universal cup 3 stage 18 : Southest East

Created: 5/14/2025 20:59
Updated: 5/14/2025 21:14
Exported: 5/14/2025 21:15

🇻🇳 Bài A. Ngôi sao toàn năng

⏱️ **Giới hạn thời gian:** 1 giây
🧠 **Giới hạn bộ nhớ:** 512 MB

🟢 Đề bài

Shrek có **n nhà vệ sinh** trong đầm lầy của mình, được đánh số từ **1 đến n**. Các nhà vệ sinh này được nối với nhau bởi **n - 1 con đường hai chiều**, đảm bảo tạo thành **một cây** (tức là có thể đi từ bất kỳ nhà vệ sinh nào đến bất kỳ nhà vệ sinh nào khác bằng một số con đường nào đó, không có chu trình).

Trong **mỗi bước di chuyển**, Shrek có thể chọn **ba nhà vệ sinh phân biệt** x, y, z sao cho:

- Có đường nối giữa y với x và giữa y với z,
- Không có đường nối giữa x và z.

Khi đó, Shrek có thể **thay thế** đường nối giữa y và z bằng một đường nối giữa x và z.

🌟 Shrek muốn biến cây này thành **cây hình ngôi sao**: tức là tồn tại một đỉnh duy nhất nối trực tiếp đến tất cả các đỉnh còn lại.

Hãy giúp Shrek thực hiện **ít bước nhất** để đạt được điều đó.

🔢 Input

- Dòng đầu tiên: số nguyên n — số nhà vệ sinh.
- $n - 1$ dòng tiếp theo: mỗi dòng gồm 2 số nguyên u, v biểu diễn một con đường nối u và v.

Ràng buộc:

- $3 \leq n \leq 10^3$
- Luôn tồn tại cách giải với tối đa 10^6 bước.

🧾 Output

- Dòng đầu tiên: số nguyên m — số bước cần thực hiện.
- m dòng tiếp theo: mỗi dòng gồm 3 số nguyên x, y, z, mô tả một bước như đã nêu.

Nếu có nhiều cách, bạn có thể in ra bất kỳ một cách nào.

📌 Ví dụ

Input

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

Output

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

✍️ Tóm tắt ý tưởng bài toán

Bạn có một cây và cần biến nó thành **cây sao** (một đỉnh trung tâm nối đến tất cả đỉnh còn lại).
Phép biến đổi cho phép bạn "kéo" các cạnh về gần đỉnh trung tâm thông qua việc thay thế một cạnh $y - z$ bằng $x - z$, nếu x và y đều nối với y và $x - z$ chưa tồn tại.

→ Hãy chọn **một đỉnh gốc** (bất kỳ) làm trung tâm ngôi sao, sau đó lần lượt dùng phép biến đổi để nối các đỉnh khác về nó.
→ Mỗi đỉnh ban đầu chỉ có một cạnh, ta cần tạo thêm $n - 2$ cạnh mới để kết nối về đỉnh trung tâm (trừ 1 cạnh sẵn có), nên tối đa cần $n - 2$ bước.

🇻🇳 Bài B. Tổng đầu và cuối khác 0

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Shrek giao cho bạn một nhiệm vụ:

Hãy đếm số lượng **mảng** $[a_1, a_2, \ldots, a_n]$ gồm các số nguyên thoả mãn **các điều kiện sau**:

1. Với mọi $1 \leq i \leq n$, ta có:

$$\text{giá trị tuyệt đối } |a_i| \leq m$$

2. **Tồn tại một hoán vị** $[b_1, b_2, \ldots, b_n]$ của mảng a sao cho:
    - Tổng các **tiền tố** khác 0:

$$b_1 + b_2 + \cdots + b_k \neq 0 \quad \text{với mọi } 1 \leq k \leq n$$

    - Tổng các **hậu tố** khác 0:

$$b_k + b_{k+1} + \cdots + b_n \neq 0 \quad \text{với mọi } 1 \leq k \leq n$$

✳️ **Nói cách khác:** phải có **ít nhất một cách sắp xếp lại** các phần tử sao cho **mọi tổng đoạn đầu và đoạn cuối đều khác 0**.

