


# universal cup 3 stage 25 : Hangzhou


Created: 4/30/2025 14:59

Updated: 4/30/2025 15:09

Exported: 4/30/2025 15:08

## Bài A. AUS

 **Đại học Pigeland** sẽ tổ chức cuộc thi **ACPC 2224**. Không như mọi năm, năm nay Pigeland sẽ cử các **đội chính thức** tham gia thi đấu, mỗi đội có **3 máy tính**. Tuy nhiên, **huấn luyện viên Pig-head** vẫn **ngghi ngờ cơ hội giành huy chương vàng** của đội mình nên quyết định **lấy trộm đề** từ nhóm ra đề AUS với lý do cần **tải dữ liệu lên hệ thống chấm thi**.

 Để chống gian lận, nhóm AUS mã hóa đề bài bằng một **hàm mã hóa đặc biệt**. Cụ thể:

- Mỗi đề bài là một **xâu chữ thường tiếng Anh**.
- Hàm mã hóa là một ánh xạ  **$f(x)$**  từ chữ thường sang chữ thường.
- Nếu đề bài là  $S = s_1 s_2 \dots s_n$ , thì sau mã hóa thành  $F(S) = f(s_1) f(s_2) \dots f(s_n)$ .

 **Yêu cầu:** Phải thiết kế hàm mã hóa  **$f$**  **sao cho:**

- $F(S_1) = F(S_2)$  (2 đề mã hóa ra giống nhau)
- $F(S_1) \neq F(S_3)$  (khác đề thứ 3)

 Với mỗi bộ 3 đề  $S_1, S_2, S_3$ , bạn phải **kiểm tra** xem **có tồn tại hàm  $f$**  thỏa mãn các điều kiện trên không.

---

## Input

- Số bộ test  $T$  ( $1 \leq T \leq 10^4$ )

- Với mỗi test:
    - Dòng 1: xâu  $S_1$
    - Dòng 2: xâu  $S_2$
    - Dòng 3: xâu  $S_3$   
( $1 \leq$  độ dài mỗi xâu  $\leq 1000$ , tổng độ dài tất cả test  $\leq 30000$ )
- 

## Output

- Với mỗi test, in ra:
    - ``YES`` nếu tồn tại hàm mã hóa phù hợp
    - ``NO`` nếu không
- 

## Ví dụ

Input	Output
4	
abab	
cdcd	
abce	
abab	
cdcd	
abcd	
abab	

Input	Output
cdcd	
abc	
x	
yz	
def	
	YES
	NO
	YES
	NO

## Tóm tắt đề bài

- Cho 3 xâu  $S_1, S_2, S_3$ .
- Bạn cần kiểm tra xem có **hàm ánh xạ chữ cái sang chữ cái** nào sao cho:
  - $F(S_1) = F(S_2)$
  - $F(S_1) \neq F(S_3)$
- In ``YES`` nếu tồn tại hàm như vậy, ``NO`` nếu không.

## Bài B. Barkley III

 Ở **Pigeland**, có  **$n$  chú heo**, tất cả đều **giỏi lập trình thi đấu**. Mỗi chú heo thứ  $i$  có một mức **xếp hạng**  $a_i$ .

🤖 Nếu một đội gồm  $k$  chú heo  $p_1, p_2, \dots, p_k$  được thành lập, **điểm xếp hạng của đội** đó sẽ là:

$$a_{p_1} \& a_{p_2} \& a_{p_3} \& \dots \& a_{p_k}$$

trong đó  $\&$  là **phép toán AND bit**.

🎯 Có một số cuộc thi lập trình sắp diễn ra, mỗi cuộc thi chỉ cho phép một đội tham gia. Với **cuộc thi thứ  $i$ , chỉ những chú heo từ  $l_i$  đến  $r_i$**  mới có thể tham gia.

Tuy nhiên, vì thiếu kinh phí, **mỗi lần thi phải loại đúng 1 chú heo trong đoạn  $[l_i, r_i]$** , và phần còn lại sẽ tạo thành đội thi.

🎯 Nhiệm vụ là chọn chú heo bị loại sao cho **điểm đội cao nhất** (tức là **giá trị AND của  $n-1$  con còn lại là lớn nhất**).

---

🔧 Ngoài ra, điểm số của các chú heo có thể thay đổi qua luyện tập và thi đấu. Bạn được yêu cầu xử lý chuỗi các thao tác bao gồm 3 loại:

🔧 Các loại thao tác:

1. `1 l r x`: cập nhật đoạn từ  $l$  đến  $r$ :

$$\text{với mọi } i \in [l, r], \quad a_i := a_i \& x$$

2. `2 s x`: gán lại điểm cho chú heo số  $s$ :

$$a_s := x$$

3. `3 l r`: truy vấn điểm đội cao nhất nếu chọn tất cả chú heo từ  $l$  đến  $r$  và **loại đúng 1 con**.

---

## Input

- Dòng đầu: 2 số nguyên  $n, q$  — số chú heo và số thao tác ( $2 \leq n \leq 10^6, 1 \leq q \leq 10^6$ )
  - Dòng 2:  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i < 2^{63}$ )
  - Mỗi dòng tiếp theo là một thao tác thuộc 1 trong 3 loại như trên.
- 

## Output

- Với mỗi thao tác loại 3, in ra **điểm đội tối đa** đạt được khi loại đúng 1 chú heo trong đoạn.
- 

## Ví dụ


Input	Output
5 9	
7 7 7 6 7	
3 1 5	7
2 1 3	
3 1 5	6
3 1 3	7
1 1 2 3	
3 1 3	3

Input	Output
2 2 8	
3 1 3	3
3 1 2	8


## Tóm tắt đề bài

- Có  $n$  con heo, mỗi con có điểm.
- Có 3 loại thao tác:
  1. Cập nhật đoạn bằng toán tử `AND`
  2. Gán lại giá trị cho 1 con heo
  3. Truy vấn đoạn: loại đúng 1 con để AND những con còn lại sao cho kết quả là **lớn nhất**.
- Với mỗi truy vấn loại 3, in ra kết quả.

