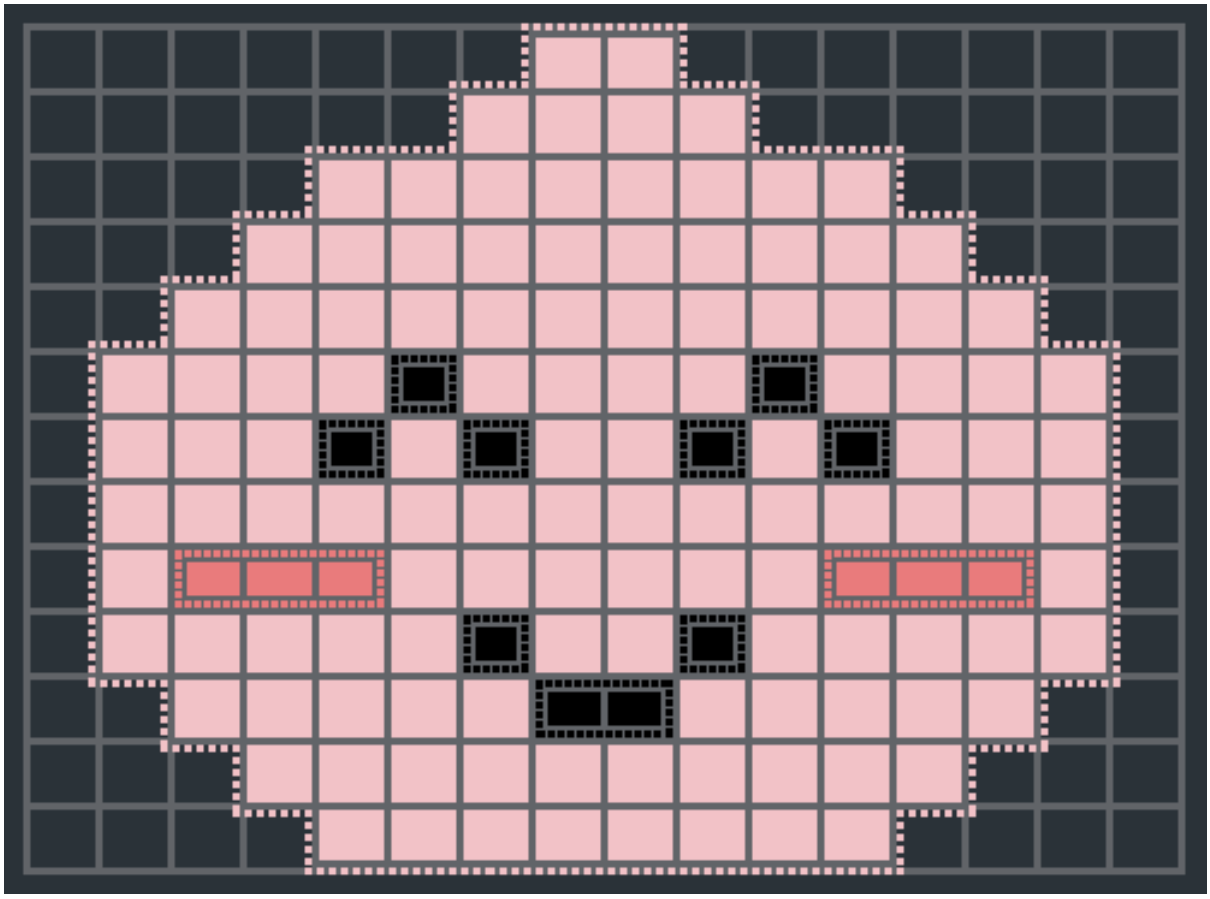
CODE FESTIVAL

문제 1

카카오프렌즈 컬러링북

출판사의 편집자인 어피치는 네오에게 컬러링북에 들어갈 원화를 그려달라고 부탁하여 여러 장의 그림을 받았다. 여러 장의 그림을 난이도 순으로 컬러링북에 넣고 싶었던 어피치는 영역이 많으면 색칠하기가 까다로워 어려워진다는 사실을 발견하고 그림의 난이도를 영역의 수로 정의하였다. (영역이란 상하좌우로 연결된 같은 색상의 공간을 의미한다.)