🔢 Input

Một dòng duy nhất chứa ba số nguyên:

- n — độ dài của mảng
- m — giá trị tuyệt đối lớn nhất có thể của mỗi phần tử
- p — một số nguyên tố lớn, bạn cần in kết quả modulo p

📤 Output

- Một số nguyên duy nhất: là **số lượng mảng hợp lệ**, modulo p

---

📌 Ràng buộc

- $2 \leq n \leq 100$

- $1 \leq m \leq 100$
- $10^8 < p < 10^9$, và p là số nguyên tố

📘 Ví dụ

Input

```
2 1 998244353
```

Output

```
2
```

Giải thích:

Tất cả các mảng có phần tử trong đoạn $[-1, 1]$ gồm 9 mảng tổng cộng.
Nhưng chỉ có 2 mảng **[−1, −1]** và **[1, 1]** là có thể được sắp xếp sao cho:

- Mọi tổng đoạn đầu và đoạn cuối đều khác 0.

✍️ Tóm tắt yêu cầu bài toán

- Liệt kê tất cả các mảng có n phần tử, mỗi phần tử trong đoạn $[-m, m]$
- Với mỗi mảng, kiểm tra xem **có tồn tại một hoán vị** mà **mọi prefix sum và suffix sum đều khác 0**
- In ra tổng số mảng như vậy, modulo p

> *“Đây là một bài toán **liệt kê kết hợp với ràng buộc tổ hợp đặc biệt**, kiểm tra tổng prefix/suffix sau khi hoán vị. Bài này có thể giải bằng **quy hoạch động + sinh cấu hình + memoization** hoặc **backtracking có pruning mạnh**.”*

🇻🇳 Bài C. Mạng lưới Duloc *(bài tương tác)*

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Farquaad đang xây dựng một **mạng lưới trạm gác** ở vương quốc Duloc. Mạng lưới này là một **đồ thị vô hướng đơn** $G = (V, E)$, trong đó:

- Mỗi đỉnh tương ứng với một **trạm gác**
- Mỗi cạnh tương ứng với một **con đường nối hai trạm**

🧠 Farquaad muốn **đảm bảo rằng mạng này liên thông**, tức là có thể đi từ **bất kỳ trạm nào đến trạm nào khác** thông qua một số con đường.

Tuy nhiên, bạn **không được nhìn thấy toàn bộ đồ thị**, mà chỉ được phép **truy vấn** dưới hình thức sau:

❓ Truy vấn (Query)

Một truy vấn cho phép bạn chọn một tập con S các trạm và hỏi:

> *“Có bao nhiêu trạm **không thuộc** S được nối **trực tiếp** với **ít nhất một trạm trong** S?”*

Biểu thức toán học:

$$\text{query}(S) = |N(S) \setminus S|$$
$$\text{với } N(S) = \{x \in V : \exists y \in S, (x, y) \in E\}$$

📤 **Cú pháp khi truy vấn:**

```
? s
```

- Trong đó `s` là một chuỗi nhị phân dài n: `s[i] = 1` nghĩa là đỉnh $i + 1$ thuộc tập S, `0` thì không.

Sau mỗi truy vấn, bạn sẽ **đọc một số nguyên** — là kết quả trả về như mô tả ở trên.

🎯 Nhiệm vụ của bạn

Sử dụng các truy vấn hợp lệ (tối đa **3500** truy vấn) để xác định:

> *"**Đồ thị có liên thông hay không?**"*

Khi đã biết câu trả lời, in:

```
! x
```

- $x = 1$: nếu liên thông
- $x = 0$: nếu không liên thông

💡 **Lưu ý:** Đồ thị không thay đổi trong quá trình truy vấn.

🔢 Input

- Dòng đầu tiên: số lượng đỉnh n
- Sau đó bạn có thể thực hiện truy vấn như mô tả

📌 Ràng buộc

- $1 \leq n \leq 200$
- Số truy vấn tối đa: 3500

📘 Ví dụ

Ví dụ 1 (liên thông)

Input:

```
4
```

Interaction:

```
? 1100   → trả về: 2
? 0010   → trả về: 2
? 1001   → trả về: 2
! 1
```

Ví dụ 2 (không liên thông)

Input:

```
2
```

Interaction:

```
? 10   → 0
? 11   → 0
! 0
```

✍️ Tóm tắt

Bạn cần kiểm tra xem đồ thị có **liên thông** không, chỉ bằng cách hỏi:

> *"Với tập S, có bao nhiêu đỉnh ngoài S nối trực tiếp với S?"*

Đây là một bài toán tương tác, thường dùng các thuật toán như **tìm thành phần liên thông bằng Union-Find hoặc BFS mô phỏng**, nhưng phải **khéo léo khai thác truy vấn** để dần dựng được cấu trúc liên thông của đồ thị.

🇻🇳 Bài D. Donkey và Mèo đi hia

⏱️ **Giới hạn thời gian:** 1 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Ở vương quốc Far Far Away, Shrek có **n đống kẹo**, mỗi đống thứ i có a_i viên kẹo.

