수천개의 섬

수천개의 섬은 자바 해역에 위치한 아름다운 섬들의 그룹이다. 수천개의 섬은 N개의 섬으로 구성되며 0부터 $N-1$까지 번호가 붙어 있다.

섬들 사이를 오가는데 사용될 수 있는 카누가 M개 있고 0부터 $M-1$까지 번호가 붙어 있다. $0 \leq i \leq M-1$인 각 i에 대해, i번 카누는 섬 $U[i]$나 섬 $V[i]$에 정박해 있거나 $U[i]$와 $V[i]$ 사이를 운항 중일 수 있다. 특별히, 카누가 섬 $U[i]$에 정박해 있을 때에는 섬 $U[i]$에서 섬 $V[i]$로 운항할 수 있고 이후 섬 $V[i]$에 정박하게 된다. 비슷하게, 카누가 섬 $V[i]$에 정박해 있을 때에는 섬 $V[i]$에서 섬 $U[i]$로 운항할 수 있고 이후 섬 $U[i]$에 정박하게 된다. 처음에 카누는 섬 $U[i]$에 정박해 있다. 여러 카누가 동일한 두 섬 사이를 오가는 것도 가능하다. 한 섬에 여러 카누가 정박하는 것도 가능하다.

안전상의 이유로 카누는 매 운항 이후 유지보수가 필요하고, 이로 인해 같은 카누가 두 번 연속해서 운항하는 것을 금지한다. 즉, i번 카누가 사용된 후에는, i번 카누가 다시 사용되기 전에 다른 카누가 반드시 사용되어야 한다.

부 뎅클렉은 몇 개의 섬을 여행할 계획을 세우려고 한다. 그녀의 여행이 **유효**하다는 것은 다음 조건들이 만족됨을 의미한다.

- 그녀의 여행은 섬 0에서 시작해서 섬 0에서 끝난다.
- 그녀는 섬 0이 아닌 다른 섬을 최소한 하나는 방문한다.
- 여행이 끝나면, 각 카누는 여행이 시작되기 전과 동일한 섬에 정박하게 된다. 즉, $0 \leq i \leq M-1$인 각 i에 대해, i번 카누는 섬 $U[i]$에 정박해야만 한다.

당신은 부 뎅클렉이 최대 $2\,000\,000$번 운항하는 유효한 여행을 찾도록 도와주거나, 그러한 유효한 여행이 없음을 결정하도록 도와야 한다. 이 문제에서 제시하는 제약조건(Constraints 부분 참고) 하에서는, 만약 유효한 여행이 존재한다면 운항 횟수가 $2\,000\,000$번을 넘지 않는 유효한 여행이 존재함을 증명할 수 있다.

Implementation Details

다음 함수를 구현해야 한다:

```
union(bool, int[]) find_journey(int N, int M, int[] U, int[] V)
```

- N: 섬의 개수.
- M: 카누의 개수.
- U, V: 카누를 나타내는 길이 M인 배열.
- 이 함수는 boolean 또는 정수 배열을 리턴해야 한다.
    - 유효한 여행이 존재하지 않는다면 `false`를 리턴한다.

    - 유효한 여행이 존재한다면, 두 가지 옵션이 있다:
        - 만점을 받기 위해서는, 유효한 여행을 나타내는 최대 $2\ 000\ 000$개의 정수값을 갖는 배열을 리턴해야 한다. 더 정확히는, 이 배열의 원소들은 여행에 사용되는 카누들의 번호(사용되는 순서대로)여야 한다.
        - 부분 점수를 받기 위해서는, `true`를 리턴하거나, $2\ 000\ 000$개보다 많은 정수값을 갖는 배열을 리턴하거나, 또는 유효한 여행을 나타내지 않는 정수 배열을 리턴해야 한다(더 자세한 내용은 Subtasks 부분 참고).
- 이 함수는 정확히 한번만 호출된다.

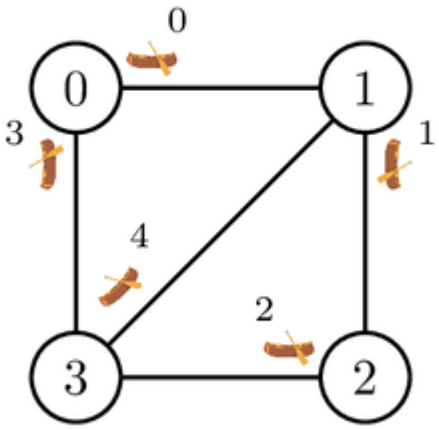
Examples

Example 1

다음 호출을 생각해보자:

```
find_journey(4, 5, [0, 1, 2, 0, 3], [1, 2, 3, 3, 1])
```

섬들과 카누들은 아래 그림과 같다.



