạn:

- $1 \leq T \leq 100$
- $2 \leq N \leq 2000$
- $1 \leq Q \leq 2000$
- Tọa độ $a_i, b_i, c_i, d_i, X_i, Y_i, Z_i, W_i$ thuộc $[-10^8, 10^8]$
- Các đường thẳng là **phân biệt và hợp lệ**
- Tổng các truy vấn ≤ 200000

📤 Output

- Với mỗi truy vấn, in `Yes` nếu Alice **có thể đến được điểm (Z, W)** từ (X, Y) (theo định nghĩa "reachable"), ngược lại in `No`.

📌 Ví dụ

Input:

```
2
3
0 0 1 0
0 0 0 1
0 2 2 0
4
1 0 2 3
1 -2 -1 2
1 1 -1 0
3 3 3 3
3
0 0 1 0
0 0 0 1
-2 1 2 3
2
2 1 -1 5
-1 -1 3 3
```

Output:

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

🧠 Tóm tắt bài toán

- Có N đường thẳng, mỗi lần chọn 1 đường và lấy đối xứng điểm hiện tại qua nó.
- Ta có thể thực hiện thao tác đối xứng **vô số lần**.

- Hỏi: với chuỗi phép đối xứng, **ta có thể đến gần điểm đích bao nhiêu tùy ý không?** (theo định nghĩa "reachable").

🧮 Ý tưởng giải bài (tóm tắt)

- Phép đối xứng qua đường thẳng là **phép biến đổi affine tuyến tính**, tức là dạng:

$$T(x) = Mx + b$$

- Các phép đối xứng qua các đường tạo ra một **nhóm (group)** các phép biến đổi.
- Tập điểm reachable từ S là **tập con nhỏ nhất bất biến** dưới nhóm đó chứa S.
- Nếu nhóm sinh bởi các phép đối xứng **chứa một mạng lưới hai chiều (2D lattice)**, thì **mọi điểm đều reachable** (theo nghĩa trong đề).
- Giải thuật:
  - Với mỗi đường, ta lấy **vector pháp tuyến** để xác định cách nó "phản chiếu".
  - Dùng **tổ hợp tuyến tính** các vector để tìm xem:
    - Từ một điểm, ta có thể sinh ra tất cả điểm nguyên (hoặc xấp xỉ cực nhỏ) trong không gian hay không?
  - Sử dụng **mô đun 2** để biểu diễn hành vi lặp lại

Bạn có muốn mình phân tích và xây dựng chi tiết hướng giải hoặc đưa code cho bài này không?

🇻🇳 Bài G. Phân Kỳ và Hội Tụ (Diverge and Converge)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho **hai cây A và B**, mỗi cây có **N đỉnh**, đánh số từ **1 đến N**.

- Cây **A** có $N-1$ cạnh: mỗi cạnh thứ i nối hai đỉnh u_i và v_i.
- Cây **B** có $N-1$ cạnh: mỗi cạnh thứ j nối hai đỉnh x_j và y_j.

🎯 Nhiệm vụ của bạn là:

Tìm **hai hoán vị** $(P_1, P_2, ..., P_{N-1})$ **và** $(Q_1, Q_2, ..., Q_{N-1})$ sao cho thực hiện chuỗi hoán đổi sau vẫn đảm bảo cả A và B **luôn là cây**:

🔁 Quy trình hoán đổi (với mỗi $k = 1, 2, ..., N-1$):

1. **Trên cây A**:
   - **Xóa** cạnh (u_{P_k}, v_{P_k})
   - **Thêm** cạnh (x_{Q_k}, y_{Q_k})
   - Sau bước này, A vẫn là cây
2. **Trên cây B**:
   - **Xóa** cạnh (x_{Q_k}, y_{Q_k})

- **Thêm** cạnh (u_{P_k}, v_{P_k})
- Sau bước này, B vẫn là cây

✅ Ràng buộc

- Bảo đảm rằng sau mỗi bước hoán đổi, **cả A và B luôn là cây**.
- Bài toán đã đảm bảo **luôn tồn tại nghiệm hợp lệ**.