Shrek mê kẹo đến mức ngủ quên vì no. Khi đó, **Donkey** và **Puss in Boots** quyết định "ăn ké" và chơi một trò chơi:

🎮 Luật chơi:

- Donkey **đi trước**.
- Trong **một lượt**, Donkey **chọn một đống** và **ăn một số dương viên kẹo** từ đống đó.
- Sau đó, Puss in Boots đi, **thực hiện đúng n lượt như Donkey**, mỗi lượt cũng là chọn một đống và ăn một số dương viên kẹo.

➡️ Tức là mỗi lượt của Donkey = 1 bước, còn mỗi lượt của Puss = **n bước** liên tiếp.

👎 Ai không còn lượt hợp lệ để đi thì **thua**.

🔢 Input

- Dòng đầu: số nguyên n — số đống kẹo, cũng là số bước của Puss.
- Dòng sau: n số nguyên $a_1, a_2, \ldots, a_n$ — số viên kẹo mỗi đống.

📤 Output

- In ra `"Donkey"` nếu Donkey thắng, `"Puss in Boots"` nếu mèo thắng.

📘 Ví dụ

Input

```
2
2 2
```

Output

```
Puss in Boots
```

Input

```
4
0 47 0 0
```

Output

```
Donkey
```

🧠 Phân tích và ý tưởng

Đây là **biến thể của trò chơi Nim**, với quy tắc đặc biệt:

- Donkey đi 1 lượt mỗi vòng.
- Puss đi liên tục n lượt mỗi vòng.

💡 Ý tưởng then chốt:

- Tính **tổng số viên kẹo** $\text{sum} = \sum a_i$
- Nếu `sum % (n + 1) == 0`, thì **Puss in Boots** thắng.
- Ngược lại, **Donkey** thắng.

✅ Điều này đúng vì:

- Mỗi vòng Donkey + Puss = $1 + n = n + 1$ lượt.
- Nếu tổng số kẹo chia hết cho $n + 1$ thì người **đi sau** sẽ lấy viên cuối cùng → thắng.

✅ Tóm tắt

- Tính tổng tất cả số kẹo.
- Nếu `sum % (n + 1) == 0` → **Puss in Boots**
- Ngược lại → **Donkey**

📌 Code mẫu

```
n = int(input())
a = list(map(int, input().split()))
total = sum(a)
if total % (n + 1) == 0:
    print("Puss in Boots")
else:
    print("Donkey")
```

🇻🇳 Bài E. Shrooks

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Shrek muốn đặt **n quân xe (rook)** lên bàn cờ $n \times n$ sao cho thỏa mãn **các điều kiện sau**:

1. **Mỗi hàng và mỗi cột chỉ có đúng một quân xe**.
2. **Khoảng cách Manhattan giữa bất kỳ hai quân xe không vượt quá** n.

🐴 Tình huống éo le:

Donkey đã **đặt sẵn một số quân xe** trên bàn cờ (nhưng vẫn đảm bảo mỗi hàng và mỗi cột có **tối đa một** quân xe).
Bạn cần tính **số cách hợp lệ** để **đặt nốt các quân xe còn lại**, sao cho vẫn thỏa mãn 2 điều kiện trên.

In ra kết quả **modulo 998244353**.

🔢 Input

- Dòng đầu tiên chứa số nguyên t — số test.
- Với mỗi test:
  - Dòng đầu: số nguyên n — kích thước bàn cờ.
  - Dòng thứ hai: n số nguyên $a_1, a_2, ..., a_n$, mô tả tình trạng mỗi hàng:
    - Nếu $a_i = -1$: hàng i **chưa có** quân xe nào.
    - Nếu $a_i = j$: hàng i đã có một quân xe ở **cột** j.

📌 Ràng buộc

- $1 \leq t \leq 10^5$
- $2 \leq n \leq 2 \cdot 10^5$
- Tổng tất cả các n qua các test ≤ $2 \cdot 10^5$
- Với mọi i, $a_i = -1$ hoặc $1 \leq a_i \leq n$
- Không có hai hàng nào có quân xe đặt trùng cột

📤 Output

Với mỗi test, in ra một số nguyên — số cách đặt quân xe còn lại **modulo 998244353**

📘 Ví dụ

Input

```
6
2
1 2
3
-1 -1 -1
4
1 -1 -1 -1
5
1 -1 -1 -1 5
6
3 -1 -1 -1 -1 4
10
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Output

