게임

고대 이집트인들은 0부터 $n-1$로 번호를 매긴 n개의 행성을 발견하였다. 이들 사이를 이동하기 위해서 **일방 통행 순간 이동기**를 이용하여 연결하려고 한다. 각 순간 이동기는 출발 행성과 도착 행성을 연결한다. 여행자가 출발 행성에서 순간 이동기를 이용하면, 도착 행성으로 순간 이동한다. 출발 행성과 도착 행성이 같을 수도 있다. 출발 행성이 u이고 도착 행성이 v인 순간 이동기를 (u, v)로 표시하자.

순간 이동기 사용을 장려하기 위해서, 이집트의 왕인 파라오는 순간 이동기를 이용하여 여행자가 즐길 수 있는 게임을 만들었다. 여행자는 어떤 행성에서도 게임을 시작할 수 있다. 행성 $0, 1, \ldots, k-1$ $(k \leq n)$를 **특별 행성**이라고 부르자. 여행자가 특별 행성에 도착할 때마다 도장 하나를 받는다.

처음에는 각 i $(0 \leq i \leq k-2)$마다, 순간 이동기 $(i, i+1)$가 있다. 이 $k-1$ 개의 순간 이동기를 **기본 순간 이동기**라고 부르자.

순간 이동기가 하나씩 추가된다. 이렇게 순간 이동기를 추가하다 보면, 여행자가 도장을 무한히 받게 되는 경우가 생길 수 있다. 정확히 말하면, 도장을 무한히 받는 경우는 다음 조건을 만족하는 행성의 서열 $w[0], w[1], \ldots, w[t]$가 존재할 때이다.

- $1 \leq t$
- $0 \leq w[0] \leq k-1$
- $w[t] = w[0]$
- 각 i $(0 \leq i \leq t-1)$마다, 순간 이동기 $(w[i], w[i+1])$가 있다.

여행자가 기본 순간 이동기 뿐 아니라 추가된 **어떤** 순간 이동기도 사용할 수 있음에 유의하자.

당신의 임무는 파라오를 도와서, 순간 이동기를 추가할 때마다 여행자가 도장을 무한히 받는 경우가 생기는지 여부를 점검하는 것이다.

상세 구현

다음 함수를 구현해야 한다.

```
init(int n, int k)
```

- n: 행성의 수
- k: 특별 행성의 수
- 이 함수는 `add_teleporter`를 호출하기 전 정확히 한 번만 호출된다.

```
int add_teleporter(int u, int v)
```

- u, v: 추가되는 순간 이동기의 출발 행성과 도착 행성
- 이 함수는 최대 m 번 호출된다. (m 값에 대해서는 제약조건 참조)
- 만약 순간 이동기 (u, v)를 추가했을 때 여행자가 도장을 무한히 받을 수 있다면 이 함수의 리턴 값은 1이고, 그렇지 않다면 0이어야 한다.
- 일단 함수가 1을 리턴하면, 여러분의 프로그램은 종료한다.

예제

예제 1

다음 호출을 생각해보자.

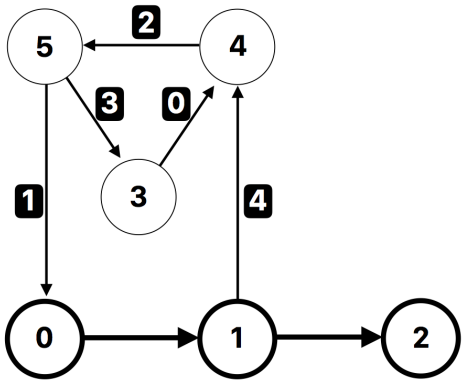
```
init(6, 3)
```

이 예제에서, 6개의 행성과 3개의 특별 행성이 있다. 행성 $0, 1, 2$가 특별 행성이다. 기본 순간 이동기는 $(0, 1), (1, 2)$이다.

그레이더가 다음과 같이 함수를 호출한다고 가정하자.

- (0) `add_teleporter(3, 4)`: 0을 리턴해야 한다.
- (1) `add_teleporter(5, 0)`: 0을 리턴해야 한다.
- (2) `add_teleporter(4, 5)`: 0을 리턴해야 한다.
- (3) `add_teleporter(5, 3)`: 0을 리턴해야 한다.
- (4) `add_teleporter(1, 4)`: 이 시점에서 무한히 도장을 받을 수 있다. 예를 들어, 여행자는 행성 0에서 시작해서, 차례로 행성 $1, 4, 5, 0, 1, 4, 5, 0, \ldots$을 방문한다. 따라서 1을 리턴하고 여러분의 프로그램이 종료한다.

다음 그림이 이 예제를 설명하고 있다.특별 행성과 기본 순간 이동기는 굵게 표시했다. `add_teleporter`로 추가된 순간 이동기는 차례대로 0부터 4로 레이블을 붙였다.



예제 2

다음 호출을 생각해보자.

```
init(4, 2)
```

이 예제에서, 4개의 행성과 2개의 특별 행성이 있다. 행성 $0, 1$가 특별 행성이다. 기본 순간 이동기는 $(0, 1)$이다.

그레이더가 다음과 같이 함수를 호출한다고 가정하자.

- `add_teleporter(1, 1)`: 순간 이동기 $(1, 1)$를 추가한 다음에는 무한히 도장을 받을 수 있다. 예를 들어, 여행자는 행성 1에서 시작해서, 순간 이동기 $(1, 1)$을 이용하여 행성 1을 무한번 방문할 수 있다. 따라서 1을 리턴하고 여러분의 프로그램이 종료한다.

다른 예제 입출력은 첨부된 패키지에서 얻을 수 있다.

제약 조건

- $1 \le n \le 300\,000$
- $1 \le m \le 500\,000$
- $1 \le k \le n$

`add_teleporter` 함수를 호출할 때마다:

- $0 \le u \le n - 1$ and $0 \le v \le n - 1$
- 순간 이동기 (u, v)를 추가하기 전에 행성 u에서 출발해서 행성 v로 가는 순간 이동기가 없다.

부분 문제

1. (2 점) $n = k, n \le 100, m \le 300$
2. (10 점) $n \le 100, m \le 300$
3. (18 점) $n \le 1\,000, m \le 5\,000$
4. (30 점) $n \le 30\,000, m \le 50\,000, k \le 1\,000$
5. (40 점) 추가적인 제약 조건이 없다.

샘플 그레이더

샘플 그레이더는 다음 형식으로 입력을 읽는다:

- line 1: $n\ m\ k$
- line $2 + i$ $(0 \le i \le m - 1)$: $u[i]\ v[i]$

샘플 그레이더는 먼저 `init`를 호출한 다음, $i = 0, 1, \ldots, m - 1$ 차례대로 $u = u[i]$, $v = v[i]$로 `add_teleporter` 함수를 호출한다.

샘플 그레이더는 `add_teleporter` 함수가 처음으로 1을 리턴한 인덱스 (0 이상 $m - 1$ 이하)를 출력한다. 만약 모든 `add_teleporter` 함수의 리턴값이 0이었다면 샘플 그레이더는 대신 m을 출력한다.

만약 `add_teleporter` 함수가 0과 1 이외의 다른 값을 리턴한다면, 샘플 그레이더는 −1을 출력하고 여러분의 프로그램은 바로 종료한다.