📥 Input

```
N
u1 v1
...
u(N-1) v(N-1)
x1 y1
...
x(N-1) y(N-1)
```

- $2 \leq N \leq 1000$
- A và B là cây (đồ thị có N đỉnh và N-1 cạnh, liên thông, không chu trình)

📤 Output

In ra hai dòng:

```
P1 P2 ... P(N-1)
Q1 Q2 ... Q(N-1)
```

- Đây là hai hoán vị của các số từ 1 đến $N-1$

📌 Ví dụ

Input:

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

Output:

```
3 1 2
2 1 3
```

Diễn giải:

- Cây A ban đầu là **đường thẳng** 1–2–3–4
- Cây B là **sao** với tâm tại 2: 1–2, 2–4, 2–3
- Sau các bước hoán đổi theo P, Q, A trở thành cây sao và B trở thành đường thẳng — **đúng là "swap structure"** như tên đề bài.

🧠 Tóm tắt đề bài

- Cho 2 cây A và B có cùng số đỉnh.
- Thực hiện $N - 1$ bước **hoán đổi cạnh** giữa A và B (cứ mỗi bước, đổi 1 cạnh từ A sang B và ngược lại).
- Mỗi bước phải đảm bảo **cả hai cây vẫn là cây**.
- Tìm hai hoán vị thứ tự thực hiện các cặp cạnh để đảm bảo yêu cầu.

Bạn có muốn mình trình bày cách xây dựng hoán vị hợp lệ hay code mẫu không?

🇻🇳 Bài H. Hai Tập Lồi (Two Convex Sets)

⏱️ **Giới hạn thời gian:** 4 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Một tập hợp điểm U trên mặt phẳng tọa độ được gọi là **tốt (good)** nếu **không có điểm nào nằm trong phần nội thất của bao lồi (convex hull) của tập đó**.

- (Tức là mỗi điểm trong U đều nằm trên biên của bao lồi — không bị "bao quanh").
- **Tập rỗng cũng được xem là tốt.**

Bạn được cho **N điểm phân biệt** $v_1, v_2, ..., v_N$ trên mặt phẳng, với:

- $v_i = (x_i, y_i)$

- **Không có 3 điểm nào thẳng hàng**.

🎯 Nhiệm vụ:

Hãy **đếm số lượng tập con** $S \subseteq V$ sao cho **cả hai tập**:

- S
- $V \setminus S$

đều là **tập tốt (good sets)**.

📥 Input

```
N
x1 y1
x2 y2
...
xN yN
```

- $1 \leq N \leq 40$
- $|x_i|, |y_i| \leq 10^6$
- Tất cả các điểm đều **khác nhau**
- Không có **3 điểm nào thẳng hàng**

📤 Output

- In ra **số lượng tập con** S thoả mãn điều kiện đề bài.

📌 Ví dụ

Input:

```
4
0 0
3 0
0 3
1 1
```

Output:

```
14
```

Giải thích:

- Có $2^4 = 16$ tập con của 4 điểm.
- Trong đó, chỉ có 2 tập **không thỏa mãn điều kiện** là:
  - Tập rỗng $\emptyset$ và tập đầy đủ V → vì $V \setminus S$ hoặc S sẽ là tập rỗng (cần loại).
- Các tập còn lại đều chia ra làm 2 phần và mỗi phần đều là "good set".

🧠 Tóm tắt đề bài

- Một tập là "good" nếu không có điểm nào nằm trong **nội thất** của **bao lồi** của tập đó.

- Ta cần **đếm số lượng cách chia tập V thành 2 phần S và $V \setminus S$** sao cho **cả hai đều là "good"**
- Tổng số tập con là 2^N, ta duyệt qua tất cả $S \subseteq V$, loại bỏ những S sao cho S hoặc $V \setminus S$ **không good**.

Bạn có muốn mình giải thích cách kiểm tra "good set" nhanh bằng thuật toán convex hull + duyệt bitmask không?

🇻🇳 Bài I. Cặp Gần Nhau (Near Pair)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB
📡 **Dạng bài:** *Tương tác (interactive)*

📘 Đề bài (dịch tiếng Việt)

Bạn được cung cấp 3 số nguyên N, K, và Q:

- Một hoán vị ẩn `a = (a₁, a₂, ..., aₙ)` của dãy $(1, 2, ..., N)$ được sinh ngẫu nhiên **trước khi bạn bắt đầu**.
- Bạn có thể thực hiện tối đa **Q truy vấn**.