## Bài C. Catch the Star

 **BaoBao** mua kính thiên văn để quan sát một ngôi sao vào ban đêm. Ngôi sao này được mô tả bằng một **đa giác lồi**  $S$ .

○ Tuy nhiên, có  $n$  **mặt trăng**, mỗi mặt trăng là một **đa giác lồi**  $M_i$  có thể **che khuất tầm nhìn** của BaoBao.

 BaoBao có thể **đặt kính thiên văn ở bất kỳ vị trí nào trên trục hoành** (trục  $x$ ) từ điểm  $(l, 0)$  đến  $(r, 0)$ , **ngoại trừ chính xác tại hai đầu  $l$  và  $r$** .

## Mục tiêu

Tìm **tổng chiều dài các đoạn trên trục hoành mà BaoBao có thể đặt kính thiên văn để quan sát được ngôi sao  $S$  mà không bị bất kỳ mặt trăng nào che khuất.**

*“Một **tâm nhìn không bị che khuất** nghĩa là:*

*từ điểm đặt kính, bất kỳ đoạn thẳng nào đến một điểm thuộc hoặc nằm bên trong ngôi sao  $S$ , **không được cắt qua bên trong mặt trăng  $M_i$**  (được chạm biên).”*

## Input

- Số bộ test  $T$  ( $1 \leq T \leq 25,000$ )
- Mỗi test:
  - Dòng 1: 3 số nguyên  $n, l, r$ : số mặt trăng, và khoảng được đặt kính thiên văn ( $1 \leq n \leq 10^4, -10^9 \leq l < r \leq 10^9$ )
  - Dòng 2: thông tin đa giác ngôi sao:
    - Số đỉnh  $k_0$ , tiếp theo là  $2k_0$  số: tọa độ  $(x_{0,j}, y_{0,j})$  theo thứ tự ngược chiều kim đồng hồ
  - $n$  dòng tiếp theo: mỗi dòng là đa giác mặt trăng thứ  $i$ , có:
    - Số đỉnh  $k_i$ , tiếp theo là  $2k_i$  số: tọa độ các đỉnh

## Ràng buộc đảm bảo:

- $S$  và các  $M_i$  là đa giác lồi
- $S$ , các mặt trăng  $M_i$ , và đoạn  $(l, 0) \rightarrow (r, 0)$  không giao nhau
- Các mặt trăng  $M_i$  có thể giao nhau với nhau
- Tổng số đỉnh của tất cả đa giác qua mọi test  $\leq 10^6$
- Tổng số đa giác (gồm cả ngôi sao và các mặt trăng) qua tất cả test  $\leq 5 \times 10^4$

## Output

Với mỗi test, in ra **tổng chiều dài** các đoạn mà BaoBao có thể đặt kính thiên văn để quan sát được  $S$  mà không bị chắn tầm nhìn. Nếu không có vị trí nào hợp lệ, in ra:

-1

Kết quả được chấp nhận nếu **sai số tuyệt đối hoặc tương đối**  $\leq 10^{-9}$ .

## Ví dụ

Input	Output
2	
4 -8 8	
3 -7 7 -6 8 -7 8	
3 -9 -2 -7 3 -9 -1	
3 -2 3 0 2 -4 5	
3 5 1 5 3 4 2	
3 1 -1 2 -2 3 -1	
5 -8 8	
5 -14 -3 -10 -2 -9 2 -10 4 -12 5	
3 -16 0 -15 0 -15 1	
3 -15 6 -9 5 -15 7	
3 -10 5 -9 5 -10 6	




Input	Output
3 -7 3 -3 2 -8 4	
3 -6 -1 -6 -2 -5 -1	
	9.404761904761905
	6.000000000000000

## Tóm tắt đề bài

- Bạn được cho một đa giác lồi ngôi sao  $S$ , và  $n$  đa giác lồi là các mặt trăng  $M_i$
- Bạn cần tính tổng độ dài đoạn trên trục  $x$  từ  $l$  đến  $r$  (trừ điểm đầu cuối) mà từ đó có thể nhìn thấy  $S$  **mà không có đoạn nào từ điểm nhìn đến  $S$  cắt qua mặt trăng  $M_i$**
- Nếu không thể đặt ở đâu, in `-1`

Bạn muốn mình giải thích cách tiếp cận thuật toán hay xây dựng chương trình không?

## Bài D. Dividing Sequence

 **Alice** có một dãy số nguyên  $A$  được hàng xóm đưa cho. Do không thích dãy dài, Alice muốn chia nó thành **hai dãy con** (có thể rỗng)  $B$  và  $C$  sao cho:

### Ràng buộc:

- $B$  và  $C$  là **dãy con** (subsequence) của  $A$ .
- Mỗi phần tử trong  $A$  phải nằm **chính xác** trong **hoặc**  $B$  hoặc  $C$ .
- $B \leq C$  theo **thứ tự từ điển** (lexicographical order).

