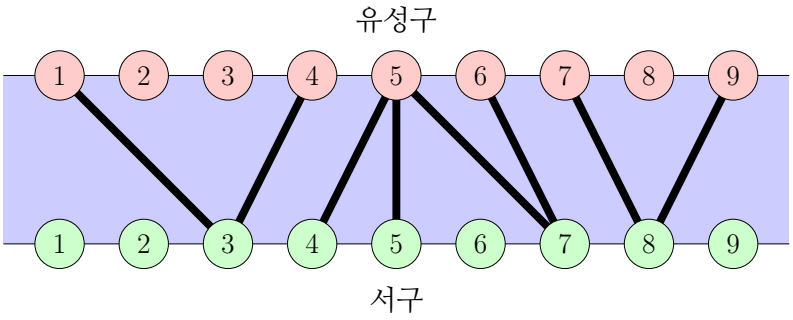
다리 찾기

모두 알다시피, 카이스트 근처에는 갑천이라는 무한히 긴 강이 있어 유성구와 서구의 경계 역할을 하고 있다. 이 강의 모습은 2차원 평면 위에 평행한 두 직선을 그리고, 그 사이에 물을 채워 나타낼 수 있다. 편의상 아래 그림에서 강 위의 구역을 유성구, 강 아래의 구역을 서구라고 하자.



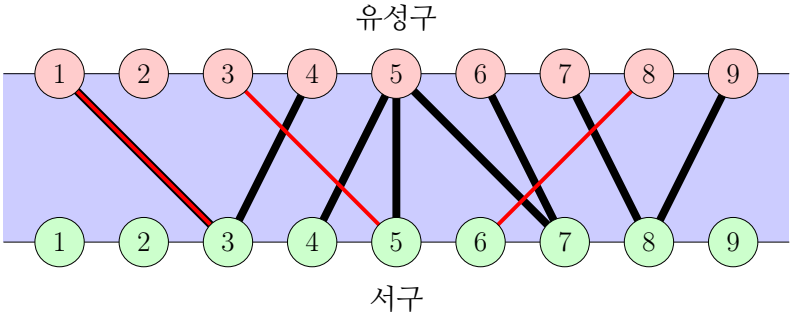
강의 유성구 쪽 경계와 서구 쪽 경계에는 각각 N개의 지점이 있으며, 인접한 두 지점 간의 간격은 정확히 100m이다. 각 지점에는 1 이상 N 이하의 번호가 왼쪽에서부터 오른쪽 순서대로 붙어 있다.

강에는 유성구와 서구를 잇는 다리가 0개 이상 놓여 있다. 각 다리는 유성구 쪽 경계 위의 한 지점과 서구 쪽 경계 위의 한 점을 잇는 선분으로 생각할 수 있다. 그림과 같이, 각 지점에 대해 그 지점을 연결하는 다리가 없을 수도 (유성구 쪽 2번 지점 등), 1개일 수도 (서구 쪽 5번 지점 등), 2개 이상일 수도 (유성구 쪽 5번 지점 등) 있다. 하지만, 서로 다른 두 다리가 양 끝 지점을 제외한 점에서 교차하지는 않는다.

당신은 갑천 위에 있는 모든 다리를 지도에 표시하려고 한다. 이를 위해서는 각 다리가 연결하고 있는 유성구 쪽 지점의 번호와 서구 쪽 지점의 번호를 알아야 할 것이다. 하지만 날씨가 좋지 않아 육안으로 다리가 어디에 놓여 있는지 관찰하기가 쉽지 않았다. 그래서 당신은 배를 타고 강 위를 돌아다니면서 필요한 모든 정보를 알아내기로 하였다. 당신은 아래와 같은 작업을 **최대 100 000번** 반복할 수 있다.

- 먼저 유성구 쪽의 한 지점 a와 서구 쪽의 한 지점 b를 고른다.
- a에서 출발하여, 배를 타고 두 지점을 잇는 직선 경로를 통해 b로 이동한다.
- 이동하면서, 출발 지점과 도착 지점을 제외하고 배가 다리 바로 아래를 지나는 횟수를 센다. 즉, 배의 이동 경로가 나타내는 선분과, 갑천 위에 있는 다리들이 만드는 선분들의 교점의 개수를 세는 것이다. 갑천 위에 a와 b를 잇는 다리가 있다면 교점의 개수가 무한하므로, ∞번 만나는 것으로 간주된다.

예를 들어, 위 그림에서 $a = 3, b = 5$로 두면, 배가 유성구 쪽 3번 지점과 서구 쪽 5번 지점을 잇는 직선 경로를 따라 이동하게 되며, 배가 다리 밑을 지나는 횟수는 2번이다. $a = 8$, $b = 6$일 때에는 3번이며, $a = 1, b = 3$일 때에는 ∞번이다.



당신은 이러한 일을 반복하여 얻은 정보만을 가지고 모든 다리가 어떤 지점을 연결하고 있는지 정확히 알아내야 한다. 다리가 있는 위치에 다리가 없다는 결과를 내거나, 다리가 없는 위치에 다리가 있다는 결과를 낸다면 오답 판정을 받게 된다.

상호 작용(Interaction)

먼저 한 개의 줄에 각 구역에 있는 지점의 개수 N $(1 \leq N \leq 50\,000)$이 주어진다.

배를 타려면, '? a b'를 한 개의 줄에 출력해야 한다. 여기서 a는 유성구 쪽 지점의 번호이고, b는 서구 쪽 지점의 번호이다. a와 b는 $1 \leq a, b \leq N$를 모두 만족하는 정수여야 하며, 그렇지 않을 시 오답 판정을 받게 된다. 그 다음, a와 b를 잇는 직선 경로로 이동하는 중에 다리 밑을 지나는 횟수가 한 개의 줄에 주어진다. 단, 그 횟수가 ∞라면 '`-1`'이 대신 주어진다. 이 작업은 최대 100 000번까지만 할 수 있고, 100 001번 이상 한다면 오답 판정을 받게 된다.

