Alice, Bob, and Circuit

Tổ chức Cyberland Circuit bao gồm n thành viên. Mỗi thành viên có con số yêu thích của mình và một tên riêng (số yêu thích có thể không khác nhau).

m bức thư đã được gửi giữa các thành viên. Mỗi bức thư từ một người gửi đến một người nhận và nội dung của bức thư là con số yêu thích của người gửi.

Mỗi thành viên tính tổng nội dung (con số yêu thích của người gửi) mà họ nhận được và lấy phần dư trong phép chia cho 65536 (tức là 2^{16}) làm số kết quả của mình.

Nhiệm vụ của bạn là xác định tất cả các số kết quả.

Tuy nhiên, tình hình không đơn giản như vậy. Alice, Bob và Circuit quyết định giải bài toán này theo cách phức tạp hơn một chút:

- Alice biết tất cả n thành viên (tên và số yêu thích), nhưng không biết thông tin về các bức thư. Cô ấy cần gửi một chuỗi nhị phân tới Circuit có độ dài không quá 10^5.
- Bob biết tất cả m bức thư (tên người gửi và người nhận), nhưng không biết thông tin về các thành viên. Anh ấy cần gửi một chuỗi nhị phân tới Circuit có độ dài không quá 10^5.
- Circuit có thể nhận các chuỗi nhị phân do Alice và Bob gửi, sau đó tạo ra một chuỗi nhị phân chứa $16n$ bit làm đầu ra. Tuy nhiên, do khả năng tính toán hạn chế, Circuit chỉ có khả năng thực hiện các phép toán logic cơ bản (ví dụ: AND, OR, NOT).

Sau đây, chúng tôi sẽ giới thiệu chi tiết cách thức hoạt động của mạch.

Chi tiết về mạch

Cổng là thành phần cơ bản của một mạch. Một cổng bao gồm không hoặc hai đầu vào bool (phụ thuộc vào loại cổng) và một đầu ra bool. Có hai loại cổng: cổng dữ liệu và cổng tính toán.

- Cổng dữ liệu không có đầu vào và biểu diễn cho các bit từ chuỗi nhị phân được gửi bởi Alice và Bob.
  - Sẽ có $l_A + l_B$ cổng dữ liệu, được gắn nhãn từ 0 đến $(l_A + l_B - 1)$, trong đó l_A, l_B lần lượt là độ dài của các chuỗi từ Alice và Bob.
  - Đối với $0 \le i < l_A$, thì đầu ra của cổng thứ i là bit thứ i của chuỗi từ Alice;
  - Đối với $0 \le i < l_B$, đầu ra của cổng thứ $(i + l_A)$ là bit thứ i của chuỗi từ Bob.
- Cổng tính toán có hai đầu vào và biểu diễn quá trình tính toán.
  - Nhãn cho các cổng tính toán bắt đầu từ $(l_A + l_B)$.
  - Đối với mỗi cổng tính toán, bạn cần cung cấp nhãn của hai cổng phụ thuộc làm đầu vào và loại hoạt động $p (0 \le p \le 15)$.
    - Để tránh phụ thuộc vòng tròn, nhãn của hai cổng phụ thuộc phải nhỏ hơn nhãn của cổng tính toán.
    - Nếu đầu ra của hai cổng phụ thuộc lần lượt là x_0 và x_1 ($x_0, x_1 \in \{0, 1\}$) thì đầu ra của cổng tính toán là:

$$f(p, x_0, x_1) = \left\lfloor \frac{p}{2^{x_0 + 2x_1}} \right\rfloor \bmod 2$$

Dưới đây là một số ví dụ có thể hữu ích cho bạn:

x_0	x_1	x_0 AND x_1 $f(8, x_0, x_1)$	x_0 OR x_1 $f(14, x_0, x_1)$	x_0 XOR x_1 $f(6, x_0, x_1)$	NOT x_0 $f(5, x_0, x_1)$
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

Chi tiết cài đặt

Xin lưu ý:

- Tất cả các chỉ số mảng bắt đầu từ 0. Ví dụ: nếu `a` là một mảng có độ dài `n`, thì `a[0]` đến `a[n-1]` là dữ liệu hợp lệ, việc truy cập các chỉ số ngoài phạm vi này có thể gây ra lỗi tràn giới hạn .
- Tất cả các chuỗi được kết thúc bằng ký tự null `\0`.