*“Ghi nhớ:*

*Dãy  $P = (p_1, \dots, p_u)$  nhỏ hơn dãy  $Q = (q_1, \dots, q_v)$  nếu:”*

- “ $u < v$  **và**  $P$  là prefix của  $Q$ , hoặc”*
- “tồn tại chỉ số  $k$  sao cho  $p_i = q_i$  với mọi  $i < k$  và  $p_k < q_k$ ”*

🎯 Alice muốn **chia sao cho**  $C$  (dãy còn lại) là **nhỏ nhất về mặt từ điển**.

---

## Input

- Dòng 1: số lượng test  $T$  ( $1 \leq T \leq 10^4$ )
- Với mỗi test:
  - Dòng 1: số nguyên  $n$  ( $1 \leq n \leq 5000$ ) — độ dài dãy  $A$
  - Dòng 2:  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^5$ )

 Tổng  $n$  của tất cả test  $\leq 10^4$

---

## Output

Với mỗi test, in:

- Dòng 1: độ dài  $m$  của dãy  $C$
- Dòng 2:  $m$  số nguyên là dãy  $C$  (nhỏ nhất từ điển có thể)

---

## Ví dụ


Input	Output
5	
5	
3 1 2 3 2	1
	3
3	3
1 1 2	1 1 2
3	2
3 3 3	3 3
5	3
1 3 1 3 1	1 3 1
5	4
2 2 1 3 3	2 1 3 3


## Tóm tắt đề bài

- Cho dãy  $A$ , chia thành 2 dãy con  $B$  và  $C$
- Mỗi phần tử chỉ thuộc hoặc  $B$  hoặc  $C$
- Ràng buộc:  $B \leq C$  (theo từ điển)
- Mục tiêu: chọn cách chia sao cho  $C$  **nhỏ nhất từ điển**
- Xuất ra độ dài và nội dung của  $C$

Bạn có muốn mình giải thích thuật toán hoặc viết code C++ / Python cho bài này không?

## Bài E. Elevator II

 Có một tòa nhà **cao 10<sup>9</sup> tầng** nhưng chỉ có **một thang máy**. Ban đầu, thang máy đang ở tầng  $f$ .

 Có  $n$  người đang **chờ thang máy**. Mỗi người thứ  $i$  đang ở tầng  $l_i$  và muốn di chuyển đến tầng  $r_i$ , với  $l_i < r_i$ .

 **Thang máy chỉ chở được 1 người mỗi lần.**

---

### Quy tắc tính năng lượng:

- Di chuyển **lên 1 tầng** mất **1 đơn vị năng lượng**.
- Di chuyển **xuống** thì **không tốn năng lượng**.
- Tổng năng lượng để đưa người  $a_i$  (người thứ  $a_i$  được phục vụ thứ  $i$ ) là:

$$\text{cost} = \sum_{i=1}^n (\max(l_{a_i} - r_{a_{i-1}}, 0) + (r_{a_i} - l_{a_i}))$$

với  $r_{a_0} = f$  (tầng ban đầu của thang máy).

---

### Yêu cầu

- Tìm một **thứ tự phục vụ người** sao cho tổng năng lượng **tối thiểu**
- Xuất ra:
  - Dòng 1: tổng năng lượng nhỏ nhất

- Dòng 2: một hoán vị của  $1..n$  là thứ tự phục vụ người

---

## Input

- Số test  $T$  ( $1 \leq T \leq 10^4$ )
- Mỗi test:
  - Dòng 1:  $n, f$  ( $1 \leq n \leq 10^5, 1 \leq f \leq 10^9$ )
  - $n$  dòng sau: mỗi dòng là  $l_i, r_i$  ( $1 \leq l_i < r_i \leq 10^9$ )

**Tổng  $n$  qua tất cả test  $\leq 3 \times 10^5$**

---

## Output

- Mỗi test:
  - Dòng 1: tổng năng lượng tối thiểu
  - Dòng 2: thứ tự phục vụ người

## Ví dụ

Input	Output
2	
4 2	
3 6	


Input	Output
1 3	
2 7	
5 6	
2 5	
2 4	
6 8	
	11
	2 1 4 3
	5
	2 1

## Tóm tắt đề bài

- Có  $n$  người, mỗi người muốn đi từ tầng  $l_i$  đến tầng  $r_i$
- Thang máy ban đầu ở tầng  $f$ , chỉ chở 1 người mỗi lần
- Tìm thứ tự phục vụ sao cho tổng năng lượng di chuyển là **nhỏ nhất**
- Xuất ra năng lượng tối thiểu và thứ tự người được phục vụ

Bạn có muốn mình giải thích chiến lược tối ưu hóa thuật toán hay viết code minh họa không?

## Bài F. Fuzzy Ranking

 Ở **Pigeland**, có  $n$  trường đại học. Năm nay có  $k$  bảng xếp hạng các trường (ranking), mỗi bảng là một **hoán vị** của các số từ 1 đến  $n$ , đại diện cho thứ hạng các trường (số đứng càng sớm thì trường xếp hạng càng cao).