❓ Một truy vấn như sau:

Bạn gửi lên một chuỗi nhị phân dài N, gọi là `s1s2...sN`, tương ứng với một tập con $S \subseteq \{1, 2, ..., N\}$:

- $s_i = 1$ nghĩa là bạn chọn vị trí i vào tập S, ngược lại thì không.

📩 **Phản hồi từ hệ thống:**
Một số nguyên T là **số lượng cặp chỉ số** (i, j) **với** $i < j$**, và** $|a_i - a_j| \leq K$, với $i, j \in S$.

🎯 Mục tiêu của bạn:

Tìm ra hai vị trí x và y sao cho:

- $a_x = 1$
- $a_y = N$

Bạn chỉ cần tìm tập $\{x, y\}$ **— không cần biết ai là 1 và ai là N.**

⚠️ Lưu ý tương tác

- Khi truy vấn: In ra dòng `` `? <chuỗi nhị phân>` `` rồi `` `flush` `` **ngay lập tức**.
- Khi trả lời: In ra dòng `` `! x y` `` (hai chỉ số), rồi **kết thúc chương trình ngay**.
- Nếu nhận được phản hồi `` `-1` ``, phải dừng lại ngay lập tức.

📥 Input từ hệ thống

```
N K Q
```

- $N = 20000$

- $1 \leq K \leq 10$
- $Q = 30 \cdot (K + 1)$

📤 Output

- Các dòng `? <bitmask>` cho đến khi bạn đưa ra câu trả lời.
- Khi tìm ra: in ra `! x y` rồi thoát ngay.

📌 Ví dụ (không thực tế vì N nhỏ)

```
Input:
5 2 90

Output:
? 11000
-> nhận về: 1
? 10011
-> nhận về: 2
! 2 4
```

🧠 Tóm tắt và hướng giải

- Hoán vị ẩn chứa một phần tử bằng 1 (nhỏ nhất) và một phần tử bằng N (lớn nhất).
- Bạn có thể truy vấn tập con để tìm **các cặp có hiệu nhỏ hơn hoặc bằng K**.

- Vì 1 và N **không có ai gần được ai (chênh lệch = N − 1)** → **không tạo ra cặp "gần" với ai cả**.
- ⇒ Nếu ta truy vấn từng phần nhỏ (block), phần nào có ít hoặc 0 cặp "near" ⇒ có thể chứa 1 hoặc N.
- Sau khi lọc ra các vị trí nghi ngờ, kiểm tra từng vị trí **bằng cách thêm nó vào tập khác** rồi xem tổng số cặp thay đổi bao nhiêu.

Bạn có muốn mình trình bày hướng giải chi tiết bằng chiến lược phân vùng + kiểm thử nhị phân và đưa code mẫu Python không?

🇻🇳 Bài J. Xây Dựng Lưới (Grid Construction)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho hai số nguyên dương H và W.
Bạn cần **vẽ lưới H hàng × W cột** bằng cách thực hiện nhiều lần thao tác sau:

🪛 Một thao tác "vẽ hình chữ U" gồm:

1. Chọn một ô lưới tại tọa độ (x, y) với $1 \leq x \leq H, 1 \leq y \leq W$
2. Chọn **3 trong 4 đoạn thẳng sau** (mỗi đoạn là cạnh độ dài 1 nối hai điểm lưới):
   - (x−1, y−1) → (x−1, y) (trên)

- (x−1, y−1) → (x, y−1) (trái)
- (x, y) → (x−1, y) (dưới)
- (x, y) → (x, y−1) (phải)

3. Vẽ 3 đoạn đó **miễn là không trùng với đoạn nào đã vẽ trước đó (trùng điểm thì được, không trùng đoạn)**

🎯 Mục tiêu

Dùng các thao tác trên để vẽ **toàn bộ các đoạn thẳng độ dài 1 tạo thành lưới H × W** (nghĩa là vẽ đủ cạnh của các ô lưới).