Bạn cần xây dựng các hàm sau:

Alice

```
int alice(const int n, const char names[][5], const unsigned short numbers[], bool outputs_alice[]);
```

Chỉ thị	Giá trị	Kích thước	Ý nghĩa	Ràng buộc
Input	n	1	n	$0 \le n \le 700$
	names	n	Tên của mỗi thành viên.	Tất cả các tên đều khác nhau, chỉ bao gồm các chữ cái in thường tiếng Anh và có độ dài tối đa là 4 kí tự.
	numbers	n	Số yêu thích của mỗi thành viên.	Mỗi số thuộc đoạn từ 0 đến 65535.
Output	outputs_alice	l_A	Chuỗi nhị phân được gửi cho Circuit.	
	(Hàm trả về)	1	l_A	**Bạn cần bảo đảm rằng l_A không vượt quá 10^5 và khi n bằng nhau thì l_A không đổi.**

Bob

```
int bob(const int m, const char senders[][5], const char recipients[][5], bool outputs_bob[]);
```

Chỉ thị	Giá trị	Kích thước	Ý nghĩa	Ràng buộc
Input	m	1	m	$0 \le m \le 1000$
	senders	m	Tên người gửi trên mỗi bức thư.	Tất cả các tên xuất hiện trong đầu vào của Alice
	recipients	m	Tên người nhận trên mỗi bức thư.	
Output	outputs_bob	l_B	Chuỗi nhị phân được gửi cho Circuit.	
	(Hàm trả về)	1	l_B	**Bạn cần bảo đảm rằng l_B không vượt quá 10^5 và khi m bằng nhau thì l_B không đổi.**

Circuit

Để bảo đảm rằng quá trình tính toán của Circuit giống như một mạch chung, bạn không thể lấy trực tiếp các chuỗi nhị phân được gửi từ Alice và Bob đến Circuit. Bạn chỉ biết độ dài của hai chuỗi này và đưa ra cấu trúc mạch.

```
int circuit(const int la, const int lb, int operations[], int operands[][2], int outputs_circuit[][16]);
```

Chỉ thị	Giá trị	Kích thước	Ý nghĩa	Ràng buộc
Input	la	1	l_A	
	lb	1	l_B	

Chỉ thị	Giá trị	Kích thước	Ý nghĩa	Ràng buộc
Output	operations	l	Loại hoạt động được thực hiện bởi mỗi cổng trong mạch.	Một số nguyên từ 0 đến 15.
	operands	l	Nhãn cổng được sử dụng bởi mỗi cổng trong mạch.	Số phải nhỏ hơn nhãn của cổng hiện tại.
	outputs_circuit	n	Nhãn cổng của mạch trả ra.	`outputs_circuit[i][j]` nhãn cổng trả ra bit thứ j (tính từ bit trọng số nhỏ nhất) của kết quả cuối cùng cho thành viên thứ i. Các thành viên được sắp xếp theo đầu vào của Alice.
	(Hàm trả về)	1	l, là tổng số cổng (bao gồm cả cổng dữ liệu).	**Bạn cần bảo đảm rằng** $l \leq 2 \times 10^7$

Mặc dù bạn có thể sửa đổi thông tin của các cổng có chỉ số nhỏ hơn $l_A + l_B$ trong mảng `operations` và `operands`, trình chấm sẽ bỏ qua sự sửa đổi đó.