---

### Định nghĩa “ưu thế” (superior) của sinh viên Supigar:

Một trường  $x$  **ưu thế hơn**  $y$  nếu:

1.  $x$  xếp **cao hơn**  $y$  trong **ít nhất một bảng xếp hạng**, hoặc
2. Có một trường  $z \neq x, z \neq y$ , sao cho  $x$  xếp **cao hơn**  $z$  trong một bảng và  $z$  **ưu thế hơn**  $y$

*“ $\Rightarrow$  Mỗi quan hệ “ưu thế” là **đệ quy** và **truy vết qua tất cả bảng xếp hạng**”*

---

### Định nghĩa fuzzy pair:

Cặp trường  $(x, y)$  là **fuzzy** nếu:

- $x < y$ , và
  - $x$  **ưu thế hơn**  $y$ , đồng thời  $y$  cũng **ưu thế hơn**  $x$
- 

### Yêu cầu của từng truy vấn:

Cho bảng xếp hạng thứ  $i$ , và một đoạn từ vị trí  $l$  đến  $r$  trong bảng đó (nghĩa là một danh sách các trường), **đếm số lượng các cặp fuzzy** trong danh sách này.

---

## Input

- Dòng 1: số test  $T$  ( $1 \leq T \leq 200,000$ )
- Mỗi test:
  - Dòng 1:  $n, k, q$  — số trường, số bảng xếp hạng, số truy vấn
  - $k$  dòng tiếp theo: mỗi dòng là một hoán vị độ dài  $n$
  - $q$  dòng tiếp theo: mỗi dòng là **truy vấn mã hóa**:
    - $id'_i, l'_i, r'_i$

---

## Giải mã truy vấn:

Dùng biến  $v_{i-1}$  là kết quả của truy vấn trước (ban đầu  $v_0 = 0$ ):

$$\begin{aligned} id_i &= ((id'_i + v_{i-1}) \bmod k) + 1 \\ l_i &= ((l'_i + v_{i-1}) \bmod n) + 1 \\ r_i &= ((r'_i + v_{i-1}) \bmod n) + 1 \end{aligned}$$

“ $\Rightarrow$  Phải giải từng truy vấn **tuần tự**, vì truy vấn sau phụ thuộc vào kết quả trước”

---

## Output

Với mỗi truy vấn, in một dòng là **số lượng cặp fuzzy** trong danh sách trường được xét.



## Ví dụ

### Input:

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

### Output:

```
3
10
1
1
2
```

## Tóm tắt đề bài

- Có  $n$  trường,  $k$  bảng xếp hạng (mỗi bảng là hoán vị)
- Một trường được coi là "ưu thế hơn" trường khác nếu nó:
  - xếp cao hơn trong 1 bảng, hoặc
  - xếp cao hơn một trường trung gian, và trường đó ưu thế hơn trường kia

- Mỗi truy vấn cho một đoạn trong bảng thứ  $i$ , yêu cầu **đếm số cặp fuzzy**

Bạn có muốn mình trình bày ý tưởng giải bài toán này bằng thuật toán hoặc cấu trúc dữ liệu cụ thể không?

## Bài G. Gathering Mushrooms

 Trong khu rừng sương mù, **BaoBao** đi hái nấm tại  $n$  **địa điểm**. Mỗi địa điểm:

- Có **một loại nấm**  $t_i$  (vô hạn cây loại đó).
- Có **một tấm biển chỉ dẫn** đến một địa điểm khác  $a_i$  (có thể là chính nó).

### Luật di chuyển và hái nấm:

- BaoBao bắt đầu từ một vị trí  $s$  (với  $1 \leq s \leq n$ ) với **giỏ trống**.
- Mỗi lần đi vào 1 vị trí:
  - Hái **1 cây nấm loại**  $t_c$  (kể cả khi đã từng đến rồi).
  - Di chuyển đến vị trí mà tấm biển ở đó chỉ đến ( $a_c$ ).

---

## Mục tiêu

Cho số nguyên  $k$ , với mỗi vị trí bắt đầu  $s = 1..n$ , xác định **loại nấm đầu tiên** được hái **đủ  $k$  lần**.

Sau đó tính:

$$\sum_{i=1}^n (i \times v_i)$$

với  $v_i$  là **loại nấm đầu tiên được hái ít nhất  $k$  lần** khi bắt đầu tại vị trí  $i$ .

---

## Input

- Dòng 1: số lượng test  $T$  ( $1 \leq T \leq 10^4$ )
  - Mỗi test:
    - Dòng 1:  $n, k$  ( $1 \leq n \leq 2 \times 10^5, 1 \leq k \leq 10^9$ )
    - Dòng 2:  $t_1, t_2, \dots, t_n$  — loại nấm tại mỗi địa điểm
    - Dòng 3:  $a_1, a_2, \dots, a_n$  — chỉ dẫn biến từ mỗi địa điểm
  - Tổng  $n$  trên tất cả test  $\leq 2 \times 10^5$
- 

## Output

- Mỗi test: in 1 dòng là tổng  $\sum i \times v_i$
- 

## Ví dụ

**Input:**

```
3
5 3
2 2 1 3 3
2 5 1 2 4
5 4
2 2 1 3 3
2 5 1 2 4
3 10
1 2 3
1 3 2
```

## Output:

41  
45  
14


## Tóm tắt đề bài

- Bạn đi theo **chu trình định hướng** từ điểm bắt đầu  $s$ , mỗi bước:
  - Hái nấm ở vị trí đang đứng
  - Đi tiếp theo biển chỉ
- Cần tìm **loại nấm đầu tiên** đạt  $k$  lần hái
- Với mọi  $s = 1..n$ , tìm loại đó, sau đó tính tổng:

$$\sum_{i=1}^n (i \times \text{loại nấm đạt } k \text{ lần sớm nhất})$$

Bạn có muốn mình triển khai chiến lược thuật toán tối ưu hoặc viết code minh họa không?

