가장 긴 여행

IOI 2023 운영진은 심각한 문제가 있음을 알았다! 부다페스트로 가는 여행 계획을 세우지 않은 것이다. 하지만, 너무 늦지는 않았을 수도 있다...

부다페스트에는 0부터 $N-1$까지 번호가 붙은 N개의 명소가 있다. 명소들의 쌍들 중 일부는 *양방향* **길**로 연결되어 있다. 한 쌍의 명소들은 최대 1개의 길로 연결되어 있다. 운영진은 어떤 명소의 쌍들이 길로 연결되어 있는지 *모른다*.

길들의 연결망에 대해, 임의의 3개의 명소들을 보아도 그 명소들을 연결하는 길의 개수가 최소 δ개인 경우, 그 연결망의 **밀도**가 δ라고 말한다. 다르게 말하면, 임의의 3개의 명소 u, v, w에 대해 $(0 \leq u < v < w < N)$, 가능한 3개의 명소의 쌍들 $(u, v), (v, w), (u, w)$ 중 최소 δ 개가 연결되어 있다는 뜻이다.

운영진은 길들의 연결망의 밀도가 적어도 D라는 것을 알고 있다. D의 값은 3을 초과할 수 없음에 주의하라.

운영진은 부다페스트의 전화국에 **연락**해서 어떤 명소들 간에 존재하는 길들에 대한 정보를 얻을 수 있다. 각각의 연락을 할 때, 2개의 명소 배열 $[A[0], \ldots, A[P-1]]$와 $[B[0], \ldots, B[R-1]]$을 만들어야 한다. 배열에 저장된 명소들은 모두 달라야 한다. 즉,

- 모든 i, j $(0 \leq i < j < P)$에 대해서 $A[i] \neq A[j]$;
- 모든 i, j $(0 \leq i < j < R)$에 대해서 $B[i] \neq B[j]$;
- 모든 i, j $(0 \leq i < P, 0 \leq j < R)$에 대해서 $A[i] \neq B[j]$.

각 연락에 대해서 전화국은 A에 속한 명소들 중 하나와 B에 속한 명소들 중 하나를 연결하는 길이 하나라도 존재하는지의 여부를 알려준다.

즉, 전화국은 모든 i와 j $(0 \leq i < P, 0 \leq j < R)$에 대해 명소 $A[i]$와 $B[j]$가 길로 연결되어 있는지 확인하여 길로 연결된 경우가 적어도 하나 존재하면 `true`를 리턴하고, 그런 경우가 전혀 없는 경우 `false`를 리턴한다.

길이가 l인 **여행**은 서로 다른 명소들의 순서 $t[0], t[1], \ldots, t[l-1]$로 정의된다. 여행 $t[0], t[1], \ldots, t[l-1]$에서 각 $t[i]$와 $t[i+1]$ $(0 \leq i \leq l-2)$은 길로 연결되어야 한다. 길이가 l인 여행이 존재하고, 길이가 $l+1$ 이상인 여행은 존재하지 않을 때, 길이가 l인 여행은 **가장 긴 여행**이라고 불린다.

여러분이 할 일은 운영진을 도와 부다페스트의 명소들을 방문하는 가장 긴 여행들 중 하나를 찾는 것이다.

Implementation Details

다음의 함수를 작성해야 한다:

```
int[] longest_trip(int N, int D)
```

- N: 부다페스트의 명소 개수.
- D: 입력 연결망에서 보장되는 최소의 밀도.
- 이 함수는 가장 긴 여행 중 하나에 해당하는 명소들의 배열 $t = [t[0], t[1], \ldots, t[l-1]]$를 리턴해야 한다.
- 이 함수는 하나의 테스트 케이스에서 **여러 번** 호출될 수 있다.

