คำต้องห้าม (forbiddenwords)

ในจักรวาลแห่งหนึ่ง คำคือคู่ของสัญญะที่มี M รูป (ในโจทย์จะมีค่า $M = 100$ และมีปัญหาย่อยเดียวที่ $M = 55$) การเอ่ยคำใดออกมาคือการระบุสับเซตที่มีขนาดเท่ากับ 2 ของเซต $\{0, 1, 2, \ldots, M-1\}$ ในความเข้าใจนี้ คำหนึ่งคำคือการเชื่อมโยงกันของสองสัญญะ สังเกตว่าคำเป็นเซตไม่ใช่คู่ลำดับ ดังนั้น $\{0, 1\}$ และ $\{1, 0\}$ จึงเป็นคำเดียวกัน และจะใช้สลับกันอย่างไรก็ได้

Alice อยู่ในประเทศที่ห้ามการพูดบางคำ เรียกว่า *คำต้องห้าม* การห้ามนี้งอกเงยมาจากบางคำแรกที่ดูจะเป็นภัยคุกคามต่อสัญญะ 0 จากนั้นค่อย ๆ ลุกลามไปยังสัญญะอื่น ๆ ต่อไป ทำให้ลักษณะของคำต้องห้ามทั้งหมดมีลักษณะเชื่อมต่อกันเป็นต้นไม้ของสัญญะ ซึ่งเรียกว่า *ต้นไม้ของคำต้องห้าม* ต้นไม้ดังกล่าวจะมีลักษณะดังนี้ คำต้องห้ามจะมีทั้งสิ้น N คำ ($N \leq M - 1$) ที่เชื่อมโยงสัญญะที่ $0, 1, \ldots, N$ เข้าด้วยกัน รวม $N + 1$ รูป กล่าวคือระหว่างสองสัญญะ i และ j ใด ๆ ที่ $0 \leq i, j \leq N$ อาจจะมีคำต้องห้าม $\{i, j\}$ ในรายการ หรืออาจจะมีลำดับของคำต้องห้ามที่เชื่อมโยงจากสัญญะ i ไปยังสัญญะ j ได้

Alice มีจำนวนเต็ม X ที่ $0 \leq X \leq 10^{18}$ ที่ต้องการจะส่งให้กับ Bob ที่อยู่ในประเทศอื่นผ่านทางเซตของคำจำนวนไม่เกิน K คำ (ในโจทย์จะมีค่า $K = 100$ และมีปัญหาย่อยเดียวที่ $K = 70$) แต่ในการส่งนี้ Alice ห้ามใช้คำต้องห้ามของประเทศเธอเอง และเนื่องจาก Bob อยู่คนละประเทศจึงไม่ทราบอะไรทั้งสิ้นเกี่ยวกับต้นไม้ของคำต้องห้ามในประเทศของ Alice

นอกจากนี้หมายเลขของสัญญะที่ประเทศของ Alice กับของ Bob นั้นอาจจะมีค่าไม่ตรงกัน นั่นคือจะมี permutation p ที่ Alice ไม่ทราบ ที่ถ้า Alice ส่งคำ $\{i, j\}$ ไปให้ Bob สิ่งที่ Bob จะได้รับคือ $\{p(i), p(j)\}$ (หมายเหตุ มีปัญหาย่อยที่ $p(i) = i$ ด้วย)

พิจารณาตัวอย่างต่อไปนี้ สมมติว่า $N = 10$ และคำต้องห้ามมีดังนี้

$$\{0, 1\}, \{1, 2\}, \{2, 3\}, \{2, 4\}, \{4, 5\},$$
$$\{0, 6\}, \{0, 7\}, \{7, 8\}, \{7, 9\}$$