📥 Input

```
H W
```

- $1 \leq H, W \leq 1000$

📤 Output

- Nếu không thể vẽ được toàn bộ lưới:

```
No
```

- Nếu vẽ được:

```
Yes
S₁
S₂
...
Sₕ
```

Với mỗi ô (i, j):

- Nếu không dùng để vẽ: ký tự là `.`
- Nếu dùng để vẽ, ký hiệu là một trong:
  - `v`: bỏ cạnh **trên** (không vẽ đoạn (x−1, y−1) → (x−1, y))
  - `>`: bỏ cạnh **trái**
  - `<`: bỏ cạnh **dưới**
  - `^`: bỏ cạnh **phải**

🧠 Tóm tắt đề bài

- Mỗi thao tác vẽ tại một ô sử dụng **3 cạnh hình chữ U**.
- Không được vẽ trùng đoạn đã có.
- Mục tiêu là vẽ **tất cả đoạn dài 1** tạo nên lưới chuẩn.
- Hỏi: có thể hay không?

📌 Ví dụ

Input

```
3 3
```

Output

```
Yes
<<^
v.^
v>>
```

➡ 9 ô, nhưng chỉ dùng 8 ô để vẽ — bỏ ô giữa trống, vẫn đủ để hoàn thành lưới.

Bạn có muốn mình trình bày điều kiện khi nào **có thể / không thể** và hướng xây dựng lời giải cùng code Python mẫu không?

🇻🇳 Bài K. Tổng bằng Một (Sum is One)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Cho một dãy nhị phân $A = (A_1, A_2, \ldots, A_N)$ chỉ gồm các số **0 và 1**.

Từ đó, định nghĩa một **đồ thị vô hướng đơn giản** $G = (V, E)$ như sau:

🔷 Tập đỉnh V:

Gồm tất cả các cặp chỉ số (i, j) thỏa $1 \leq i < j \leq N$
→ Tổng số đỉnh: $\frac{N(N-1)}{2}$

🔷 Tập cạnh E:

Xét mọi **bộ ba chỉ số** (i, j, k) thỏa $1 \leq i < j < k \leq N$, nếu:

$$A_i + A_j + A_k = 1$$

thì **nối đỉnh (i, j)** với **đỉnh (j, k)**

Không có cạnh nào khác.

🎯 Yêu cầu:

Với mỗi test, **đếm số thành phần liên thông (connected components)** của đồ thị G

📥 Input

```
T
N₁
A₁₁ A₁₂ ... A₁N₁
...
Nₜ
Aₜ₁ Aₜ₂ ... AₜNₜ
```

- $1 \leq T \leq 10^5$
- $3 \leq N \leq 10^6$

- $A_i \in \{0, 1\}$
- Tổng tất cả N trên mọi test ≤ 10^6

📤 Output

- Với mỗi test case, in ra **một số nguyên** là **số thành phần liên thông** trong đồ thị G tương ứng.

📌 Ví dụ

Input:

```
2
3
0 0 1
4
0 1 0 0
```

Output:

```
2
5
```

🧠 Tóm tắt đề bài

- Với mỗi test, tạo ra một đồ thị:
  - Mỗi đỉnh là một cặp (i, j), với i < j

- Nếu có bộ ba (i, j, k) sao cho $A_i + A_j + A_k = 1$, thì nối (i, j) — (j, k)

- Hỏi: đồ thị có bao nhiêu thành phần liên thông?

Bạn có muốn mình trình bày chiến lược tính nhanh số thành phần liên thông dựa trên tính chất đặc biệt của dãy nhị phân không?

🇻🇳 Bài K. Tổng bằng Một (Sum is One)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Cho một dãy nhị phân $A = (A_1, A_2, ..., A_N)$ chỉ gồm các số **0 và 1**.