```
1
4
1
0
6
92
```

🧠 Giải thích thêm

- Trường hợp các quân xe đã đặt **đã vi phạm** khoảng cách Manhattan > n, thì **đáp án là 0**
- Nếu hợp lệ, cần đếm số hoán vị còn lại (mỗi hoán vị phải **tôn trọng** các cột đã dùng, và không tạo ra cặp nào có khoảng cách Manhattan > n)

🧾 Tóm tắt yêu cầu

- Đặt phần còn lại của một hoán vị hàng ↔ cột sao cho:
  - Không trùng hàng, không trùng cột
  - Mọi cặp quân xe đều có khoảng cách Manhattan ≤ n
- Có thể có một số quân xe đã đặt trước
- Trả về số cách hợp lệ để hoàn thiện bàn cờ, mod 998244353

🇻🇳 Bài F. Túi Phép Thuật

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Ở vùng đất Far Far Away, Donkey tìm thấy **n chiếc túi phép thuật**, mỗi túi chứa một số **đồ vật phép** có **giá trị ma thuật khác nhau** (là các số nguyên dương).

Donkey thách Shrek chơi trò:

> *""Giữ lại ít đồ vật nhất có thể, nhưng phải đảm bảo rằng: **mỗi cặp túi mà ban đầu là "tốt" thì vẫn phải "tốt" sau khi rút gọn**.""*

✅ Định nghĩa "tốt" (good):

Một cặp túi A, B là **tốt** nếu tồn tại:

- Một đồ vật trong A nhỏ hơn một đồ vật trong B, và
- Một đồ vật khác trong A lớn hơn một đồ vật trong B
  (có thể trùng nhau ở túi hoặc vật)

Nói cách khác:

> *"Trong A phải có ít nhất **một phần tử nhỏ hơn** và một phần tử **lớn hơn** phần tử trong B."*

🎯 Nhiệm vụ

- Giữ lại **ít nhất 1 đồ vật ở mỗi túi**
- Nhưng vẫn đảm bảo: nếu **ban đầu** cặp túi (i, j) là tốt thì **sau khi rút gọn** chúng **vẫn là tốt**
- Hãy tìm **số lượng ít nhất đồ vật còn lại** sau khi rút gọn theo yêu cầu trên

---

🔢 Input

- Dòng 1: số nguyên n — số túi
- n dòng tiếp theo: mỗi dòng có:
  - Một số nguyên k_i — số đồ vật trong túi thứ i
  - k_i số nguyên tiếp theo — giá trị phép của các vật trong túi

---

📌 Ràng buộc

- $2 \leq n \leq 2 \cdot 10^5$
- Tổng tất cả $k_i \leq 5 \cdot 10^5$
- $1 \leq a_{ij} \leq 10^9$
- Mỗi số phép là **duy nhất toàn bộ input**

📤 Output

- Một số nguyên — số lượng **ít nhất đồ vật phải giữ lại** để các cặp túi ban đầu “tốt” vẫn “tốt”

📘 Ví dụ

Input

```
4
3 4 7 10
2 1 9
4 11 2 8 14
3 6 12 13
```

Output

```
7
```

Giải thích:

- Ban đầu **tất cả cặp túi đều “tốt”**
- Sau khi rút gọn, mỗi túi giữ lại 1 hoặc 2 phần tử, vẫn đảm bảo các cặp túi "tốt" như ban đầu

---

✍️ Tóm tắt yêu cầu

- Một số túi chứa các số nguyên khác nhau (toàn cục duy nhất)
- Giảm số phần tử trong các túi nhưng:
  - Giữ lại ít nhất 1 phần tử mỗi túi

- Với **mọi cặp túi ban đầu là "tốt"**, thì sau khi rút gọn **vẫn phải "tốt"**

- Tìm tổng số phần tử còn lại **ít nhất** thỏa yêu cầu đó.

🇻🇳 Bài G. Bài Hát Đầm Lầy của Shrek

⏱️ **Giới hạn thời gian:** 4 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Shrek khám phá một bản giai điệu bí ẩn trong vùng đầm lầy, được ghi lại bằng **một dãy số nguyên**.

Shrek muốn giải mã **giai điệu dài nhất thỏa mãn điều kiện "Hòa Âm của Vương Quốc Duloc"**:

🎵 Quy tắc Hòa Âm Duloc:

Một **dãy con (subsequence)** được gọi là "hòa âm" nếu:

- Mỗi âm thanh (số) trong dãy con **phải có ít nhất một "hàng xóm" giống hệt nó** (ở vị trí liền kề trong dãy con).

Cụ thể, với dãy con $x_1, x_2, ..., x_k$, thì với mọi $1 \leq i \leq k$, phải thỏa mãn:

$$x_i = x_{i-1} \quad \text{hoặc} \quad x_i = x_{i+1}$$

(bỏ qua x_0 và x_{k+1} vì không tồn tại — chỉ xét trong phạm vi dãy con)

🔢 Input

- Dòng đầu tiên: số nguyên n — độ dài dãy nhạc
- Dòng thứ hai: n số nguyên $s_1, s_2, ..., s_n$ — giai điệu của bài hát

---

📌 Ràng buộc

- $1 \leq n \leq 10^6$
- $-10^9 \leq s_i \leq 10^9$

📤 Output

- Một số nguyên — độ dài của dãy con dài nhất thỏa mãn “hòa âm”
- Nếu không có dãy con nào hợp lệ → in `0`

📘 Ví dụ

Input