และ $X = 123\,456$ เมื่อ Alice ทราบ X อาจจะส่งคำดังตัวอย่างด้านล่างให้กับ Bob

$$\{1, 3\}, \{3, 5\}, \{9, 8\}, \{6, 5\}$$

สังเกตว่าจะไม่มีคำใดอยู่ในรายการคำต้องห้าม ในการส่งให้ Bob สมมติว่า permutation p ที่ใช้แปลงสัญญะข้างประเทศนิยามได้เป็น $p(i) = (i + 1) \bmod M$ รายการของคำที่ Bob อาจจะได้รับหลังการแปลงคือ

$$\{9, 10\}, \{4, 6\}, \{2, 4\}, \{6, 7\}$$

จากคำดังกล่าว Bob จะต้องตอบว่า $X = 123\,456$ เช่นเดียวกัน

รายละเอียดการเขียนโปรแกรม

ฟังก์ชันที่คุณเขียนจะถูกเรียกใช้หลายรอบ (ไม่เกิน 10 รอบ) ในแต่ละรอบฟังก์ชันทั้งสองจะถูกเรียกอย่างละครั้ง ดังนั้นฟังก์ชันจะต้องถูกเขียนให้สามารถเรียกซ้ำได้หลายรอบโดยไม่ผิดพลาด

หมายเหตุ: ในการตรวจจริงบนเซิร์ฟเวอร์ อาจจะมีการเรียกใช้ฟังก์ชันทั้งสองโดยเกรดเดอร์ที่ทำงานคนละ process เพื่อจำกัดการสื่อสารระหว่างฟังก์ชัน และในการเรียกอาจจะเรียกฟังก์ชัน `alice` หลายรอบต่อเนื่องกันใน process หนึ่ง ก่อนจะส่งข้อมูลทั้งหมดให้อีก process เพื่อเรียกใช้งานฟังก์ชัน `bob` อีกหลายรอบ

สำหรับแต่ละรอบ ในขั้นตอนแรกฟังก์ชันที่ถูกเรียกคือฟังก์ชัน `alice`

```
vector<pair<int,int>> alice(int M, int N, int K, long long X,
                            vector<pair<int,int>> F)
```

- ฟังก์ชันนี้จะถูกเรียกใช้โดยเกรดเดอร์หนึ่งครั้งในหนึ่งรอบ
- ในการเรียกจะส่งจำนวนสัญญะ M, จำนวนคำต้องห้าม N, จำนวนคำที่ Alice ใช้ได้ K และจำนวนเต็ม X ที่ต้องการส่ง
- อาร์เรย์ F จะแทนต้นไม้ของคำต้องห้าม กล่าวคือ สำหรับ $0 \le i < N$ คำต้องห้ามคำที่ i คือ $\{F[i].first, F[i].second\}$
- ฟังก์ชันจะต้องคืนอาร์เรย์ของคำที่มีขนาดไม่เกิน K และสัญญะในแต่ละคำจะมีค่าระหว่าง 0 ถึง $M-1$
- จะต้องไม่มีคำใด ๆ ในอาร์เรย์ที่คืนมา อยู่ในต้นไม้ของคำต้องห้ามเลย (อย่าลืมว่าการตรวจสอบจะทำแบบเซตไม่ใช่คู่ลำดับ)

จากนั้นเกรดเดอร์จะนำรายการของคำที่คุณส่งมาจากฟังก์ชัน `alice` ไปส่งให้กับโปรแกรมของคุณเองในการเรียกใช้โปรแกรมครั้งที่สอง ในการเรียกใช้นี้ โปรแกรมจะไม่มีข้อมูลใด ๆ จากการเรียกใช้ครั้งแรกเลย

และเนื่องจากการส่งข้อมูลเป็นรายการของคำ (ที่ไม่มีลำดับ) รายการของคำที่ถูกส่งมาอาจ `alice` จะถูกนำไปแปลง เช่น สลับตำแหน่งใน `pair` หรือสลับลำดับในอาร์เรย์ จากนั้นสัญญะในคำทั้งหมดจะถูกนำไปผ่าน permutation p ด้วย

ถ้ารายการของคำที่ส่งจากฟังก์ชัน `alice` นั้นไม่ผิดเงื่อนไข ในการเรียกครั้งที่สองเกรดเดอร์จะเรียกฟังก์ชัน `bob`

```
long long bob(int M, vector<pair<int,int>> W)
```

- ฟังก์ชันนี้จะถูกเรียกใช้ 1 ครั้งในหนึ่งรอบ
- ฟังก์ชันจะรับอาร์เรย์ของคำที่ส่งมาจากฟังก์ชัน `alice` หลังจากผ่านการแปลงอาร์เรย์และสัญญะแล้ว
- จะต้องคืนค่า X ให้ตรงกับที่มีการเรียกใช้ในฟังก์ชัน `alice`

เงื่อนไข

- $M \in \{100, 55\}, K \in \{100, 70\}$
- $2 \le N \le M-1$
- $0 \le X \le 10^{18}$