그림에 몇 개의 영역이 있는지와 가장 큰 영역의 넓이는 얼마인지 계산하는 프로그램을 작성해보자.



위의 그림은 총 12개 영역으로 이루어져 있으며, 가장 넓은 영역은 어피치의 얼굴면으로 넓이는 120이다.

CODE FESTIVAL

입력 형식

입력은 그림의 크기를 나타내는 `m`과 `n`, 그리고 그림을 나타내는 `m × n` 크기의 2차원 배열 `picture`로 주어진다. 제한조건은 아래와 같다.

- `1 <= m, n <= 100`
- `picture`의 원소는 `0` 이상 `2^31 - 1` 이하의 임의의 값이다.
- `picture`의 원소 중 값이 `0`인 경우는 색칠하지 않는 영역을 뜻한다.

출력 형식

리턴 타입은 원소가 두 개인 정수 배열이다. 그림에 몇 개의 영역이 있는지와 가장 큰 영역은 몇 칸으로 이루어져 있는지를 리턴한다.

예제 입출력

m	n	picture	answer
6	4	[[1, 1, 1, 0], [1, 2, 2, 0], [1, 0, 0, 1], [0, 0, 0, 1], [0, 0, 0, 3], [0, 0, 0, 3]]	[4, 5]

예제에 대한 설명

예제로 주어진 그림은 총 4개의 영역으로 구성되어 있으며, 왼쪽 위의 영역과 오른쪽의 영역은 모두 `1`로 구성되어 있지만 상하좌우로 이어져있지 않으므로 다른 영역이다. 가장 넓은 영역은 왼쪽 위 `1`이 차지하는 영역으로 총 5칸이다.