Ví dụ

Xét các lời gọi hàm:

```
alice(3, {"alic", "bob", "circ"}, {10000, 20000, 30000}, outputs_alice);
bob(5, {"alic", "bob", "bob", "circ", "circ"}, {"circ", "circ", "alic", "circ", "circ"}, outputs_bob);
```

thể hiện cho kịch bản sau:

- Alice biết có 3 thành viên, thành viên với tên `alic` có số yêu thích 10000, ... Một đầu ra có thể có cho `alice()` là,
  - Giá trị trả về của `alice()` là 2, thể hiện cho $l_A = 2$.
  - Bên trong hàm `alice()`, đặt `outputs_alice[0] = 1, outputs_alice[1] = 0`, biểu thị chuỗi nhị phân kết quả là 10.
- Bob biết có 5 bức thư, bức thư đầu tiên là từ `alic` gửi đến `circ`, ... Một đầu ra có thể cho `bob()` là,
  - Giá trị trả về của `bob()` là 3, thể hiện cho $l_B = 3$.
  - Bên trong hàm `bob()`, đặt `outputs_bob[0] = 1, outputs_bob[1] = 1, outputs_bob[2] = 0`, biểu thị chuỗi nhị phân kết quả là 110.

Dựa trên kết quả đầu ra trước đó của `alice()` và `bob()`, sẽ có lệnh gọi sau:

```
circuit(2, 3, operations, operands, outputs_circuit);
```

Một đầu ra đúng cho hàm này sẽ là

- Giá trị trả về của `circuit()` là 7, nghĩa là chúng ta thêm hai cổng tính toán, có nhãn là 5 và 6.
- Bên trong `circuit()`, đặt `operations`, `operands`, và `outputs_circuit` theo cách sau:
  - `operations = {-1, -1, -1, -1, -1, 8, 14}`, trong đó chúng tôi sử dụng `-1` để biểu thị thông tin bị bỏ qua từ các cổng đầu vào;
  - `operands = {{-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {-1, -1}, {0, 4} , {2, 5}};`
  - `outputs_circuit = {{5, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 6, 5, 5, 6, 5}, ...}`. Mảng hơi dài, bạn có thể kiểm tra `abc.cpp` trong phần đính kèm để biết mảng đầy đủ.

Theo đầu ra, quy trình tính toán là,

- Thêm cổng tính toán loại hoạt động 8, với đầu vào từ cổng 0 và cổng 4. Đầu ra của cổng 0 là bit thứ 0 của chuỗi từ Alice, là 1; Đầu ra của cổng 4 là bit thứ 2 của chuỗi từ Bob, là 0. Vì vậy, đầu ra cho cổng 5 là $f(8, 0, 1) = 0 \text{ AND } 1 = 0$.
- Thêm cổng tính toán loại hoạt động 14, với đầu vào từ cổng 2 và cổng 5. Đầu ra của cổng 2 là bit thứ 0 của chuỗi từ Bob, là 1; Đầu ra của cổng 5 là 0. Vì vậy, đầu ra cho cổng 6 là $f(14, 1, 0) = 1 \text{ OR } 0 = 1$.
- `output_circuit[0]` cho biết kết quả cuối cùng của `alic`, là $(0100111000100000)_2 = 20000$. Vì `alic` chỉ nhận được một lá thư từ `bob`, nên kết quả cuối cùng của `alic` là 20000.
- Kết quả cuối cùng của `bob` phải là 0, vì anh ta không nhận được lá thư nào; Kết quả cuối cùng của `circ` phải là $(10000 + 20000 + 30000 + 30000) \bmod 65536 = 24464$.

`abc.cpp` trong tệp đính kèm có thể vượt qua ví dụ này, nhưng chúng tôi không bảo đảm rằng nó có thể vượt qua các trường hợp thử nghiệm khác.