ปัญหาย่อย

1. (6 points) $M = 100, K = 100, X \leq 100$
2. (6 points) $M = 100, K = 100, N = M - 1$, ต้นไม้ของคำต้องห้ามเป็นดาว (star) ที่จุดศูนย์กลางอยู่ที่สัญญะ 0, และ permutation p ของสัญญะสอดคล้องกับเงื่อนไข $p(i) = i$
3. (6 points) $M = 100, K = 100, N = M - 1$ และต้นไม้ของคำต้องห้ามเป็นดาว (star) ที่จุดศูนย์กลางอยู่ที่สัญญะ 0
4. (6 points) $M = 100, K = 100, N = M - 1$ และต้นไม้ของคำต้องห้ามเป็นเส้น (line) เริ่มที่ที่สัญญะ 0 สิ้นสุดที่สัญญะ $M - 1$
5. (16 points) $M = 100, K = 100, X \leq 10^6$
6. (24 points) $M = 100, K = 100, X \leq 10^9$
7. (26 points) $M = 100, K = 100, X \leq 10^{18}$
8. (10 points) $M = 55, K = 70, X \leq 10^{18}$

ตัวอย่าง

ในตัวอย่างข้างต้นถ้า $M = 100, K = 100$ ในการทำงานหนึ่งรอบ เกรดเดอร์จะเรียกฟังก์ชัน `alice` ดังนี้

```
alice(100,10,100,123456,
      [[0,1], [1,2], [2,3], [2,4], [4,5],
       [0,6], [0,7], [7,8], [7,9]])
```

ฟังก์ชันดังกล่าวอาจจะคืนค่าเป็น

```
[[1,3], [3,5], [9,8], [6,5]]
```

รายการของคำดังกล่าวอาจจะถูกแปลงเป็น `[[9,10], [4,6], [2,4], [6,7]]` ก่อนจะส่งไปให้กับฟังก์ชัน `bob`

```
bob(100, [[9,10], [4,6], [2,4], [6,7]])
```

ซึ่งถ้าทุกอย่างถูกต้องจะต้องคืนค่า `123456` กลับมา

กระบวนการทำงานบนเกรดเดอร์จริง

ในเกรดเดอร์จริงจะมีการแยก process การทำงานของ `alice` และ `bob` ออกจากกัน ทำให้สื่อสารผ่านกันได้ในรูปแบบที่กำหนดเท่านั้น โดยทำผ่านทางโค้ดของเกรดเดอร์เท่านั้น

เกรดเดอร์จริงจะเรียก `alice` ใน process แรกทำงานหลายครั้ง จากนั้นถ้าผลลัพธ์ที่ได้ไม่มีข้อผิดพลาด จะนำผลลัพธ์มาผ่านการแปลงก่อนจะส่งให้ฟังก์ชัน `bob` ใน process ที่สองเพื่อสร้างกลับค่า X

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะเรียกใช้ฟังก์ชัน `alice` และ `bob` ภายใน process เดียวกัน ดังนั้นอาจจะมีความแตกต่างในการทำงานกับเกรดเดอร์จริงถ้าฟังก์ชันทั้งสองของคุณมีการใช้ตัวแปร global ร่วมกัน ควรระวังในประเด็นนี้

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าดังนี้

- บรรทัดที่ 1 ระบุจำนวนเต็ม R แทนจำนวนรอบในการทดสอบ

หลังจากนั้นข้อมูลนำเข้าจะมี R ชุดในรูปแบบต่อไปนี้

- บรรทัดที่ 1: $M \;\; N \;\; K \;\; X$
- บรรทัดที่ $2+i$ to $2+N-1$: $F[i].first \;\; F[i].second$

จากนั้นเกรดเดอร์จะเรียกใช้งาน `alice` และ `bob` จำนวน R รอบ ทุกรอบจะมีการ permute ผลลัพธ์จาก `alice` ก่อนส่งให้ `bob` โดยคุณสามารถแก้พฤติกรรมได้ในการเรียกใช้ฟังก์ชัน `transform_words`

ถ้า `alice` และ `bob` ทำงานถูกต้องเกรดเดอร์จะพิมพ์ `Correct`

ขีดจำกัด

- Time limit: 1 seconds
- Memory limit: 1024 MB