문제 2

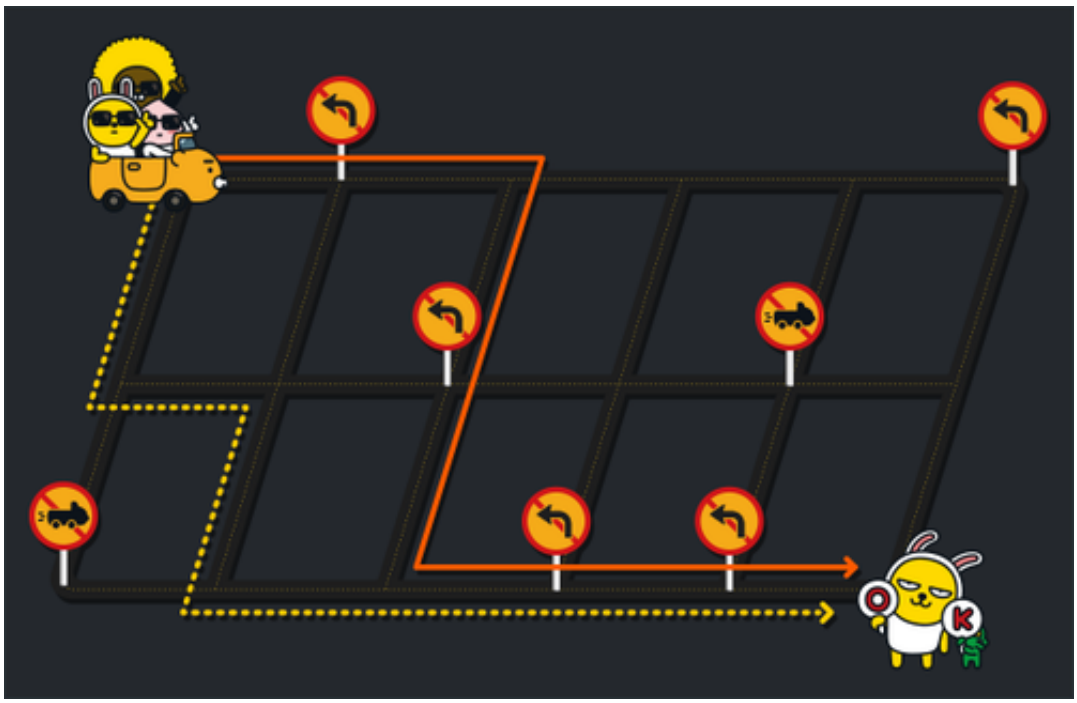
보행자 천국

카카오내비 개발자인 제이지는 시내 중심가의 경로 탐색 알고리즘 개발 업무를 담당하고 있다. 최근 들어 보행자가 자유롭고 편리하게 걸을 수 있도록 보행자 중심의 교통 체계가 도입되면서 도심의 일부 구역은 자동차 통행이 금지되고, 일부 교차로에서는 보행자 안전을 위해 좌회전이나 우회전이 금지되기도 했다. 복잡해진 도로 환경으로 인해 기존의 경로 탐색 알고리즘을 보완해야 할 필요가 생겼다.

도시 중심가의 지도는 `m × n` 크기의 격자 모양 배열 `city_map`으로 주어진다. 자동차는 오른쪽 또는 아래 방향으로 한 칸씩 이동 가능하다.

`city_map[i][j]`에는 도로의 상황을 나타내는 값이 저장되어 있다.

- `0`인 경우에는 자동차가 자유롭게 지나갈 수 있다.
- `1`인 경우에는 자동차 통행이 금지되어 지나갈 수 없다.
- `2`인 경우는 보행자 안전을 위해 좌회전이나 우회전이 금지된다. (왼쪽에서 오던 차는 오른쪽으로만, 위에서 오던 차는 아래쪽으로만 진행 가능하다)



도시의 도로 상태가 입력으로 주어졌을 때, 왼쪽 위의 출발점에서 오른쪽 아래 도착점까지 자동차로 이동 가능한 전체 가능한 경로 수를 출력하는 프로그램을 작성하라. 이때 가능한 경로의 수는 컴퓨터가 표현할 수 있는 정수의 범위를 넘어설 수 있으므로, 가능한 경로 수를 `20170805`로 나눈 나머지 값을 출력하라.

입력 형식

입력은 도시의 크기를 나타내는 `m`과 `n`, 그리고 지도를 나타내는 2차원 배열 `city_map`으로 주어진다. 제한조건은 아래와 같다.

- `1 <= m, n <= 500`
- `city_map`의 크기는 `m × n`이다.
- 배열의 모든 원소의 값은 `0`, `1`, `2` 중 하나이다.
- 출발점의 좌표는 `(0, 0)`, 도착점의 좌표는 `(m - 1, n - 1)`이다.
- 출발점과 도착점의 `city_map[i][j]` 값은 `0`이다.

출력 형식

출발점에서 도착점까지 이동 가능한 전체 경로의 수를 `20170805`로 나눈 나머지를 리턴한다.

예제 입출력

m	n	city_map	answer
3	3	[[0, 0, 0], [0, 0, 0], [0, 0, 0]]	6
3	6	[[0, 2, 0, 0, 0, 2], [0, 0, 2, 0, 1, 0], [1, 0, 0, 2, 2, 0]]	2

예제에 대한 설명

첫 번째 예제는 모든 도로가 제한 없이 통행 가능한 경우로, 가능한 경우의 수는 6가지이다.
두 번째 예제는 문제 설명에 있는 그림의 경우이다. 가능한 경우의 수는 빨간 실선과 노란 점선 2가지뿐이다.

CODE FESTIVAL

문제 3

브라이언의 고민

카카오스토리의 개발자 브라이언에게 최근 고민이 생겼다. 하루에도 수백만 명이 사용하는 서비스답게 사람들이 많이 보는 글에 광고성 댓글을 달아 불쾌감을 유발하는 사용자가 증가하고 있는데, 신고를 받은 글이 광고글인지를 운영자가 판단하여 차단하는 시스템으로는 빠르게 늘어나는 광고글을 처리하기 어렵기 때문이다. 그래서 브라이언은 신고된 글이 광고글인지를 자동으로 판단하는 시스템을 만들었다. 이제 사용자가 광고글을 보고 신고하면 그 글이 광고글로 판단된 경우 자동으로 차단된다! 드디어 깨끗한 카카오스토리를 만들었다는 기쁨도 잠시, 광고글을 올리는 사람들이 자동 차단 시스템을 회피할 수 있는 방법을 찾기 시작했고, 얼마 지나지 않아 광고 문구 사이에 특수문자를 넣으면 차단되지 않는다는 점이 알려지게 되었다. 즉, 아래와 같은 식으로 작성하면 광고글 차단이 적용되지 않는다.