Từ đó, xây dựng một **đồ thị vô hướng đơn giản** $G = (V, E)$ như sau:

🔷 Đỉnh V:

- Với mọi cặp chỉ số (i, j) thỏa mãn $1 \leq i < j \leq N$, tồn tại một đỉnh $(i, j) \in V$
- Tổng số đỉnh: $\frac{N(N-1)}{2}$

🔷 Cạnh E:

- Với mọi bộ ba chỉ số (i, j, k) thỏa mãn $1 \leq i < j < k \leq N$
  → Nếu $A_i + A_j + A_k = 1$, thì **tạo cạnh nối** giữa hai đỉnh:
  - (i, j) và (j, k)

🧠 Yêu cầu:

- Với mỗi test, tính **số lượng thành phần liên thông** trong đồ thị G

📥 Input

```
T
N₁
A₁₁ A₁₂ ... A₁N₁
...
Nₜ
Aₜ₁ Aₜ₂ ... AₜNₜ
```

- $1 \leq T \leq 10^5$
- $3 \leq N \leq 10^6$
- $A_i \in \{0, 1\}$
- Tổng tất cả N trong mọi test không vượt quá 10^6

📤 Output

- Với mỗi test, in ra một số nguyên: **số thành phần liên thông** trong đồ thị G tương ứng

---

📌 Ví dụ

Input:

```
4
5
1 0 0 1 0
5
1 1 1 1 1
12
0 0 1 1 1 0 0 0 1 0 1 0
20
0 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 1
```

Output:

```
4
10
13
58
```

🔍 Tóm tắt đề bài

- Mỗi đỉnh là một cặp (i, j)
- Mỗi cạnh nối từ $(i, j) \to (j, k)$ nếu $A_i + A_j + A_k = 1$
- Mỗi đỉnh có thể **nối với nhiều đỉnh khác**, tùy thuộc vào điều kiện đó

- **Đếm số lượng thành phần liên thông** trong đồ thị này

🧠 Ý tưởng giải nhanh

- Tổng số đỉnh $V = \frac{N(N-1)}{2}$ quá lớn ⇒ không thể lưu toàn bộ
- Nhận xét: chỉ có các **đỉnh nằm giữa các số 0 và 1**, khi cộng lại bằng 1 mới tạo ra cạnh
- Với mỗi chỉ số $j \in [2, N-1]$, xét các bộ ba (i, j, k) với $i < j < k$
- Ta chỉ quan tâm đến các bộ ba có tổng = 1 ⇒ tức là chỉ có **một số 1** trong ba số đó
- Với mỗi bộ ba thỏa $A_i + A_j + A_k = 1$, ta **nối (i,j) ↔ (j,k)** trong đồ thị

🔗 Cách làm hiệu quả:

- Duyệt j từ 2 đến N−1
- Với mỗi j, kiểm tra các (i = j−1) và (k = j+1) xem có tạo ra cạnh không
- Sử dụng DSU (Disjoint Set Union) để gộp các đỉnh (i,j), (j,k)

Bạn có muốn mình viết code chi tiết cho giải thuật này dùng DSU và tối ưu để chạy trong thời gian giới hạn không?

🇻🇳 Bài L. Đảo ngược của dãy dài 2 (Long Sequence Inversion 2)

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Bạn được cho:

- Một hoán vị P độ dài L (các số từ 0 đến $L-1$)
- L hoán vị $V_0, V_1, ..., V_{L-1}$, mỗi hoán vị có độ dài B

🧩 Cách xây dựng dãy A độ dài B^L:

Với mỗi chỉ số $n \in [0, B^L - 1)$:

1. Chuyển n thành biểu diễn cơ số B độ dài L
   → Gọi mảng chữ số đó là $N[0..L-1]$, với $N[i] \in [0, B-1]$
2. Tính:

$$A[n] = \sum_{i=0}^{L-1} V_i[N[i]] \cdot B^{P[i]}$$

🎯 Yêu cầu:

Tính **số lượng nghịch thế (inversion count)** trong dãy A, modulo 998244353

> *"Nghịch thế: số cặp chỉ số (i, j) với $i < j$ và $A[i] > A[j]$"*

📥 Input

```
L B
P[0] P[1] ... P[L - 1]
V0[0] ... V0[B - 1]
...
VL-1[0] ... VL-1[B - 1]
```

- $1 \leq L$
- $2 \leq B$
- $L \times (B + 1) \leq 5 \times 10^5$
- Tất cả P, V_i là hoán vị hợp lệ

📤 Output

- In ra **số lượng nghịch thế** của dãy A, modulo `998244353`

📌 Ví dụ

Input:

```
3 2
2 0 1
1 0
1 0
0 1
```

Output:

```
14
```

> *"Dãy A được tính là: $[5, 1, 4, 0, 7, 3, 6, 2]$*
> *Số lượng nghịch thế = 14"*