```
9
1 2 3 1 3 1 2 3 2
```

Output

```
5
```

Giải thích: Dãy con `[1, 1, 1, 2, 2]` là hợp lệ (mỗi số đều có ít nhất một hàng xóm giống nó).
Còn `[1, 2, 2, 1, 1]` là **không hợp lệ** vì số đầu tiên không có hàng xóm nào giống.

✍️ Tóm tắt yêu cầu

- Tìm **dãy con dài nhất** sao cho:
  - Mỗi phần tử trong đó có **hàng xóm liền kề giống nó** (phía trước hoặc phía sau trong dãy con)
- Dãy con **không cần liên tiếp** trên chuỗi gốc (subsequence, không phải substring)
- Trả về độ dài của dãy con này. Nếu không tồn tại → trả về `0`.

🇻🇳 Bài H. Shreckless

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Farquaad có một **bảng số nguyên** a kích thước $n \times m$. Ông ta muốn **mọi hàng** trong bảng đều được **sắp xếp không giảm** (nondecreasing).
Nhưng Shrek thì muốn phá bĩnh: **anh ấy được phép hoán vị các phần tử trong mỗi cột một cách tùy ý**.

👉 Shrek muốn đảm bảo rằng: **không có hàng nào là dãy không giảm** sau khi hoán đổi các cột.

🎯 Nhiệm vụ:

Với mỗi test, bạn cần xác định:

- Liệu **Shrek có thể hoán vị các cột** sao cho **không hàng nào** trong bảng là **dãy không giảm**?

📘 Định nghĩa

- Một dãy $a_1, a_2, ..., a_k$ là **không giảm** nếu $a_1 \leq a_2 \leq ... \leq a_k$

🔢 Input

- Dòng đầu: số lượng test t
- Với mỗi test:
  - Dòng đầu: hai số nguyên n, m — số hàng và cột
  - Tiếp theo n dòng, mỗi dòng m số nguyên — là nội dung bảng

---

📌 Ràng buộc

- $1 \leq t \leq 2 \cdot 10^4$
- $2 \leq n, m \leq 10^5$
- Tổng tất cả các phần tử trong toàn bộ input ≤ $2 \cdot 10^5$
- $1 \leq a_{i,j} \leq 10^9$

📤 Output

- Với mỗi test, in ra `YES` nếu **Shrek có thể làm cho tất cả hàng không còn không giảm**
- Ngược lại, in `NO`

📘 Ví dụ

Input

```
3
2 2
69 69
2024 42
3 3
1 1 1
1 1 1
2 2 2
3 4
1 1 1 1
1 1 1 1
2 2 2 2
```

Output

```
YES
NO
YES
```

✍️ Tóm tắt yêu cầu

- Bạn được **hoán vị từng cột độc lập** (nhưng không sửa giá trị nào)
- Sau đó, bạn cần **chống lại sự sắp xếp** của Farquaad:
  Không hàng nào trong bảng được sắp xếp theo thứ tự không giảm
- Trả lời `YES` nếu làm được, `NO` nếu không.

🇻🇳 Bài I. Donkey, Hãy Canh Gác

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Donkey, Shrek và Puss in Boots đột nhập vào nhà máy thuốc tiên của Bà Tiên để tìm thuốc giải cứu hôn nhân cho Shrek. Họ không tìm thấy sẵn, nên phải **tự pha chế** thuốc từ **các lọ thuốc có sẵn**.

Có **n lọ thuốc**, mỗi lọ có một **giá trị hiệu lực** a_i. Họ sẽ chọn ra **một tập con không rỗng** của các lọ để pha chế.

🍵 Thuốc tạo ra có 3 đặc tính:

- **Sweetness**: XOR của tất cả các a_i trong tập con
- **Aroma**: AND của tất cả các a_i trong tập con

- **Duration**: OR của tất cả các a_i trong tập con

✅ Một thuốc là "tốt" nếu:

$$\text{Sweetness} + \text{Aroma} = \text{Duration}$$

🎯 Nhiệm vụ

- Tìm **số lượng tập con không rỗng** (subsets) của dãy a sao cho tạo ra được **thuốc tốt**
- In ra kết quả **modulo** $10^9 + 7$

🔢 Input

- Dòng 1: số nguyên n — số lọ thuốc
- Dòng 2: n số nguyên $a_1, a_2, ..., a_n$ — hiệu lực của từng lọ

📌 Ràng buộc

- $1 \leq n \leq 10^6$
- $0 \leq a_i \leq 15000$ (giá trị nhỏ)

📤 Output

- Một số nguyên: số cách chọn tập con không rỗng tạo thành thuốc tốt, **modulo** $10^9 + 7$

📘 Ví dụ

Input

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

Output

