섬 (island)

IOI 나라는 특이하게도 정N각형 모양의 섬에 세워졌다. N개의 각 꼭짓점에 해당하는 위치마다 **지역**이 있으며, 이 지역들에는 시계 방향으로 순서대로 $0, 1, \ldots, N-1$ 의 번호가 붙어 있다. IOI 나라의 도로망은 다음과 같은 두 종류의 도로로 이루어진다:

- **해변 도로**: 해변 도로는 정N각형의 인접한 꼭짓점에 해당하는 지역 사이를 연결하는 N개의 도로이다. 다시 말해, 모든 정수 $0 \le i \le N-2$에 대해 i번 지역과 $i+1$번 지역을 잇는 도로가 존재하며, $N-1$번 지역과 0번 지역을 잇는 도로가 존재한다.
- **육지 도로**: 해변 도로로 직접 연결되어 있지 않은 두 지역을 선분 형태로 연결하는 육지 도로들이 총 $N-3$개 존재한다. 이 때, 각 육지 도로들은 끝점을 제외하고는 서로 만나지 않는다. 즉, 정N각형에서 교차하지 않는 서로 다른 대각선 $N-3$개에 해당한다.

한편, K개의 지역을 잇는 어떤 도로망에 대해, 도로의 집합 T가 다음 조건을 만족할 때 T를 **트리**라고 한다.

- $|T| = K-1$
- T에 포함된 도로만 이용해서 모든 지역 사이를 이동할 수 있다.

트리는 모든 지역을 연결하므로 운송에서 매우 중요한 역할을 차지한다. 하지만, 트리의 도로를 사용할 수 없을 때 이용할 수 있는 또 다른 트리가 있다면 안정성에 크게 도움이 될 것이다. 이에 도로망에 두 트리 T_1와 T_2가 존재하여 $T_1 \cap T_2 = \emptyset$를 만족할 때, 즉 어떠한 도로도 겹치지 않는 두 트리가 존재할 때, 그 도로망을 **좋은 도로망**이라고 정의한다.

IOI 나라에서는 다음과 같이 새로운 지역과 도로를 건설하는 방안을 통해 좋은 도로망을 구축하고자 한다.

- **지역 건설**: 지역 a, b, c에 대해 a와 b 사이, b와 c 사이, c와 a 사이를 직접 잇는 도로가 모두 존재할 때, 세 지역이 이루는 삼각형의 내심에 새로운 지역 d를 만들고, a와 d 사이, b와 d 사이, c와 d 사이를 도로로 연결한다. 새로운 지역 d의 번호는 N부터 순서대로 붙여진다. 동일한 세 지역에 대해서 지역 건설을 두 번 이상 할 수 없다. 다시 말해, 지역 건설에서 사용한 집합 $\{a, b, c\}$ 는 매 건설마다 서로 달라야 한다.

IOI 나라에서는 **지역 건설**을 여러 번 할 수 있지만, 가능한 적은 횟수의 지역 건설을 통해 겹치지 않는 두 트리가 존재하는 좋은 도로망으로 바꾸고자 한다. 좋은 도로망이 되기 위해서는 기존의 N개 지역뿐만 아니라 새로 건설된 지역도 연결하는 겹치지 않는 두 트리가 존재해야 함에 주의하라. 여러분은 IOI 나라를 도와 도로망 문제를 해결해야 한다. **지역 건설의 횟수를 최소화하지 않아도 부분 점수를 받을 수 있음에 유의하라**.

함수 목록 및 정의

여러분은 아래 함수들을 구현해야 한다.

```
void construct_two_trees(int N, std::vector<int> U, std::vector<int> V)
```

- U, V: 크기가 $N-3$인 정수 배열. 모든 정수 $0 \le i \le N-4$에 대해, $U[i]$번 지역과 $V[i]$번 지역을 잇는 육지 도로가 존재한다.
- 이 함수는 단 한 번만 호출되며, 여러분은 이 함수 내에서 추후에 정의될 `add_vertex` 함수를 호출하여 지역을 건설하고, 도로를 공유하지 않는 두 트리를 찾아 `report` 함수를 호출하여야 한다.

```
int add_vertex(int a, int b, int c)
```

- 이 함수는 지역 a, b, c 에 대한 지역 건설을 나타낸다.
- 함수 실행 이전에 지역 a, b, c 중 임의의 두 지역을 뽑아도 도로로 직접 연결되어 있어야 한다.
- 동일한 세 지역에 대해 이 함수를 두 번 이상 호출하지 않아야 한다. 다시 말해, 지역 건설에서 사용한 집합 $\{a, b, c\}$ 는 매 호출마다 서로 달라야 한다.
- 이 함수는 새로 건설된 지역의 번호를 반환한다. 즉, 이 함수가 j번째로 실행될 때, $N-1+j$를 반환한다.
- 이 함수는 `report` 함수가 한 번이라도 호출된 이후에는 호출되지 않아야 한다.

```
void report(std::vector<std::array<int, 2> > tree)
```