🧠 Tóm tắt đề bài

- Từ các hoán vị V_i và trọng số cơ số $B^{P[i]}$, bạn sinh ra một dãy A có độ dài B^L
- Sau đó đếm số lượng **cặp nghịch thế** trong dãy A
- Kích thước tối đa của dãy A có thể lên tới B^L, với $B = 2, L = 19 \rightarrow 2^{19} = 524288$

✅ Tối đa kích thước dãy A: khoảng 5×10^5 → có thể dùng:

- **merge sort** hoặc
- **Fenwick Tree (BIT)** hoặc
- **Segment Tree** để tính inversion count trong $O(N \log N)$

Bạn có muốn mình viết lời giải chi tiết và mã giả / code Python để tính dãy A và đếm số nghịch thế không?

🇻🇳 Bài M. Cây Cartesian

⏱️ **Giới hạn thời gian:** 3 giây
💾 **Giới hạn bộ nhớ:** 1024 MB

📘 Đề bài (dịch tiếng Việt)

Cho dãy hoán vị $A = (A_1, A_2, ..., A_N)$ của $(1, 2, ..., N)$.

Xét đoạn con từ chỉ số l đến r, định nghĩa **Cây Cartesian** $C(l, r)$ như sau:

1. Gốc là phần tử nhỏ nhất trong đoạn $[A_l, A_{l+1}, ..., A_r]$ — gọi chỉ số đó là m
2. Nếu $l < m$: con trái là $C(l, m-1)$; ngược lại không có con trái
3. Nếu $m < r$: con phải là $C(m+1, r)$; ngược lại không có con phải

---

🎯 Nhiệm vụ:

Cho Q truy vấn (l_i, r_i), hãy đếm có **bao nhiêu cây Cartesian khác nhau** trong tập $C(l_1, r_1), C(l_2, r_2), ..., C(l_Q, r_Q)$

📥 Input

```
N
A₁ A₂ ... Aₙ
Q
l₁ r₁
...
l_Q r_Q
```

- $1 \le N \le 4 \times 10^5$
- A là hoán vị của $(1, 2, ..., N)$
- $1 \le Q \le 4 \times 10^5$
- Mỗi (l_i, r_i) là đoạn con riêng biệt của A

📤 Output

- Một số nguyên: số lượng **khác nhau** của cây Cartesian ứng với các truy vấn

---

📌 Ví dụ

Input:

```
6
1 4 2 6 3 5
3
1 4
2 5
3 6
```

Output:

```
2
```

🧠 Tóm tắt đề bài

- Với mỗi đoạn $A[l..r]$, xây cây nhị phân dựa theo phần tử nhỏ nhất tại gốc, chia trái/phải đệ quy
- Hai cây được xem là **giống nhau** nếu **cấu trúc nhánh trái/phải giống nhau** (không xét giá trị đỉnh)
- Đề bài yêu cầu đếm bao nhiêu **dạng cây khác nhau** trong Q truy vấn

---

🔍 Ý tưởng giải nhanh

Tối ưu hóa bằng:

- Với mỗi đoạn (l, r), cây Cartesian được xác định bởi **cấu trúc xây dựng đệ quy**, không phụ thuộc vào giá trị cụ thể
- Vì A là hoán vị → mỗi đoạn có phần tử nhỏ nhất duy nhất
- Dùng **Segment Tree hoặc Sparse Table** để tìm nhanh chỉ số m nhỏ nhất trong đoạn $[l, r]$
- Sau đó, sử dụng **Hàm băm cây (tree hash)** đệ quy:
  - Duyệt cây Cartesian của đoạn $[l, r]$
  - Tính hàm băm cho mỗi node dựa vào:
    - Hash của con trái
    - Hash của con phải
    - Kết hợp lại qua $\text{hash} = \text{combine}(left\_hash, right\_hash)$

Chi tiết:

- Cache kết quả hàm băm cho từng đoạn (l, r)
- Dùng `set` hoặc `unordered_set` lưu tất cả hash các cây, rồi đếm số lượng khác nhau

---

Bạn có muốn mình trình bày mã Python hoặc C++ cho thuật toán dùng **Sparse Table + Tree Hashing** để giải nhanh các truy vấn không?