> ♚프☆렌☆즈☆레☆이☆싱♚★사전예약★진행중
> $지금$예약시♛이모티콘♛100%※증정※
> ★라이언★카트♨전원♨획@득@기@회
> 즉시이동 http://...

생각지 못한 광고글 패턴에 당황하던 브라이언은 광고글이 일정한 규칙에 의해 만들어진다는 사실을 알게 되었는데, 그 규칙은 다음과 같다.
(아래 설명 및 그 이후의 내용에서 영문 대문자는 원래 문구, 소문자는 특수기호를 의미한다.)

- 광고글은 원래 문구에 다음 규칙을 적용하여 만들 수 있다.
  - (규칙 1) 특정 단어를 선택하여 글자 사이마다 같은 기호를 넣는다. ex) `HELLO` -> `HaEaLaLaO`
  - (규칙 2) 특정 단어를 선택하여 단어 앞뒤에 같은 기호를 넣는다. ex) `WORLD` -> `bWORLDb`
  - 위의 두 가지 규칙은 한 단어에 모두 적용될 수 있지만 같은 규칙은 두 번 적용될 수 없다.
  - 한 번 쓰인 소문자(특수기호)는 다시 쓰일 수 없다.
- 마지막으로 원래 문구에 있던 공백을 제거한다.

위의 규칙에 따라, `HELLO WORLD`는 다음의 광고 문구로 변환될 수 있다.

- `HELLOWORLD` (기호 삽입 없이 마지막 규칙인 공백 제거만 적용되었다.)
- `HaEaLaLaObWORLDb` (첫 번째 단어에는 규칙 1이, 두 번째 단어에는 규칙 2가 적용되었다.)
- `aHbEbLbLbOacWdOdRdLdDc` (모든 단어에 모든 규칙이 적용되었다.)

단, 아래의 문구는 올바르게 변환된 광고문구가 아니다.

- `aHELLOa bWORLDb` (공백이 제거되어야 한다.)
- `HaEaLaLObWORLDb` (규칙 1은 단어의 모든 글자 사이에 적용되어야 한다. 단, 이 문장은 원문이 `HELL O WORLD`인 경우 올바른 변환이다.)
- `aHELLOWORLDa` (규칙 2는 한 단어에 적용되어야 한다. 단, 이 문장은 원문이 `HELLOWORLD`인 경우 올바른 변환이다.)
- `HaEaLaLaOWaOaRaLaD` (첫 번째 단어에 쓰인 기호 `a`를 두 번째 단어에 쓸 수 없다.)
- `abHELLObaWORLD` (하나의 규칙을 같은 단어에 두 번 적용할 수 없다.)

신고된 글에 대해 위 규칙이 적용되기 전 문구를 찾을 수 있으면 자동 차단 시스템을 다시 온전하게 실행할 수 있게 된다. 카카오스토리가 광고글 없는 깨끗한 공간이 될 수 있도록 프로그램을 만들어보자.

입력 형식

입력은 문자열 변수 `sentence`로 주어진다. 이 문자열은 영문 대소문자로만 이루어져 있으며, 길이는 `1,000` 이하이다.

출력 형식

입력으로 주어진 광고 문구의 규칙 적용 전 원래 문구를 리턴한다. 단 원래 문구의 경우 문장 앞뒤의 공백이 없어야 하며, 단어 사이의 공백은 한 글자여야 한다. 가능한 답이 여러 가지인 경우 그중 하나를 리턴하면 된다. 규칙에 따른 변환으로 만들 수 없는 문자열이 입력된 경우에는 소문자로 `invalid`를 리턴한다.

예제 입출력

sentence	answer
"HaEaLaLaObWORLDb"	"HELLO WORLD"
"SpIpGpOpNpGJqOqA"	"SIGONG JOA"
"AxAxAxAoBoBoB"	"invalid"

예제에 대한 설명

첫 번째 테스트 케이스는 문제 설명에 제시된 데이터와 같다.