위 함수에서 아래의 함수를 호출할 수 있다:

```
bool are_connected(int[] A, int[] B)
```

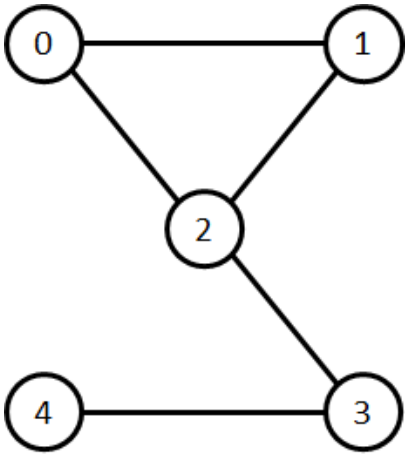
- A: 서로 다른 명소들을 포함하는 크기 1 이상인 배열.
- B: 서로 다른 명소들을 포함하는 크기 1 이상인 배열.
- A와 B의 명소들은 모두 달라야 한다.
- 이 함수는 A의 명소들 중 하나와 B의 명소들 중 하나를 연결하는 길이 적어도 하나 존재하는 경우 `true`를 리턴하고, 그러한 길이 하나도 없는 경우 `false`를 리턴한다.
- 이 함수는 `longest_trip`이 한번 호출될 때 마다 최대 $32\,640$번 호출될 수 있고, 테스트 케이스 전체에서 최대 $150\,000$번 호출될 수 있다.
- 테스트 케이스 전체에서, 배열 A와 B의 크기의 총 합은 $1\,500\,000$을 넘을 수 없다.

그레이더는 **적응적이지 않다**. 즉, 각 테스트 케이스에서 N과 D의 값들과 길로 연결된 명소들의 쌍들은 `longest_trip`이 호출되기 전에 미리 정해져 있다.

Examples

Example 1

$N = 5, D = 1$이고 다음 그림과 같이 길들이 존재하는 경우를 생각해 보자:



함수 `longest_trip`은 다음과 같이 호출된다:

```
longest_trip(5, 1)
```

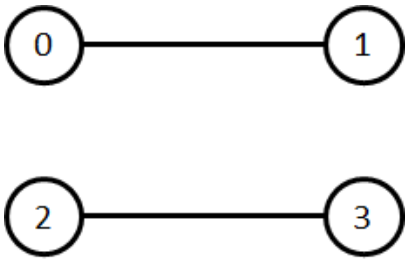
위 함수는 `are_connected`를 아래와 같이 호출한다.

호출	길로 연결된 명소 쌍	리턴 값
`are_connected([0], [1, 2, 4, 3])`	$(0,1), (0,2)$	`true`
`are_connected([2], [0])`	$(2,0)$	`true`
`are_connected([2], [3])`	$(2,3)$	`true`
`are_connected([1, 0], [4, 3])`	없음	`false`

네번째 호출 직후에 명소의 쌍들 $(1,4), (0,4), (1,3), (0,3)$ 중 어떤 쌍도 길로 연결되어 있지 않다는 것을 알게 된다. 연결망의 밀도가 최소 $D = 1$이므로 명소들 $0, 3, 4$ 중 쌍 $(3,4)$는 길로 연결되어 있다는 것을 알 수 있다. 비슷하게, 명소 0과 1도 길로 연결되어 있어야 한다.

이제, 길이가 5를 초과하는 여행은 존재할 수 없으므로, 여행 $t = [1,0,2,3,4]$이 가장 긴 여행이라는 것을 알 수 있다. 따라서, 함수 `longest_trip`은 $[1,0,2,3,4]$을 리턴할 수 있다.

$N = 4, D = 1$이고 길들이 다음 그림과 같은 경우를 생각해 보자:



함수 `longest_trip`은 다음과 같이 호출된다:

```
longest_trip(4, 1)
```

이 경우 가장 긴 여행의 길이는 2이다. 따라서, `are_connected`를 몇 차례 호출한 후, `longest_trip` 함수는 $[0,1], [1,0], [2,3], [3,2]$ 중 하나를 리턴할 수 있다.

Example 2

Subtask 0에는 $N = 256$인 연결망이 있는 테스트 케이스가 포함되어 있다. 이 테스트 케이스는 콘테스트 시스템에서 다운로드 받을 수 있는 패키지에 포함되어 있다.