```
11
```

Giải thích:
Một số tập hợp như `{1, 2}, {1, 3}, {4, 5}` đều thỏa mãn:

- Ví dụ `{1, 2, 4}`:
  - XOR = 7
  - AND = 0
  - OR = 7
    → $0 + 7 = 7 \Rightarrow$ thỏa mãn điều kiện

✍️ Tóm tắt yêu cầu

- Dãy gồm các số nhỏ (tối đa 15000), nhưng số lượng phần tử lên tới 10^6

- Cần đếm số **tập con không rỗng** có:

$$\text{XOR} + \text{AND} = \text{OR}$$

- Xuất kết quả modulo $10^9 + 7$

> *"Bài toán yêu cầu **kết hợp xử lý bit**, **các phép AND/OR/XOR**, cùng với **nhóm giá trị theo đặc trưng bit** để giảm độ phức tạp, vì duyệt toàn bộ 2^n là không khả thi."*

🇻🇳 Bài J. Làm Đầm Lầy Vĩ Đại Trở Lại

⏱️ **Giới hạn thời gian:** 1 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Xung quanh đầm lầy lớn nơi Shrek sống có **n sinh vật**, mỗi sinh vật sống trong một **ngôi nhà xếp thành vòng tròn**.

- Sinh vật i là hàng xóm của $i + 1$, và sinh vật thứ n là hàng xóm của sinh vật thứ 1.

Mỗi sinh vật có một **nhiệt độ ưa thích** t_i.
Shrek muốn **tất cả sinh vật cùng có một nhiệt độ ưa thích giống nhau**, để tạo ra một môi trường sống hài hòa.

🌙 Mỗi buổi tối, Shrek tổ chức một buổi họp gồm 3 sinh vật liên tiếp

Trong buổi họp:

- Chọn một trong ba sinh vật đó

- Cho phép nó thay đổi nhiệt độ ưa thích thành:
  - **giá trị nhỏ nhất**, hoặc
  - **giá trị lớn nhất** trong 3 sinh vật

🎯 Nhiệm vụ

Với mỗi sinh vật i, tính:

> *"Số buổi tối ít nhất cần thiết để **tất cả sinh vật** có cùng nhiệt độ với t_i ban đầu."*

🔢 Input

- Dòng 1: số nguyên n — số sinh vật (và số nhà)
- Dòng 2: n số nguyên $t_1, t_2, ..., t_n$ — nhiệt độ ưa thích ban đầu của các sinh vật, theo chiều kim đồng hồ

📌 Ràng buộc

- $3 \leq n \leq 10^5$
- $1 \leq t_i \leq 10^5$

📤 Output

- In ra n số nguyên — số buổi tối ít nhất để đưa toàn bộ sinh vật về đúng nhiệt độ t_i ban đầu của từng sinh vật

---

📘 Ví dụ

Input