## Bài H. Heavy-light Decomposition

 **Heavy-Light Decomposition (HLD)** là kỹ thuật thường dùng để chia nhỏ cây nhằm xử lý các chuỗi đỉnh hiệu quả. Trong bài này, bạn cần:

*“Tái dựng cây gốc từ các chuỗi heavy chain được cung cấp sau khi thực hiện HLD.”*

## Định nghĩa trong HLD:

- Mỗi **đỉnh không phải gốc** được phân loại là **heavy** hoặc **light**
- Mỗi **đỉnh không phải lá** có đúng 1 con được chọn làm **heavy**, các con còn lại là **light**
- Với đỉnh  $v$ , được chọn làm **heavy chỉ khi**:

$$s_v \geq s_w \quad \text{với mọi con } w \text{ của cha của } v$$

trong đó  $s_v$  là kích thước cây con gốc  $v$

---

## **Chuỗi heavy chain:**

Là một dãy đỉnh thỏa mãn:

- $x_1$  là **gốc** hoặc là một **light node**
- Các đỉnh tiếp theo là con của đỉnh trước và là **heavy node**
- Đỉnh cuối là **lá**

---

## **Yêu cầu đề bài:**

Cho số lượng đỉnh  $n$ , và danh sách  $k$  chuỗi heavy chain (dạng  $li \rightarrow ri$ , tức là chuỗi  $li, li+1, \dots, ri$ )

Bạn cần **xây dựng cây gốc** (một mảng cha  $p_1, \dots, p_n$ ) sao cho:

- Mỗi chuỗi  $li \rightarrow ri$  là **một heavy chain hợp lệ** trong cây
- Mỗi đỉnh nằm trong đúng 1 heavy chain
- Nếu không thể, in `IMPOSSIBLE`

---

## Input

- Số test  $T$  ( $1 \leq T \leq 10^5$ )
- Mỗi test:
  - Dòng đầu:  $n, k$  ( $1 \leq k \leq n$ )
  - $k$  dòng tiếp theo: mỗi dòng là  $l_i, r_i$  — heavy chain  $[l_i, \dots, r_i]$

Tổng  $n$  của tất cả test  $\leq 2 \times 10^5$

---

## Output

- Với mỗi test:
    - Nếu hợp lệ: in ra  $n$  số nguyên,  $p_i$  là cha của đỉnh  $i$  ( $p_i = 0$  nếu là gốc)
    - Nếu không hợp lệ: in `IMPOSSIBLE`
- 

## Ví dụ

**Input:**

```
3
12 5
1 5
9 11
7 8
6 6
12 12
4 3
```

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

### Output:

```
0 1 2 3 4 3 2 7 1 9 10 10
2 0 2 2
IMPOSSIBLE
```



## Tóm tắt đề bài

- Bạn có  $k$  chuỗi heavy chain (dạng  $l_i \rightarrow r_i$ )
- Hãy xây dựng một cây gốc  $n$  đỉnh sao cho các chuỗi này là **kết quả của một HLD hợp lệ**
- Mỗi chuỗi:
  - Phần tử đầu là gốc hoặc light
  - Các phần tử sau là heavy con của phần tử trước
- Nếu xây được, in cây (mảng cha). Nếu không, in `IMPOSSIBLE`

Bạn có muốn mình tiếp tục bằng cách **giải thích chiến lược thuật toán** hoặc **viết code C++/Python** để kiểm tra và dựng lại cây không?



## ❏ Problem 1. Identify Chord (Interactive)

---

### 🧩 Mô tả bài toán:

Bạn có một **đồ thị vô hướng tạo thành một chu trình đơn gồm  $n$  đỉnh** (tức là một chuỗi vòng:  $1 - 2 - 3 - \dots - n - 1$ ).

Grammy thấy đồ thị này **nhàm chán**, nên **thêm một cạnh (gọi là chord)** nối giữa **2 đỉnh không kề nhau**.

→ Kết quả là đồ thị có:

- $n$  đỉnh
  - $n + 1$  cạnh (1 chu trình + 1 chord)
  - Có đúng **một cạnh phụ (chord)** được thêm
- 

### 🧠 Nhiệm vụ của bạn:

- Với mỗi test, bạn có thể **tối đa 40 lần truy vấn** kiểu:

? x y

và bạn sẽ nhận lại độ dài đường đi ngắn nhất từ đỉnh  $x$  đến  $y$  (tính theo số cạnh).

- Sau đó bạn cần đoán vị trí của chord:

! u v

→ Nếu đúng, bạn nhận lại ``1``

→ Nếu sai, bạn nhận lại ``-1`` và chương trình sẽ **dừng ngay lập tức**



---

## Chi tiết đồ thị ban đầu:

- Gồm  $n$  đỉnh ( $1 \leq i \leq n$ )
  - Cạnh giữa  $i$  và  $(i \bmod n) + 1$   
⇒ Gọi là chu trình vòng
- 

## Cách phát hiện chord:

### Ý tưởng chính:

- Trong chu trình gốc, độ dài ngắn nhất giữa hai đỉnh  $x$  và  $y$  là:

$$d(x, y) = \min(|x - y|, n - |x - y|)$$

- Nếu một truy vấn trả về giá trị **nhỏ hơn** giá trị trên, thì có khả năng tồn tại **chord** kết nối  $x$  và  $y$

### Ví dụ:

- $n = 6$ , kiểm tra  $x = 1, y = 5$ 
    - Trong chu trình: khoảng cách =  $\min(4, 2) = 2$
    - Nếu truy vấn trả về 1 → rõ ràng có chord nối giữa 1 và 5
- 

