Simurgh

페르시아 전설의 영웅 Zal은 카불의 공주 Rudaba를 아주 사랑한다. Zal이 Rudaba와 결혼하고자 요청했을 때, Rudaba의 아버지는 Zal에게 도전 과제를 주었다.

페르시아에는 n개의 도시가 있다. 도시들은 0번부터 $n-1$번까지 번호가 붙어 있다. 또, m개의 길이 있다. 길들은 0번부터 $m-1$번까지 번호가 붙어 있다. 하나의 길은 서로 다른 두 도시를 양방향으로 직접 연결한다. 어떤 한 쌍의 도시를 직접 연결하는 도로는 최대 하나이다. 길들 중 어떤 것들은 **왕의 길**들이라고 한다. 왕의 길들은 왕족이 다닐 때 사용하는 길들이다. Zal이 해야 하는 일은 어떤 길들이 왕의 길들인지 알아내는 것이다.

Zal은 모든 도시와 길을 표시한 지도를 가지고 있다. 길들 중 어떤 것들이 왕의 길들인지는 모른다. 그런데 Zal은 Simurgh라는 전설의 새의 도움을 받을 수 있다. 하지만, Simurgh가 왕의 길들을 바로 알려주지는 않는다. 대신, Simurgh는 왕의 길들이 **황금 집합**을 이룬다는 것을 알려주었다. 어떤 길들의 집합은 다음의 조건이 만족되면 황금 집합이다.

- 집합은 정확히 $n-1$개의 길을 포함한다.
- 모든 쌍의 도시에 대해서, 쌍에 속한 한 도시에서 다른 도시로 집합에 속한 길들만 이용해서 이동이 가능하다.

추가로, Zal은 Simurgh에게 몇 개의 질문을 할 수 있다. 각 질문은 다음과 같이 진행된다.

1. Zal이 어떤 황금 집합을 정하여 질문한다.
2. Simurgh는 Zal이 정한 집합에 몇 개의 왕의 길들이 있는지 대답한다.

당신의 프로그램은 Zal을 도와, Simurgh에게 최대 q개의 질문을 해서 왕의 길들을 알아내야 한다. Grader가 Simurgh의 역할을 수행할 것이다.

Implementation details

다음 함수를 구현하여야 한다.

```
int[] find_roads(int n, int[] u, int[] v)
```

- n: 도시의 개수이다.
- u and v: 크기 m인 배열이다. 모든 $0 \leq i \leq m-1$에 대해서 $u[i]$와 $v[i]$는 길 i를 통해 직접 연결된다.
- 이 함수는 크기 $n-1$인 배열에 왕의 길들의 번호를 저장하고 리턴해야 한다. (저장된 순서는 상관없다.)

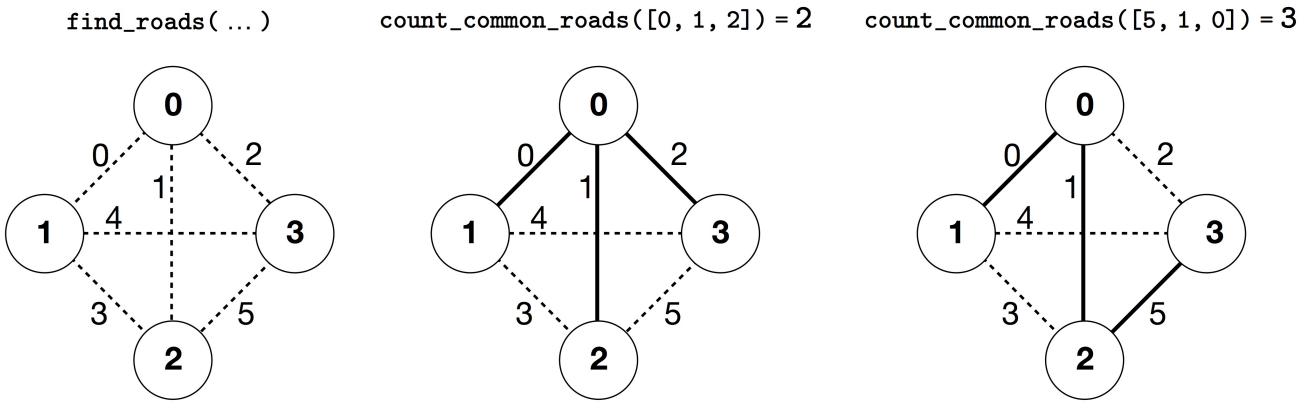
당신의 코드는 다음의 Grader 함수를 최대 q번 호출할 수 있다.

```
int count_common_roads(int[] r)
```

- r: 크기 $n-1$인 배열이다. 이 배열에 황금 집합인 길들이 저장되어야 한다. (저장된 순서는 상관 없다.)
- 이 함수는 r에 저장된 길들 중 왕의 길인 것들의 개수를 리턴한다.