유효한 여행으로 가능한 한가지는 다음과 같다. 부 뎅클렉은 처음에 $0, 1, 2, 4$번 카누를 순서대로 운항한다. 그 결과, 그녀는 섬 1에 있게 된다. 그 다음, 현재 섬 1에 정박해 있고 마지막에 사용한 카누가 아닌 0번 카누를 부 뎅클렉은 운항할 수 있다. 0번 카누를 다시 운항하면, 부 뎅클렉은 이제 섬 0에 있게 된다. 하지만 $1, 2, 4$번 카누가 여행 이전과 같은 섬에 정박해 있지 않다. 부 뎅클렉은 여행을 계속 이어서 $3, 2, 1, 4, 3$번 카누를 운항한다. 부 뎅클렉은 섬 0으로 돌아오고 모든 카누들도 여행 전과 동일한 섬에 정박해 있다.

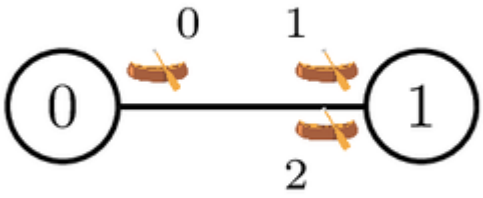
따라서, 리턴 값 $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$은 유효한 여행을 나타낸다.

Example 2

다음 호출을 생각해보자:

```
find_journey(2, 3, [0, 1, 1], [1, 0, 0])
```

섬들과 카누들은 아래 그림과 같다.



부 뎅클렉은 0번 카누를 운항해야만 하고, 그 다음 그녀는 1 또는 2번 카누를 운항할 수 있다. 참고로 그녀는 0번 카누를 연속해서 운항할 수 없다. 어느 경우던지 부 뎅클렉은 섬 0으로 돌아간다. 하지만, 카누들은 여행 전과 같은 섬에 정박해 있지 않고, 섬 0에 정박한 유일한 카누는 방금 그녀가 사용한 카누라서 부 뎅클렉은 더 이상 운항할 수 있는 카누가 없다. 유효한 여행이 없으므로, 이 함수는 `false`를 리턴해야 한다.

Constraints

- $2 \leq N \leq 100\ 000$
- $1 \leq M \leq 200\ 000$
- $0 \leq U[i] \leq N-1$이며 $0 \leq V[i] \leq N-1$ (모든 $0 \leq i \leq M-1$)
- $U[i] \neq V[i]$ (모든 $0 \leq i \leq M-1$)

Subtasks

1. (5 points) $N = 2$
2. (5 points) $N \leq 400$. 서로 다른 두 섬 x와 y ($0 \leq x < y \leq N-1$)로 이뤄지는 각 쌍에 대해, 두 섬 사이를 운항하는데 사용될 수 있는 카누는 정확히 두 개이다. 둘 중 하나는 섬 x에 정박해 있고 다른 하나는 섬 y에 정박해 있다.
3. (21 points) $N \leq 1000$, M은 짝수이고, $0 \leq i \leq M-1$인 각 **짝수** i에 대해 i번과 $i+1$번 카누는 둘 다 섬 $U[i]$와 섬 $V[i]$ 사이를 운항하는데 사용될 수 있다. i번 카누는 처음에 섬 $U[i]$에 정박해 있고 $i+1$번 카누는 처음에 섬 $V[i]$에 정박해 있다. 명확히 말해, $U[i] = V[i+1]$이고 $V[i] = U[i+1]$이다.
4. (24 points) $N \leq 1000$, M은 짝수이고, $0 \leq i \leq M-1$인 각 **짝수** i에 대해 i번과 $i+1$번 카누는 둘 다 섬 $U[i]$와 섬 $V[i]$ 사이를 운항하는데 사용될 수 있다. 두 카누 모두 처음에 섬 $U[i]$에 정박해 있다. 명확히 말해, $U[i] = U[i+1]$이고 $V[i] = V[i+1]$이다.
5. (45 points) 추가적인 제한이 없다.

유효한 여행이 존재하는 각 테스트 케이스에 대해, 당신의 답은:

- 유효한 여행을 리턴한 경우 만점을 받고,

- `true`를 리턴하거나, $2\ 000\ 000$개보다 많은 정수값을 갖는 배열을 리턴하거나, 또는 유효한 여행을 나타내지 않는 정수 배열을 리턴한 경우 35%의 점수를 받고,
- 그 외의 경우 0점을 받는다.

유효한 여행이 존재하지 않는 각 테스트 케이스에 대해, 당신의 답은:

- `false`를 리턴한 경우 만점을 받고,
- 그 외의 경우 0점을 받는다.

참고로, 각 서브태스크의 최종 점수는 그 서브태스크의 테스트 케이스들의 최소 점수로 정해진다.

Sample Grader

샘플 그레이더의 입력 형식은 다음과 같다:

- line 1: $N\ M$
- line $2 + i$ ($0 \le i \le M - 1$): $U[i]\ V[i]$

샘플 그레이더는 다음 형식으로 답을 출력한다:

- `find_journey`가 `bool`을 리턴한다면:
  - line 1: 0
  - line 2: `find_journey`가 `false`를 리턴한 경우 0, 그 외의 경우 1
- `find_journey`가 `int[]`를 리턴한다면(배열의 원소를 $c[0], c[1], \ldots c[k-1]$로 두면):
  - line 1: 1
  - line 2: k
  - line 3: $c[0]\ c[1]\ \ldots\ c[k-1]$