```
6
4 7 47 4 77 47
```

Output

```
4 6 4 4 5 4
```

Giải thích:

- Để toàn bộ sinh vật có nhiệt độ **4**, cần ít nhất **4 buổi tối**.
- Để toàn bộ sinh vật có nhiệt độ **7**, cần **6 buổi tối**
- Tương tự với các nhiệt độ còn lại...

✍️ Tóm tắt yêu cầu

Với mỗi sinh vật, giả định **nhiệt độ ban đầu của nó** sẽ là **nhiệt độ mục tiêu**.
Tính số buổi tối **ít nhất** cần thực hiện để biến **toàn bộ dãy** thành một dãy gồm toàn t_i,
sử dụng quy tắc "chọn 3 sinh vật liên tiếp và một người được đổi thành min hoặc max của 3 người đó".

→ Cần tư duy theo hướng **lan truyền** (BFS) bắt đầu từ các vị trí đã có nhiệt độ mục tiêu.
→ Vì bài toán cần xử lý cho mọi t_i (tối đa 10^5), cần có **tối ưu theo giá trị** và dùng **multisource BFS**.

🇻🇳 Bài K. Donkey Lắm Lời

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Donkey không thể ngừng nói chuyện! Và khi Donkey **lặp lại** điều gì đó, **anh ấy nhân đôi từng ký tự trong một đoạn con** của câu nói ban đầu.

🎮 Có hai loại sự kiện:

1. **Sự kiện loại 1** `1 l r`:
   Donkey chọn đoạn từ chỉ số **l đến r** trong chuỗi hiện tại, và **nhân đôi mỗi ký tự** trong đoạn đó.
   Ví dụ: `"aabc"` → chọn đoạn 2 đến 3 là `"ab"` → sau khi nhân đôi → `"aaabbc"`

2. **Sự kiện loại 2** `2 i`:
   Shrek hỏi: **ký tự thứ i** trong chuỗi hiện tại là gì?

⚠️ Quan trọng:

- Chỉ số luôn **không vượt quá độ dài hiện tại**
- Độ dài chuỗi **không vượt quá** 10^{18} (rất lớn! ⇒ không thể lưu chuỗi thực tế)

🔢 Input

- Dòng 1: hai số nguyên n, q — độ dài ban đầu của chuỗi, và số sự kiện
- Dòng 2: chuỗi ban đầu s (gồm chữ thường, độ dài n)
- Sau đó là q dòng, mỗi dòng là một sự kiện:
  - `1 l r`: Donkey nhân đôi đoạn từ l đến r
  - `2 i`: Shrek hỏi ký tự thứ i trong chuỗi hiện tại

📌 Ràng buộc

- $1 \leq n, q \leq 2 \cdot 10^5$
- $1 \leq l \leq r \leq 10^{18}$
- $1 \leq i \leq 10^{18}$
- Tổng độ dài chuỗi sau mọi phép biến đổi luôn ≤ 10^{18}
- Mọi chỉ số luôn hợp lệ với độ dài hiện tại của chuỗi

📤 Output

- Với mỗi truy vấn loại 2, in ra **ký tự tại vị trí đó** trong chuỗi hiện tại

---

📘 Ví dụ

Input

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

Output

```
b
a
b
a
a
c
```

✍️ Tóm tắt yêu cầu

- Có một chuỗi ban đầu
- Donkey **lặp lại các đoạn con**, nhân đôi ký tự trong đoạn đó → chuỗi dài ra
- Không được lưu chuỗi thật vì quá dài (10^{18})
- Mỗi khi Shrek hỏi vị trí i, cần **truy ngược lại** để tìm nó **gốc từ đâu** mà ra

➡️ Phải xây dựng một **cấu trúc log biến đổi**, sau đó **truy ngược vị trí** i qua các lần biến đổi để truy xuất về **ký tự ban đầu**.

Dùng kỹ thuật **reverse simulation (mô phỏng ngược)** hoặc **segment tree ảo / interval tracking** để giải bài toán này trong giới hạn thời gian.

🇻🇳 Bài L. Ogre Sort

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Bạn được cho một **hoán vị** v gồm n số khác nhau từ 1 đến n.

Nhiệm vụ của bạn là **sắp xếp hoán vị** đó theo thứ tự tăng dần (1, 2, ..., n), bằng **một loại phép di chuyển đặc biệt**.

🔁 Phép di chuyển được phép:

Bạn được phép:

- Chọn phần tử tại vị trí x
- Chèn phần tử đó vào vị trí y
  ⇒ Tất cả các phần tử **từ vị trí nhỏ hơn đến lớn hơn giữa x và y** sẽ **bị dịch sang trái hoặc phải**

✅ Chi phí của thao tác đó là y, **vị trí chèn vào**.

📥 Input

- Dòng 1: số nguyên n

- Dòng 2: n số nguyên — hoán vị ban đầu

📤 Output

- Dòng 1: hai số nguyên:
  - **Tổng chi phí tối thiểu**
  - **Số lượng thao tác thực hiện**
- Sau đó mỗi dòng gồm 2 số x_k, y_k — thực hiện thao tác lấy phần tử ở x_k, chèn vào vị trí y_k

Bạn không cần tối ưu số thao tác, chỉ cần tối ưu **tổng chi phí**.

📘 Ví dụ

Input