Example

```
find_roads(4, [0, 0, 0, 1, 1, 2], [1, 2, 3, 2, 3, 3])
```



이 예에는 4개의 도시와 6개의 길이 있다. 두 도시 a와 b를 연결하는 길을 (a, b)로 표시한다. 길들은 다음 순서로 0번부터 5번까지 번호가 붙어 있다: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, $(2, 3)$. 모든 황금 집합은 $n-1 = 3$개의 길을 포함한다.

길 0, 1, 5번(즉, 길 $(0, 1)$,$(0, 2)$, $(2, 3)$)이 왕의 길들이고, 프로그램이 다음의 호출들을 수행하는 경우를 살펴보자.

- `count_common_roads([0, 1, 2])`: 이 호출은 2를 리턴한다. 이 질문은 길 0, 1, 2번(즉, 길 $(0, 1)$,$(0, 2)$, $(0, 3)$)에 대해 묻는 것인데, 이들 중 2개가 왕의 길들이다.
- `count_common_roads([5, 1, 0])`: 이 호출은 3을 리턴한다. 이 질문은 모든 왕의 길을 포함한 질문이다.

함수 `find_roads`는 (값들이 저장된 순서는 상관 없이) `[5, 1, 0]`을 리턴해야 한다.

다음과 같은 호출은 허용되지 않음에 주의하라.

- `count_common_roads([0, 1])`: r의 크기가 3이 아니다.
- `count_common_roads([0, 1, 3])`: r에 포함된 길들이 황금 집합이 아니다. 여기 포함된 길들인 $(0, 1)$, $(0, 2)$, $(1, 2)$을 이용해서는 0번 도시에서 3번 도시로 갈수 있는 방법이 없다.

Constraints

- $2 \leq n \leq 500$

- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (모든 $0 \leq i \leq m - 1$)
- 모든 $0 \leq i \leq m - 1$에 대해, 길 i는 서로 다른 도시를 직접 연결한다 (즉, $u[i] \neq v[i]$).
- 한 쌍의 도시를 연결하는 길은 최대 하나이다.
- 어떤 쌍의 도시 간이든 길들을 이용해서 이동하는 것이 가능하다.
- 왕의 길들 전체의 집합은 황금 집합이다.
- `find_roads`는 `count_common_roads`를 최대 q번 호출해야 한다. 각 호출에서 r에 저장된 길들은 황금 집합을 이루어야 한다.

Subtasks

1. (13 points) $n \leq 7$, $q = 30\,000$
2. (17 points) $n \leq 50$, $q = 30\,000$
3. (21 points) $n \leq 240$, $q = 30\,000$
4. (19 points) $q = 12\,000$이고, 모든 쌍의 도시를 직접 연결하는 길이 존재한다.
5. (30 points) $q = 8000$

Sample grader

Sample Grader는 다음의 형식으로 입력을 받는다.

- line 1: n m
- line $2 + i$ (모든 $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- line $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

여기서, $s[0], s[1], \ldots, s[n-2]$은 왕의 길들의 번호들이다.

Sample Grader는, `find_roads`가 `count_common_roads`를 최대 $30\,000$번 호출하고 정확한 왕의 길들을 리턴한 경우 `YES`를 출력한다. 그렇지 않은 경우 `NO`를 출력한다.

주의: Sample Grader에 포함된 `count_common_roads`는 r이 황금 집합이 되기 위한 모든 조건을 만족하는지 **확인하지 않고**, r에 포함된 길들 중 왕의 길에 포함된 것들의 개수만을 세어서 리턴한다. 당신이 실제로 프로그램을 제출했을 때 `count_common_roads` 호출에 황금 집합이 아닌 길들의 집합을 저장하여 호출하면 그 결과는 'Wrong Answer'이다.

Technical note

C++과 Pascal에 사용된 `count_common_roads` 함수는 성능 이슈 때문에 **pass by reference** 방식의 호출로 구현되어 있다. Grader는 r에 저장된 내용을 변경하지 않는 것이 보장하므로 이 부분에 대해 신경 쓸 것은 없다.