## Giao tiếp:

- Tương tác với trình chấm:
  - Gửi truy vấn ``? x y``, đọc kết quả
  - Sau khi xác định chord, gửi ``! u v``, đọc kết quả ``1`` hoặc ``-1``

- Không tính truy vấn đoán chord vào giới hạn 40

### Ví dụ:

Input:

2  
6  
2

Output:

? 1 5  
← 1  
? 2 4  
← 2  
! 4 2  
← 1

### Tóm tắt chiến lược hiệu quả:

- Vì  $n \leq 10^9$ , không thể duyệt hết.
- Tận dụng việc truy vấn trả về khoảng cách nhỏ hơn "chu trình gốc" để phát hiện chord.
- Có thể:
  - Chọn một đỉnh làm gốc (ví dụ: 1), truy vấn với một số đỉnh ngẫu nhiên khác (cách đều) và lưu lại kết quả
  - Khi phát hiện độ lệch → xác định đỉnh đích liên quan đến chord
- Tối đa 40 truy vấn: nên dùng nhị phân hoặc các kỹ thuật chia đều để dò

---

Bạn muốn mình triển khai **code mẫu cho Python/C++** để xử lý bài toán tương tác này chứ?

## Problem J. Japanese Bands

---

### Mô tả bài toán:

Grammy đang thiết kế một trò chơi thẻ bài (TCG) gồm:

- $n_1$  **lá bài nhân vật** (character cards)
- $n_2$  **lá bài âm nhạc** (music cards)
- Mỗi lá bài sẽ được gán **một số nguyên từ 1 đến  $m$**  đại diện cho **sức mạnh phép thuật**.

### Combo gây sát thương đặc biệt:

Có  $k$  **cặp số**  $(a_i, b_i)$  ( $1 \leq a_i, b_i \leq m$ ), gọi là **combo**.

“Với mỗi cặp  $(a_i, b_i)$ , ít nhất một trong hai điều kiện **phải được thỏa mãn**:”

- Có **lá bài nhân vật** mang giá trị  $a_i$  và **lá bài âm nhạc** mang giá trị  $b_i$ , **hoặc**
- Có **lá bài âm nhạc** mang giá trị  $a_i$  và **lá bài nhân vật** mang giá trị  $b_i$

### Mục tiêu:

Tính **số cách gán giá trị** từ 1 đến  $m$  cho  $n_1$  thẻ nhân vật và  $n_2$  thẻ âm nhạc sao cho:

- Mỗi cặp combo được thỏa mãn như trên
  - Hai cách gán được xem là khác nhau nếu **số lượng mỗi giá trị trong character hoặc music cards khác nhau**
- 



### Mô hình bài toán:

Gọi:

- $C$ : đa tập các số gán cho character cards (kích thước  $n_1$ )
- $M$ : đa tập các số gán cho music cards (kích thước  $n_2$ )

Cần đếm số cặp  $(C, M)$  sao cho với mỗi cặp combo  $(a_i, b_i)$ , thỏa mãn:

- $a_i \in C$  và  $b_i \in M$ , **hoặc**
  - $a_i \in M$  và  $b_i \in C$
- 



### Ý tưởng giải (khi $m$ nhỏ, $\leq 20$ ):

- Có tối đa  $m = 20 \rightarrow$  có thể **liệt kê mọi phân phối tần số** giá trị cho các lá bài nhân vật và âm nhạc:
  - Duyệt tất cả các cách gán số cho character: duyệt mọi tổ hợp  $(c_1, \dots, c_m)$  sao cho tổng  $= n_1$
  - Duyệt tương tự cho music:  $(m_1, \dots, m_m)$  sao cho tổng  $= n_2$
- Với mỗi cặp  $(C, M)$ , kiểm tra các combo có được thỏa mãn hay không



➡ Sử dụng **dynamic programming** để sinh tổ hợp phân phối (stars and bars / integer compositions)

---



## Số tổ hợp phân phối:

Số cách phân phối  $n$  lá bài vào  $m$  loại giá trị là:

$$\binom{n+m-1}{m-1}$$

➡ Tổng số tổ hợp  $O(m^n)$  nếu sinh tất cả, nhưng ta **có thể cắt tỉa** nhờ ràng buộc combo

---



## Chiến lược thực tế (cho $m \leq 10$ ):

- Duyệt tất cả cách phân phối giá trị cho  $C$  và  $M$
  - Với mỗi cặp  $(C, M)$ , duyệt qua tất cả combo:
    - Nếu với một combo nào đó  $(a_i, b_i)$  mà  $a_i \notin C \wedge b_i \notin M \wedge a_i \notin M \wedge b_i \notin C \Rightarrow$  loại
  - Nếu hợp lệ, cộng vào tổng
- 



## Output:

- Với mỗi test: in ra **số cách gán hợp lệ**, modulo  $10^9 + 7$
- 



## Ví dụ

Input :

```
3
2 3 3 3
2 3
1 1
2 3
2 2 2 1
1 1
1 1 10 2
1 2
1 3
```

Output :

```
6
4
0
```

## Tóm tắt đề bài:

- Cho số lượng thẻ bài nhân vật và âm nhạc:  $n_1, n_2$
- Cho phạm vi giá trị gán  $m$
- Có  $k$  combo yêu cầu phải thỏa ít nhất một trong hai điều kiện giữa character và music cards
- Tính số cách gán sao cho **tất cả các combo đều hợp lệ**

Bạn có muốn mình viết code Python tối ưu cho trường hợp  $m \leq 10$ ?