```
4
1 2 4 3
```

Output

```
3 1
4 3
```

> *“Di chuyển 3 từ vị trí 4 → vị trí 3 → [1, 2, 3, 4], chi phí là 3”*

🧠 Tóm tắt yêu cầu

- Cho một hoán vị
- Được phép **di chuyển phần tử từ vị trí x → vị trí y**, chi phí = y
- Phải sắp xếp dãy theo thứ tự tăng dần
- Tìm cách **tối thiểu hóa tổng chi phí**

💡 Gợi ý hướng giải

- Duy trì **chuỗi con dài nhất đã đúng thứ tự** (longest prefix của mảng đã sorted)
- Với các phần tử **chưa đúng vị trí**, ta **đẩy về đúng vị trí** bằng thao tác tối ưu (chọn vị trí chèn y sao cho chi phí nhỏ nhất)
- Ta có thể dùng **cây Fenwick / segment tree ảo** để tracking vị trí hiệu quả sau nhiều lần chèn-dịch

> *"Tuy nhiên, yêu cầu chỉ cần **in ra lời giải hợp lệ với chi phí tối thiểu**, không cần ngắn nhất ⇒ ta có thể dùng **quy hoạch động + theo dõi LIS** hoặc **Greedy + theo dõi vị trí mục tiêu từng phần tử**."*

🇻🇳 Bài M. Nhiệm Vụ Trên Bãi Cỏ Phép Thuật

⏱️ **Giới hạn thời gian:** 2 giây
💾 **Giới hạn bộ nhớ:** 512 MB

🟩 Đề bài

Shrek và Donkey đang ở trong **khu rừng phép thuật**, nơi các **bãi cỏ** (đỉnh) được nối với nhau bởi các **đường mòn phép thuật** (cạnh).
Mỗi đường mòn có một **độ kháng phép** a_i.

Cấu trúc các đường mòn là **một cây** — tức là có đúng **một đường đi đơn** giữa bất kỳ cặp bãi cỏ nào.

🔪 Đường đi khó nhất (đường kính của cây) được định nghĩa là:

> *"Tổng kháng phép lớn nhất trên bất kỳ đường đi đơn nào giữa hai bãi cỏ."*

🌟 Shrek có thể thêm ma thuật vào các đường mòn để giảm đường kính cây.

Bạn được cho:

- w: số đơn vị ma thuật có thể thêm
- Với mỗi cạnh ban đầu a_i, bạn phải tăng lên một giá trị $b_i \geq a_i$
- Tổng các $b_i = \sum a_i + w$
- Mỗi b_i là **số nguyên**

🎯 Mục tiêu:

> *"**Phân bổ số ma thuật** w vào các cạnh sao cho **đường kính nhỏ nhất có thể**"*

🔢 Input

- Dòng đầu: n, w — số đỉnh (bãi cỏ), và lượng ma thuật
- $n - 1$ dòng tiếp theo: mỗi dòng gồm u_i, v_i, a_i — mô tả cạnh nối đỉnh u_i và v_i có kháng phép a_i

---

📌 Ràng buộc

- $2 \leq n \leq 2 \cdot 10^5$
- $1 \leq w \leq 10^{12}$
- $1 \leq a_i \leq 10^7$

📤 Output

- In ra một số nguyên — **đường kính nhỏ nhất có thể** sau khi phân bổ w đơn vị phép thuật

---

📘 Ví dụ

Input

```
5 7
1 2 2
1 3 4
1 4 5
3 5 2
```

Output

```
14
```

Giải thích:

- Tổng kháng phép ban đầu = 13
- Tổng kháng phép sau thêm = 13 + 7 = 20
- Ta phải phân bổ sao cho **đường đi dài nhất (đường kính)** là nhỏ nhất

---

✍️ Tóm tắt yêu cầu

- Cho một **cây**, mỗi cạnh có trọng số a_i
- Bạn được tăng trọng số mỗi cạnh thành $b_i \geq a_i$ sao cho tổng $b_i = \sum a_i + w$
- Tìm cách phân bổ **số dư** w vào các cạnh để **giảm đường kính cây nhỏ nhất có thể**
- In ra **đường kính tối thiểu có thể đạt được** sau khi cộng w

---

💡 Gợi ý hướng giải

- Đường kính cây phụ thuộc vào **đường đi dài nhất**
- Bài toán này có thể giải bằng:
  - **Binary search trên đáp án đường kính** D

    - Với mỗi giá trị D thử kiểm tra xem có thể **phân bổ w đơn vị** để mọi đường đơn không vượt quá D

- Dùng **DFS kết hợp phân bổ phép thuật thông minh** để tính số ma thuật tối thiểu cần thiết nếu bị giới hạn đường kính D

> *“Đây là một bài toán **tối ưu hóa trên cây** điển hình, sử dụng **kỹ thuật tìm đường kính + nhị phân kết quả + greedy DFS**.”*