Constraints

- $3 \leq N \leq 256$
- 하나의 테스트 케이스에서 `longest_trip`의 모든 호출에 대한 N의 합은 최대 $1\,024$이다.
- $1 \leq D \leq 3$

Subtasks

1. (5 points) $D = 3$
2. (10 points) $D = 2$
3. (25 points) $D = 1$. $l^\star$가 가장 긴 여행의 길이라고 하자. 함수 `longest_trip`은 길이 $l^\star$인 여행을 리턴하지 않아도 된다. 대신, 길이가 최소 $\left\lceil \frac{l^\star}{2} \right\rceil$인 여행을 리턴해야 한다.
4. (60 points) $D = 1$

Subtask 4에서 당신의 점수는 `longest_trip`이 한번 호출되었을 때 `are_connected` 함수가 몇번 호출되느냐에 따라 결정된다. 정수 q가 Subtask 4에 속한 모든 테스트 케이스에서 `longest_trip`이 한번 호출되었을 때 `are_connected` 함수가 호출된 최대 횟수라고 하자. Subtask 4의 점수는 아래 표와 같이 결정된다:

경우	점수
$2\,750 < q \leq 32\,640$	20
$550 < q \leq 2\,750$	30
$400 < q \leq 550$	45
$q \leq 400$	60

함수 `are_connected`의 호출이 규칙을 따르지 않거나 함수 `longest_trip`이 리턴한 배열이 올바르지 않은 경우가 한번이라도 있으면 해당 subtask의 점수는 0점이다.

Sample Grader

C가 시나리오의 개수라고 하자. 여기서, 시나리오란 `longest_trip`의 각각의 호출을 말한다. Sample grader는 아래의 형식으로 입력을 받는다:

- line 1: C

이후 각 C개의 시나리오가 다음과 같은 형태로 주어진다.

Sample grader는 아래의 형식으로 각 시나리오의 입력을 받는다:

- line 1: $N\ D$
- line $1+i$ ($1 \leq i < N$): $U_i[0]\ U_i[1]\ \ldots\ U_i[i-1]$

각 U_i ($1 \leq i < N$)는 크기 i인 배열이다. 이 배열들은 어떤 명소 쌍들이 길로 연결되어 있는지를 알려준다. 각 i, j ($1 \leq i < N, 0 \leq j < i$)에 대해:

- 명소 j와 i가 연결된 경우 $U_i[j]$의 값은 1이다;
- 명소 j와 i가 연결되지 않은 경우, $U_i[j]$의 값은 0이다.

각 시나리오에서 `longest_trip`을 호출하기 전에 Sample grader는 연결망의 밀도가 최소 D인지 확인한다. 이 조건이 만족되지 않는 경우 `Insufficient Density`를 프린트하고 종료한다.

Sample grader가 프로토콜이 위반된 경우를 발견하면 `Protocol Violation: <MSG>` 형식의 메시지를 출력한다. 여기서 `<MSG>`는 아래들 중 하나이다:

- `invalid array`: `are_connected`의 호출에서 A와 B 중 적어도 하나가
  - 빈 것이거나
  - 0부터 $N-1$까지의 값들 이외의 값을 가지고 있거나
  - 중복된 값을 가진다.
- `non-disjoint arrays`: `are_connected`의 호출에서, A와 B에 겹치는 원소가 있다.
- `too many calls`: `are_connected`의 호출 횟수가 `longest trip`의 호출 한번에서 $32\,640$를 초과하거나, 테스트 케이스 전체에서 $150\,000$를 초과한다.
- `too many elements`: 하나의 테스트 케이스에서 `are_connected` 호출 모두에서 전달된 명소의 총 갯수가 $1\,500\,000$을 초과한다.

위의 경우들이 아닐 때, `longest_trip`에서 리턴된 배열의 원소들이 $t[0], t[1], \ldots, t[l-1]$라고 하자. Sample grader는 해당하는 시나리오에 대해 다음을 프린트한다:

- line 1: l
- line 2: $t[0]$ $t[1]$ $\ldots$ $t[l-1]$
- line 3: 해당 시나리오에서 `are_connected`의 호출 횟수

마지막으로 Sample grader는 다음을 프린트한다:

- line $1 + 3 \cdot C$: `longest_trip`의 모든 호출에서 `are_connected` 호출 횟수의 최대값