두 번째 테스트 케이스에서, 기호 `q`는 규칙 1, 혹은 규칙 2에 의해 추가된 기호일 수 있다. 규칙 1에 해당하는 경우 원문은 `SIGONG JOA`로 예제 출력과 같으며, 규칙 2에 해당하는 경우의 원문인 `SIGONG J O A`도 올바른 답이다.

세 번째 테스트 케이스에서 `x`는 규칙 1에 의해 추가된 기호여야 한다. (규칙 2에 의해 추가되었다면 기호가 단어 앞뒤에 붙게 되므로 2개여야 한다.) 그러므로 `AAAA`가 한 단어여야 한다. 마찬가지로 `o`도 규칙 1에 의해 추가된 기호여야 하기 때문에 `ABBB`가 한 단어여야 한다. 이는 동시에 만족할 수 없는 조건이므로 주어진 문구는 규칙을 만족할 수 없게 된다. 따라서 `invalid`를 리턴한다.

CODE FESTIVAL

문제 4

4단 고음

I'm in my dream~↗ ~↗ ~↗

IU는 본인의 장기인 3단 고음으로 유명하다. 그러던 그녀가 어느 날 4단 고음을 성공했고 그녀의 고음은 학계에서 연구가 될 만큼 유명해졌다[1].

아이유의 고음 발성 특성 분석

건두현, 배명진(숭실대학교)

The High-Pitched Analysis of IU

Doo-Heon Kyon, Myung-Jin Bae(Soongsil University)

아이유는 뛰어난 가창력과 3 단 고음 발성 등으로 많은 인기를 얻고 있다. 본 논문은 아이유의 발성을 피치와 지속시간을 중심으로 분석하였다. 분석 결과 아이유는 최고음에 도달하더라도 주파수의 변동폭이 매우 안정되며 풍부한 폐활량을 통해 매우 긴 지속시간을 보이고 있다. 아이유의 성부는 소프라노이며 음악에서는 해당성부에서 요구되는 최고 음역이상까지 훌륭하게 표현하고 있다.

폭포 밑 득음 수련을 하던 어느 날, 그녀는 4단 고음이 끝이 아님을 깨달았다. 3단 고음 직후 3단 고음을 연이어하거나, 3단 고음 중 다시 3단 고음을 해서 음높이를 올리는 방법이다. 어떤 순서로 3단 고음을 했는지에 따라 최종 음높이가 달라지기 때문에, 연속 3단 고음을 연습할 때마다 그 결과를 기록으로 남기기로 했다.

---

[1] 견두헌, 배명진. “아이유의 고음 발성 특성 분석”, 한국음향학회, 2011년 춘계학술대회 학술발표논문지

3단 고음은 다음과 같이 적용된다. 1단계에서는 음높이가 세 배가 되며, 2단계와 3단계에서 음높이가 각각 1씩 증가한다. 이를 기록으로 남길 때 `*` 와 `+` 기호를 사용하기로 했다. 즉, 3단 고음을 한 번 한 경우는 문자열로 나타내면 다음과 같다.

`*++`

이때 3단 고음을 마치고 연달아 3단 고음을 한 경우는 `*++*++` 와 같이 표현할 수 있다. 3단 고음의 2단계를 마친 후 3단 고음을 새로 시작한 다음, 나머지 단계를 이어서 하는 경우는 `*+*+++`로 표현할 수 있다. (강조된 부분이 2번째 3단 고음을 부른 부분이다.)

이와 같이 `*` 와 `+` 로 구성된 문자열이 3단 고음의 규칙을 적용하여 만들 수 있는 문자열인 경우 '올바른 문자열'이라고 하자. 다음의 문자열은 3단 고음의 규칙으로 만들 수 있는 문자열이 아니므로 올바른 문자열이 아니다.

- `+**+++`
- `*+++*+`

올바른 문자열에 대해 음높이는 다음과 같이 계산할 수 있다. 시작 음높이는 항상 1이며, 문자열의 처음부터 순서대로 `*` 기호의 경우 3을 곱하고 `+` 기호의 경우 1을 더한다. `*+*+++`의 음높이를 계산하는 과정을 예로 들면 아래와 같다.

