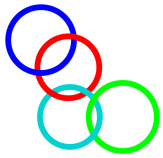
낙하산 고리들

레오나르도 다 빈치의 *코덱스 아틀란티쿠스* 에는 최초로 설계한 꽤 정교한 낙하산이 설명되어 있는데, 이 다 빈치의 낙하산은 피라미드 모양의 나무 구조를 통해 열리는 우산모양의 리넨 옷감으로 설계되어있다.

고리 구조

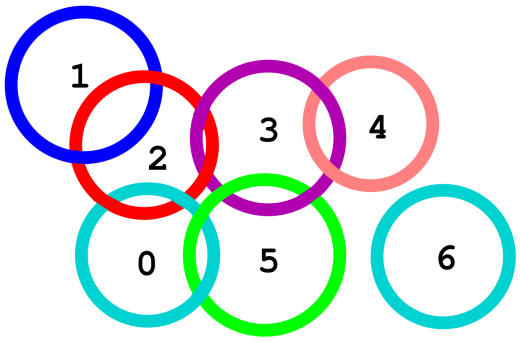
스카이다이버 아드리안 니콜라스는 500년 이상 오래된 다 빈치의 디자인을 시험해보았다. 이를 위해, 사람의 몸을 다 빈치의 낙하산에 묶기 위한 최신식 초경량 구조를 사용하였다. 우리는 리넨 옷감에도 걸 수 있는 고리 구조를 사용하고자 한다. 각각의 고리는 유연하고 강한 소재로 만들어졌다. 모든 고리들은 열고 닫을 수 있으므로, 다른 고리들과 쉽게 연결될 수 있다. 체인은 고리 구조의 특별한 형태를 말한다. 아래에 있는 그림처럼, 각각의 고리들이 (최대 두 개의) 인접한 고리들과 차례대로 연결되어 있고, 시작과 끝이 있는 고리 구조는 체인이 된다. (시작과 끝에 있는 고리들은 각각 최대 하나의 다른 고리들과 연결된다.) 특히, 고리 하나도 마찬가지로 체인이다.



하나의 고리가 세 개 이상의 다른 고리들과 연결된 고리 구조의 형태도 당연히 가능하다. 이러한 고리 구조에서, 어떤 하나의 고리를 열어서 없앤 후에 남은 다른 고리들이 모두 체인의 형태를 만들 때 (혹은 고리가 하나도 남지 않았을 때), 그 고리를 *중요한* 고리라고 한다. 다시 말해, 중요한 고리를 제거하면, 체인만 남게 된다.

예제

다음의 그림과 같이, 0번부터 6번까지 번호가 매겨진 7개의 고리를 생각해보자. 이 고리 구조에는 2개의 중요한 고리가 있다. 그 중 하나는 2번인데, 이 고리를 없애면 남은 고리들이 [1], [0, 5, 3, 4], 그리고 [6]의 체인이 된다. 다른 하나는 3번인데, 이 고리를 없애면 남은 고리들이 [1, 2, 0, 5], [4], 그리고 [6]의 체인이 된다. 만약 그 이외의 다른 고리들을 없애면, 체인이 아닌 것이 남아있게 된다. 예를 들어, 5번 고리를 없애면 [6]은 체인이 되지만, 0번, 1번, 2번, 3번, 그리고 4번 고리로 만들어진 고리 구조는 체인이 아니다.



해야할 일

여러분이 할 일은 입력으로 주어지는 고리 구조에서, 중요한 고리의 개수를 세는 프로그램을 작성하는 것이다.

처음에는, 서로 연결되지 않은 고리들의 개수가 주어진다. 다음에는, 고리 구조가 만들어진다. 입력 중에 언제든지, 여러분은 현재 고리 구조에서 중요한 고리의 개수를 답해야 된다. 이를 위해 여러분은 3개의 함수를 구현해야한다.

- `Init`(N) - 처음에, 오직 한번 실행된다. 인자 N의 의미는 0부터 N - 1까지 번호가 붙은 N개의 고리들이 있다는 것을 의미한다.
- `Link`(A, B) - A번 고리와 B번 고리를 서로 연결함을 의미한다. A와 B는 다르고, A번 고리와 B번 고리는 현재 연결되어 있지 않다는 것을 보장한다. 이를 제외하고, A번 고리와 B번 고리 사이에는 다른 제약조건이 없다. (특히, 물리적인 조건과는 관계가 없다.) 또한, `Link`(A, B)와 `Link`(B, A)는 서로 같은 의미이다.
- `CountCritical`() — 현재의 고리 구조의 형태에서, 중요한 고리의 개수를 계산하여 반환한다.

예제

N = 7 개의 고리가 주어질 때를 생각해보자. 다음의 표는 가능한 호출 순서에 대해서, 각 호출에서 얻은 결과를 보여준다.

함수 호출	반환값
`Init`(7)	
`CountCritical`()	7
`Link`(1, 2)	
`CountCritical`()	7
`Link`(0, 5)	
`CountCritical`()	7
`Link`(2, 0)	
`CountCritical`()	7
`Link`(3, 2)	
`CountCritical`()	4
`Link`(3, 5)	
`CountCritical`()	3
`Link`(4, 3)	
`CountCritical`()	2

서브태스크 1 [20점]

- $N \leq 5{,}000$.
- `CountCritical` 함수는 다른 모든 함수들의 호출이 끝난 후, 마지막에 한 번 호출된다. `Link` 함수는 최대 5,000번 호출된다.

서브태스크 2 [17점]

- $N \leq 1{,}000{,}000$.
- `CountCritical` 함수는 다른 모든 함수들의 호출이 끝난 후, 마지막에 한 번 호출된다. `Link` 함수는 최대 1,000,000번 호출된다.

서브태스크 3 [18점]

- $N \leq 20{,}000$.
- `CountCritical` 함수는 최대 100번 호출된다. `Link` 함수는 최대 10,000번 호출된다.

서브태스크 4 [14점]

- $N \leq 100{,}000$.
- `CountCritical` 함수와 `Link` 함수가 둘 다 합쳐서 최대 100,000번 호출된다.

서브태스크 5 [31점]

- $N \leq 1{,}000{,}000$.
- `CountCritical` 함수와 `Link` 함수가 둘 다 합쳐서 최대 1,000,000번 호출된다.

구현 세부 사항

`rings.c`, `rings.cpp` 혹은 `rings.pas` 중 오직 하나의 파일을 제출해야 한다. 제출하는 파일에는 다음 함수들의 선언에 맞는 프로그램이 구현되어있어야 한다.

C/C++ 프로그램

```
void Init(int N);
void Link(int A, int B);
int CountCritical();
```

Pascal 프로그램

```
procedure Init(N : LongInt);
procedure Link(A, B : LongInt);
function CountCritical() : LongInt;
```

이 부프로그램들은 문제에서 기술한대로 동작하여야 한다. 물론, 다른 부프로그램의 구현 내용은 자유롭게 하면 된다. 제출한 프로그램은 어떠한 방식으로든 표준 입/출력뿐만 아니라, 다른 어떠한 파일과도 상호작용을 하여서는 안된다.

grader 예시

주어지는 grader에서는 다음의 형식에 맞는 입력을 받아들인다.

- 1번째 줄: N, L;
- 2, ..., L + 1번째 줄:
  - `CountCritical` 함수를 호출하기 위해서는 -1;
  - `Link` 함수를 호출하기 위해서는 인자 A, B.

주어지는 grader에서는 `CountCritical`의 모든 결과를 출력해준다.