- 이 함수는 도로망에서 트리를 찾아 보고하는 함수이다.
- 이 함수는 `construct_two_trees` 함수에서 모든 `add_vertex` 함수의 호출이 종료된 후에 **정확히** 2회 호출되어야 한다.
- 파라미터 `tree`의 각 원소는 도로가 잇는 두 지역의 번호를 담고 있는 `std::array<int, 2>`여야 한다. 이 때, 두 지역의 번호의 순서는 어떻게 하여도 상관 없다.
- 두 번의 호출이 `report`(T_1), `report`(T_2)일 때, T_1과 T_2는 도로를 공유하지 않아야 하며 $1 \le k \le 2$에 대해 T_k의 간선들만을 통해 `add_vertex`로 생긴 지역들을 포함해 모든 지역들을 오고갈 수 있어야 한다.

제출하는 소스 코드의 어느 부분에서도 입출력 함수를 실행해서는 안 된다.

제약 조건

- $3 \leq N \leq 200\,000$
- 모든 $0 \leq i \leq N-4$에 대해:
  - $0 \leq U[i], V[i] \leq N-1$
  - $U[i] \neq V[i]$
- 주어지는 U와 V는 지문 상의 육지 도로 조건을 만족한다.

부분문제

1. (6점)
   - $N \leq 5$
2. (8점)
   - 자신을 제외한 모든 지역과 직접 도로로 연결된 지역이 존재한다.
3. (14점)
   - 초기 상태에서 지역 건설이 가능한 모든 지역 쌍 (a, b, c)에 대해, 서로를 잇는 3개의 도로 중 해변 도로가 1개 이상 존재한다.
4. (21점)
   - $N \leq 5\,000$
5. (51점)
   - 추가적인 제약 조건이 없다.

`construct_two_trees` 함수가 문제를 올바르게 해결하였을 때, `add_vertex`의 호출 횟수가 최솟값보다는 크지만 N을 넘지 않는 경우 **각 부분문제에서 점수의 40%를 획득한다.** `add_vertex`를 N번보다 많이 호출하는 경우 점수를 얻지 못한다. 제약 조건 아래에서 `add_vertex`를 N번 이하로 호출하여 좋은 도로망을 구성할 수 있음은 증명할 수 있다.

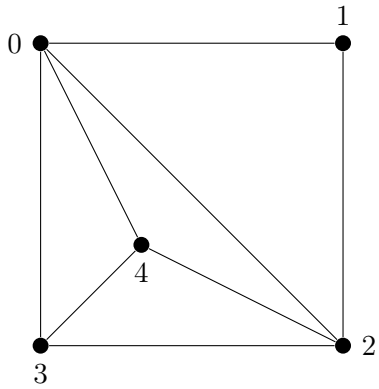
예제 1

$N = 4$, $U = [0]$, $V = [2]$인 경우를 생각해 보자.

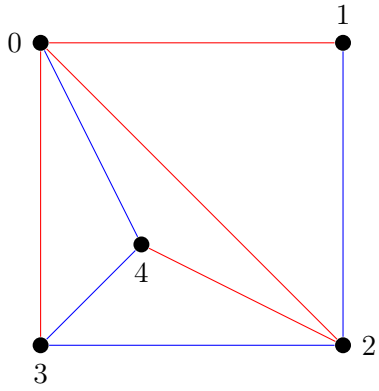
그레이더는 다음과 같이 함수를 호출한다.

```
construct_two_trees(4, [0], [2])
```

그 후 `add_vertex(0, 2, 3)`을 호출했을 때, 도로망은 다음과 같은 상태이다.



그 후 `report([[0,1],[0,2],[0,3],[4,2]])`와 `report([[4,0],[3,4],[2,3],[2,1]])`을 차례로 호출하면 이는 올바르게 두 트리를 report한 실행이다. 호출한 두 트리를 각각 빨간색과 파란색으로 나타내면 아래 그림과 같다.



Sample grader

Sample grader는 아래와 같은 형식으로 입력을 받는다.

- Line 1: N
- Line $2+i$ ($0 \le i \le N-4$): $U[i]$ $V[i]$

Sample grader는 다음과 같은 형식으로 출력한다.

먼저, `report` 함수가 호출될 때마다 그레이더는 다음과 같이 출력한다.

- Line 1: 정수 k. 함수 `report`의 k번째 호출임을 뜻한다.
- Line 2: 트리의 도로 수 M
- Line $2+i$ ($1 \le i \le M$): 트리의 i번째 도로의 양 끝점 번호 $A[i]$ $B[i]$

그 후, `construct_two_trees` 함수의 실행이 모두 종료된 뒤 그레이더는 `add_vertex` 함수의 호출에 대한 정보를 다음과 같이 출력한다.

- Line 1: `add_vertex` 함수의 총 호출 횟수 K
- Line $1 + i$ ($1 \leq i \leq K$): `add_vertex` 함수의 i번째 호출의 파라미터 $A[i]$ $B[i]$ $C[i]$

Sample grader는 실제 채점에서 사용하는 그레이더와 다를 수 있음에 유의하라.