모든 다리의 정확한 위치를 알아냈다면, 우선 '`!`'만 있는 한 개의 줄을 출력해야 한다. 다리의 개수를 k라고 하면, 바로 다음 줄에 k를 출력하고, 그 다음에 k개의 줄을 출력해야 한다. 각 줄에는 두 개의 정수 p와 q $(1 \leq p, q \leq N)$를 출력해야 하는데, 이는 유성구 쪽 p번 지점과 서구 쪽 q번 지점을 잇는 다리가 존재한다는 것을 의미한다. k가 실제 다리의 개수와 동일하고, 출력한 다리의 정보가 실제와 동일하다면 정답 판정을 받게 된다. 다리 위치의 출력 순서는 관계 없다. $k = 0$일 수 있음에 유의해야 한다. 출력이 끝나면 즉시 프로그램을 종료해야 한다.

매 줄을 출력할 때마다 C/C++에서는 '`fflush(stdout);`', C++에서는 '`std::cout << std::flush;`', Java에서는 '`System.out.flush();`'를 통해 표준 출력 버퍼를 비워줘야 한다.

여러분의 구현을 돕기 위해, 채점 시스템에서 C/C++/Java 예시 코드를 내려받을 수 있다. 이미 구현되어 있는 '`sail`' 함수와 '`report`' 함수를 그대로 사용하면 된다.

채점 방식

1. (7점) $N \leq 300$
2. (13점) 다리는 최대 한 개이다.
3. (9점) $N \leq 7\,500$
4. (13점) 각 지점은 최대 한 개의 다리와만 연결되어 있다.
5. (34점) $N \leq 25\,000$
6. (24점) 추가 제약 조건 없음.

참고로, 부분 문제 간의 포함 관계는 1번 $\subset$ 3번 $\subset$ 5번 $\subset$ 6번, 2번 $\subset$ 4번 $\subset$ 6번이다. 입력 데이터가 2번, 4번 부분문제에 해당하는지 알 수 있는 방법이 없으므로, 생각한 모든 풀이에 대한 점수를 받기 위해서는 아래와 같은 방식으로 구현하는 것을 추천한다.

```
if(N <= (300, 7500, 25000 중 하나)) {
  // 1번, 3번 또는 5번 부분문제의 조건을 만족한다고 가정하고 코드 작성
}else {
  // 2번 또는 4번 부분문제의 조건을 만족한다고 가정하고 코드 작성
}
```

참고

문제 페이지 하단에서 내려받을 수 있는 `helper.zip`에는 아래 파일들이 들어 있다.

- `bridges.(c|cpp|java)`: 예시 코드이다. "`/* IMPLEMENT HERE */`" 사이의 부분을 구현하면 된다. 모든 다리가 유성구 쪽 i번 지점과 서구 쪽 i번 지점을 연결한다고 가정한 틀린 코드가 구현되어 있다.
- `simulator.py`: 실제 채점 시스템처럼 프로그램과 입출력을 주고 받는 Python 스크립트이다. Windows 10, Ubuntu 18.04 (WSL), Mac OS X의 Python 2.7, 3.6에서 테스트되었으며, Python 버전이 다르면 실행이 제대로 되지 않을 수 있다.
  - 간단한 사용법: 명령줄에 '`python simulator.py -h`'를 입력하면 사용 가능한 옵션들을 볼 수 있다.
    - `--input INPUT`: 이 옵션을 사용하면 스크립트가 `INPUT` 파일에서 테스트 데이터를 읽는다. 입력 형식은 아래 '`example.in`' 부분을 참고한다. 입력 파일에서 다리끼리 교차하는지의 여부, 두 지점을 잇는 다리가 여러 개인지의 여부 등은 스크립트에서 검사한다.
    - `-l`, `--log`: 이 옵션을 켜면, 스크립트와 여러분이 작성한 프로그램 사이의 입출력 경과를 모두 출력한다.
    - `--limit LIMIT`: 배를 타는 횟수 제한이다. 명시하지 않았을 때의 기본값은 문제에 제시한 것과 같은 100 000이다.
    - 모든 옵션을 입력한 뒤, 마지막에 코드를 실행하는 명령어를 입력해야 한다. 명령어를 따옴표로 감싸면 안 된다.
  - 예시:
    - `python simulator.py --limit 100 -l ./bridges`
    - `python simulator.py --input input.txt java bridges`
  - **주의**: **이 스크립트는 굉장히 느리다.** 다리의 개수를 k라고 할 때, 이 스크립트는 입력에서 다리끼리 교차하지 않는지 확인하는 데에 $O(k^2)$ 시간을 들이고, 배를 탈 때마다 $O(k)$의 시간을 들여 교점의 개수를 센다. k가 커지면 스크립트의 실행 시간이 지나치게 길어질 수 있으므로, k를 크게 하는 것보다 '`--limit`' 옵션을 사용하여 항해 횟수에 제한을 두는 것을 더 권장한다.
- `example.in`: 아래의 형식으로 갑천의 다리에 대한 정보를 스크립트에게 제공한다.
  - 1번째 줄: N
  - 2번째 줄: k (다리의 수)
  - $[3, k+2]$번째 줄: p_j q_j (유성구 쪽 p_j번 지점과 서구 쪽 q_j번 지점을 잇는 다리가 있다는 뜻이다.)