시작 음 높이: 1

*	+	*	+	+	+
*3	+1	*3	+1	+1	+1

최종 음 높이: 15

그날 기분에 따라 최종 음높이를 정하는 IU는 최종 음높이를 결정했을 때 서로 다른 3단 고음 문자열이 몇 가지나 있는지 궁금하다. 여러분의 도움이 필요하다.

입력 형식

입력은 `5` 이상 `2^31-1` 이하의 정수 `n`으로 주어진다.

출력 형식

입력을 만족하는 서로 다른 문자열의 수를 리턴한다.

예제 입출력

n	answer
15	1
24	0
41	2
2147483647	1735

예제에 대한 설명

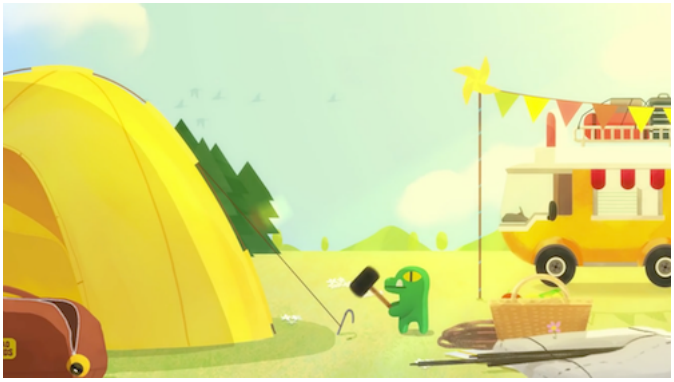
세 번째 예제의 두 가지 경우는 다음과 같다.

```
**++++*++
*+**+++++
```

CODE FESTIVAL

문제 5

캠핑



무지를 돌보느라 지친 콘은 한적한 시골의 한 캠핑장에 놀러 갔다. 캠핑장은 텐트를 칠 수 있는 넓은 평지를 제공하고 있었는데, 이 평지에는 이미 캠핑장에서 설치해 놓은 `n`개의 쐐기가 박혀 있다. 캠핑장 이용 고객은 이 쐐기들 중 두 개를 골라 다음과 같은 조건을 만족하도록 텐트를 설치해야 한다.

- 텐트는 직사각형 형태여야 한다.
- 텐트의 네 면이 정확하게 동, 서, 남, 북을 향해야 한다.
- 대각에 위치하는 텐트의 두 꼭짓점이 정확하게 선택한 두 개의 쐐기에 위치해야 한다.
- 텐트가 점유하는 직사각형 영역의 넓이는 0보다는 커야 한다.
- 텐트가 점유하는 직사각형 영역 내부에 다른 쐐기를 포함하면 안 된다. (다른 쐐기가 경계에 위치하는 경우는 허용함)

캠핑장에서는 위와 같은 조건을 만족하는 위치라면 어디든 고객이 텐트를 설치할 수 있도록 정확한 크기의 텐트를 모두 구비하여 대여해준다고 한다.

당신은 위와 같은 조건을 만족하는 텐트를 설치할 수 있는 쐐기의 쌍의 개수는 총 몇 가지가 될지 궁금해졌다. `n`개의 쐐기의 위치가 좌표로 주어질 때, 위의 조건을 만족하는 쐐기의 쌍의 개수를 계산하는 프로그램을 작성하시오. 단, 동서 방향은 x축, 남북 방향은 y축과 평행하다고 가정한다.

입력 형식

입력은 쐐기의 개수를 의미하는 `n`과, `n × 2` 크기의 2차원 배열 `data`로 주어지며, 배열의 각 행은 캠핑장에 설치된 쐐기의 x좌표와 y좌표를 의미한다. 제한 조건은 다음과 같다.

- `2 <= n <= 5,000`
- `n`개의 쐐기는 모두 x좌표 `0` 이상 `2^31-1` 이하, y좌표 `0` 이상 `2^31-1` 이하에 위치한다.
- 입력되는 `n`개의 쐐기 중 x좌표와 y좌표가 모두 같은 경우는 없다.

출력 형식

입력에 주어진 각 케이스에 대해 가능한 텐트의 쐐기의 쌍의 개수를 정수 형태로 리턴한다.