## Problem K. Kind of Bingo

## Mô tả đề bài

Bạn có một **lưới**  $n \times m$  với các ô được **đánh số từ 1 đến  $n \times m$**  theo thứ tự **hàng-major**:

- Vị trí ô  $(i, j)$  có số thứ tự:

$$\text{cell}(i, j) = (i - 1) \times m + j$$

---

## Bạn được cho một hoán vị $p$ của các số từ 1 đến $n \times m$

- Với mỗi thao tác thứ  $i$ , bạn **đánh dấu ô  $p_i$**
- Nếu sau thao tác thứ  $b$ , **một hàng nào đó có tất cả các ô đã được đánh dấu**, thì  $b$  là **số bingo**

---

## Yêu cầu:

- Bạn được phép **swap (đổi chỗ)** tối đa  $k$  lần trong dãy ban đầu
- Tính **giá trị nhỏ nhất có thể của số bingo** sau khi đổi chỗ tối đa  $k$  lần

---

## Input

- $T$ : số lượng test
- Với mỗi test:
  - Dòng 1:  $n, m, k$
  - Dòng 2: hoán vị  $p$  có  $n \times m$  phần tử

---

## Output

- Với mỗi test, in ra giá trị **bingo nhỏ nhất** có thể đạt được sau tối đa  $k$  phép hoán vị

## Chiến lược giải bài

### Mục tiêu:

Tìm chỉ số nhỏ nhất  $b \in [1, n \times m]$  sao cho:

- Sau khi thực hiện tối đa  $k$  hoán vị, có **một hàng** có tất cả các ô của nó nằm trong đoạn đầu  $p[1..b]$

### Ý tưởng:

- Với mỗi hàng  $r$ , nó gồm các chỉ số:

$$R_r = \{(r - 1) \times m + 1, \dots, r \times m\}$$

- Với đoạn đầu  $p[1..b]$ , tính xem mỗi hàng  $r$  có bao nhiêu ô nằm trong đó  $\Rightarrow$  gọi là  $\text{hit}_r$
- Để hàng  $r$  hoàn thành, ta cần:

$$\text{hit}_r + \text{swap}_r \geq m$$

với  $\text{swap}_r$  là số lần cần swap để đưa các ô còn thiếu vào hàng  $r$

 Với đoạn  $p[1..b]$ , tính  $\min \text{swap}_r$ , kiểm tra có hàng nào  $\text{swap}_r \leq k$

### Tối ưu bằng nhị phân:

- Sử dụng **binary search** trên  $b \in [1, n \times m]$
- Mỗi lần kiểm tra trung gian:



- Đếm số lượng ô của từng hàng xuất hiện trong  $p[1..b]$
- Với mỗi hàng  $r$ , tính số lượng ô cần thêm  $= m - \text{hit}_r$
- Nếu có hàng nào cần  $\leq k$  swap  $\Rightarrow \text{bingo} = b$

### Ví dụ:

#### Test 1:

```
3 5 2
1 4 13 6 8 11 14 2 7 10 3 15 9 5 12
```

→ Sau 2 swap ta có thể đưa 1 hàng vào trọn vẹn trong 7 phần tử đầu

→ **Output:** `7`

### Tóm tắt đề bài

- Cho bạn dãy thứ tự đánh dấu các ô trên lưới  $n \times m$
- Bạn có thể **đổi chỗ tối đa  $k$  lần** trong dãy
- Tìm ra **bingo index nhỏ nhất** — là chỉ số sớm nhất tại đó **một hàng được đánh dấu hoàn toàn**

Bạn có muốn mình triển khai thuật toán đầy đủ hoặc code mẫu Python cho bài này không?

## Problem L. Let's Go! New Adventure

---

### Bối cảnh câu chuyện:

- Trong game **Pishin**, mỗi nhân vật có một **cấp độ phiêu lưu** (adventure rank) từ  $0 \rightarrow m$
- Để lên cấp từ  $i-1 \rightarrow i$ , cần đạt đủ **EXP** là  $b_i$ .
- Tổng EXP để lên đến cấp  $k$  là:

$$\text{req}_k = \sum_{i=1}^k b_i$$

(chuỗi này **không giảm**:  $b_i \leq b_{i+1}$ )

---

### Luật chơi của Grammy:

- Có **n ngày** chơi
- Mỗi ngày  $i$ , chơi game sẽ nhận được  $a_i$  EXP
- Grammy mỗi lần chỉ chơi **duy nhất một nhân vật liên tục**, và **không quay lại nhân vật cũ**
- Mỗi chuỗi ngày chơi liên nhau sẽ tạo ra **1 nhân vật** mới
- Mục tiêu: **tối đa hóa**:

tổng các cấp độ đạt được  $- c \times (\text{số nhân vật})$

---



## Nhiệm vụ của bạn:

Tìm cách chia dãy  $a[1..n]$  thành các đoạn liên tiếp

→ mỗi đoạn đại diện cho **một nhân vật được chơi liên tục**

→ tính tổng cấp độ mà nhân vật đạt được theo lượng EXP trong đoạn đó

→ **Tối đa hóa:**

$$\text{tổng cấp độ} - c \times \text{số nhân vật}$$



## Input:

- $T$ : số test ( $\leq 5 \times 10^4$ )
- Với mỗi test:
  - Dòng 1:  $n, m, c$
  - Dòng 2:  $a_1, \dots, a_n$  (EXP mỗi ngày)
  - Dòng 3:  $b_1, \dots, b_m$  (EXP cần để lên cấp)