Ràng buộc

Đối với tất cả các trường hợp thử nghiệm:

- $0 \le n \le 700, 0 \le m \le 1000$.
- Tất cả các tên đều khác nhau, chỉ bao gồm các chữ cái in thường tiếng Anh và có độ dài tối đa là 4 kí tự.
- Con số yêu thích của mỗi thành viên nằm trong khoảng từ 0 đến 65535.
- Tên của tất cả người gửi và người nhận xuất hiện trong mảng `names` đầu vào của Alice.
- `alice()` và `bob()` có giới hạn bộ nhớ là 2048 MiB và giới hạn thời gian chạy là $0,02$ giây.
- `circuit()` có giới hạn bộ nhớ là 2048 MiB và giới hạn thời gian chạy là 7 giây.

Khi chấm điểm, `alice()` và `bob()` có thể được gọi nhiều lần trong một trường hợp thử nghiệm. Giới hạn thời gian là $0,02$ giây cho mỗi lần gọi.

Subtask

Subtask loại A (12 điểm)

Subtask 1,2,3 thuộc subtask loại A, trong đó $n = 1$.

Mỗi subtask có các ràng buộc bổ sung sau:

- Subtask 1 (4 điểm): $m = 0$.
- Subtask 2 (4 điểm): $0 \le m \le 1$.
- Subtask 3 (4 điểm): $0 \le m \le 1000$.

Subtask loại B (54 điểm)

Subtask 4,5,6 thuộc subtask loại B, trong đó:

- $0 \le n \le 30, \frac{n}{2} \le m \le n^2$.
- Không có hai bức thư nào có cùng người gửi và người nhận.
- Tất cả tên thành viên xuất hiện trong đầu vào của Bob (nghĩa là mỗi thành viên gửi ít nhất một lá thư hoặc nhận ít nhất một lá thư).

Mỗi subtask có các ràng buộc bổ sung sau:

- Subtask 4 (24 điểm): $n = 26$, tên của tất cả các thành viên là chữ cái viết thường và trong đầu vào của Alice, chúng xuất hiện theo thứ tự từ `a` đến `z`.
- Subtask 5 (24 điểm): $n = 26$.
- Subtask 6 (6 điểm): Không có ràng buộc nào thêm.

Subtask loại C (34 điểm)

Subtask 7,8,9 thuộc subtask loại C, trong đó $0 \le n \le 700, 0 \le m \le 1000$.

Mỗi subtask có các ràng buộc bổ sung sau:

- Subtask 7 (18 điểm): $n = 676$, tên của tất cả các thành viên là hai chữ cái viết thường và trong đầu vào của Alice, chúng xuất hiện theo thứ tự từ điển (ví dụ: `aa`, `ab`, `ac`, .. ., `az`, `ba`, ..., `bz`, `ca`, ..., `zz`).
- Subtask 8 (10 điểm): $n = 676$.
- Subtask 9 (6 điểm): Không có ràng buộc nào thêm.

Trình chấm mẫu

Trình chấm mẫu đọc dữ liệu vào theo khuôn dạng sau:

- Dòng 1: $n\ m$
- Dòng $2 + i (0 \le i \le n - 1)$: $names_i\ numbers_i$
- Dòng $2 + n + i (0 \le i \le m - 1)$: $senders_i\ recipients_i$.

Trình chấm mẫu ghi dữ liệu ra theo khuôn dạng sau:

- Nếu chương trình kết thúc thành công, trình chấm mẫu sẽ ghi ra n dòng, mỗi dòng chứa một số nguyên là kết quả cuối cùng được tính bằng các hàm bạn xây dựng cho từng thành viên.
- Trái lại, trình chấm sẽ không ghi kết quả gì ra thiết bị xuất chuẩn và in thông báo lỗi ra tệp `abc.log` trong thư mục.
- Ngoài ra, trình chấm mẫu sẽ xuất các giá trị của l_A, l_B, l và thời gian chạy của từng hàm vào tệp `abc.log`.

Trình chấm mẫu sẽ không kiểm tra giới hạn bộ nhớ và ràng buộc đối với n / m giống nhau thì l_A / l_B phải bằng nhau.