⚠ Tổng  $\sum n$  và  $\sum m$  qua tất cả test  $\leq 5 \times 10^5$



## Output:

- Với mỗi test, in ra một số nguyên — giá trị tối ưu có thể đạt được



## Ý tưởng giải bài toán:

✓ **Bước 1: Tiền xử lý  $req[k]$ :**



- $\text{req}[k] = b_1 + b_2 + \dots + b_k$

→ Dùng để tra xem với  $\text{EXP} = \text{sum}$ , sẽ lên được cấp bao nhiêu (nhị phân)

## ✓ Bước 2: Duyệt toàn bộ đoạn ngày:

Gọi:

- $\text{dp}[i]$  = giá trị lớn nhất có thể đạt được từ ngày 1 đến ngày  $i$
- Duyệt  $i$  từ  $1 \rightarrow n$ :
  - Duyệt ngược  $j$  từ  $i$  về 1 (hoặc dùng **cửa sổ sliding window**):
    - Tính  $\text{sum} = a[j] + a[j+1] + \dots + a[i]$
    - Tính  $\text{level} = \max k$  sao cho  $\text{req}[k] \leq \text{sum}$
    - Cập nhật:

$$\text{dp}[i] = \max(\text{dp}[i], \text{dp}[j-1] + \text{level} - c)$$

- Gán  $\text{dp}[0] = 0$

**Tối ưu hơn**, ta dùng **quy hoạch động đoạn liên tiếp** ( $\text{Kadane} + \text{lợi ích}$ ):

## ✓ Bước 3: Chuyển bài toán thành bài toán đoạn con:

- Mỗi đoạn liên tiếp  $[1..r]$ :
  - Tổng EXP:  $s = a[1] + \dots + a[r]$
  - Suy ra  $\text{level} = \max k: \text{req}[k] \leq s$
  - Gọi  $\text{value} = \text{level} - c$ 
    - Tối đa hóa tổng  $\text{value}$  trên một tập các đoạn không giao nhau

➡ Dùng **Greedy + Sliding window** + tiền xử lý mảng  $\text{req}$  + nhị phân tìm cấp độ

 **Ví dụ:**

### Input:

```
2
5 4 2
1 0 3 1 2
0 1 1 2
4 5 1
7 16 23 4
1 3 6 20 20
```

### Output:

```
3
6
```

#### ✅ Test 1:

- Chơi từ ngày 1 → 3: tổng EXP = 4 → lên cấp 4
- Chơi từ ngày 4 → 5: EXP = 3 → lên cấp 3
- Tổng =  $(4-2) + (3-2) = 3$

#### ✅ Tóm tắt đề bài:

- Bạn được chơi game `n` ngày, mỗi ngày có lượng EXP `a\_i`
- Mỗi đoạn liên tục là 1 nhân vật → khi dừng là bỏ
- Mỗi nhân vật đạt cấp tương ứng với tổng EXP chơi liên tục
- **Tối đa hóa:**

Tổng cấp độ đạt được  $= c \times (\text{số nhân vật})$

---

Bạn có muốn mình viết **code Python tối ưu với phân đoạn và nhị phân tìm cấp độ** cho bài này không?

## Bài M. Make It Divisible

---

### Đề bài:

Cho một dãy số nguyên dương  $a_1, a_2, \dots, a_n$ , ta nói một đoạn con  $[l, r]$  là **divisible interval** nếu tồn tại một số nguyên  $d$  thuộc đoạn  $[l, r]$  sao cho **mọi phần tử trong đoạn  $[l, r]$  đều chia hết cho  $a_d$** .

Một dãy được gọi là **divisible sequence** nếu **mọi đoạn con liên tiếp  $[l, r]$  đều là divisible interval**.

---

Bây giờ bạn được cho:

- Một dãy  $b_1, b_2, \dots, b_n$
- Một số nguyên  $k$

Yêu cầu: Tìm tất cả các giá trị nguyên  $x$  sao cho:

- $1 \leq x \leq k$
- Dãy mới  $b_1 + x, b_2 + x, \dots, b_n + x$  trở thành **divisible sequence**

Do có thể có nhiều giá trị  $x$  thỏa mãn, bạn chỉ cần in ra:

1. Số lượng giá trị  $x$  hợp lệ
2. Tổng các giá trị  $x$  đó

---

## Input:

- Số bộ test  $T$  ( $1 \leq T \leq 500$ )
- Với mỗi test:
  - Dòng 1: 2 số nguyên  $n, k$
  - Dòng 2:  $n$  số nguyên  $b_1, b_2, \dots, b_n$

(Đảm bảo tổng  $n$  trên tất cả test  $\leq 50,000$ )

---

## Output:

Với mỗi test, in ra **hai số nguyên**:

- Số lượng giá trị  $x$  hợp lệ
  - Tổng tất cả các giá trị  $x$  đó
- 

## Ví dụ:

Input :

```
3
5 10
7 79 1 7 1
2 10000000000
1 2
1 100
10000000000
```

Output :

3 8

0 0

100 5050

## Tóm tắt đề bài:

- Với mỗi  $x \in [1, k]$ , kiểm tra xem dãy  $b_i + x$  có trở thành **divisible sequence** không.
- In ra:
  - Số lượng giá trị  $x$  hợp lệ
  - Tổng các giá trị đó

Chỉ cần dịch và tóm tắt, **chưa cần thuật toán hoặc lời giải**.