예제 입출력

n	timetable	answer
4	[[0, 0], [1, 1], [0, 2], [2, 0]]	3

예제에 대한 설명

예제에는 총 4개의 쐐기가 있으며 이 중 (0,0)-(1,1), (0,2)-(1,1), (1,1)-(2,0)의 세 가지 위치에만 텐트를 설치할 수 있다. (0,0)-(0,2)와 (0,0)-(2,0)의 경우에는 직사각형 영역의 넓이가 0이 되기 때문에 조건을 만족하지 못하며, (0,2)-(2,0)의 경우 (1,1) 위치의 쐐기가 직사각형의 내부에 포함되므로 조건을 만족하지 못한다.

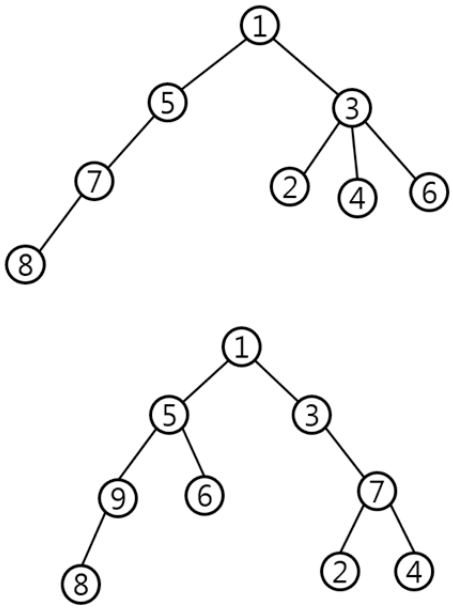
문제 6

신비로운 유적 탐험

카카오 고고학 연구팀은 20,000년 전 문명의 발상지를 조사하던 중 벽면에 그려진 비슷한 모양의 그림을 여러 개 발견하게 되었다. 각각의 그림은 트리 형태로 구성되어 있으며 그림의 일부가 유실되어 전체의 정보를 해독할 수는 없었다.

연구팀은 이 그림들이 특정한 부족의 가계도를 그린 것으로 추정하였다. 트리의 루트는 부족의 조상을 의미하며, 루트의 자식들은 트리 상에서 부모-자식 관계로 연결되어 있고, 그들의 자식은 또 새로운 가지로 연결되어 있는 식이다. 이렇게 가족 전체의 관계가 여러 개의 (같은 모양의) 그림으로 남아있으며, 유실된 정보를 제외하고 공통적으로 남아있는 정보를 토대로 이 부족에 대한 연구를 진행할 수 있다고 판단하였다. 다행히 그림의 중심부에는 유실의 흔적이 없어 두 그림의 루트는 같은 사람을 의미한다고 간주할 수 있었다.

아래 그림은 서로 다른 두 그림에서 얻은 정보를 보기 좋은 형태로 나타낸 것이다. 트리의 번호들은 한 트리에서 각각의 정점을 구별하기 위한 것으로 두 트리의 같은 번호가 같은 사람이라는 의미는 아니다.



가설이 맞는지를 확인하기 위해, 두 그림에서 얻은 트리의 공통부분이 얼마나 되는지를 알아보고 싶다. 그림에서 확인할 수 있는 정보는 상대적인 부모-자식 관계가 전부이기 때문에 자식들의 이름이나 순서 등의 정보는 없다. 따라서 자식들의 순서를 무시하고, 각 트리의 루트를 포함하는 부분 트리로서 두 트리에 모두 포함되는 트리를 공통부분으로 정의하자.

두 개의 트리를 입력으로 받아 최대 공통부분의 크기를 계산하는 프로그램을 작성하라.

입력 형식

입력은 두 트리를 나타내는 `n1`, `g1`, `n2`, `g2`로 주어진다. `n1`, `n2`는 각 트리의 노드 수를 의미한다. `g1`, `g2`는 트리의 정보를 나타내는 값으로, 각각 크기가 `(n1 - 1) × 2`와 `(n2 - 1) × 2`인 2차원 배열로 주어진다. `g1`, `g2`의 각각의 행은 연결된 두 노드를 의미하는데, 두 개의 값 중 하나가 부모 노드의 번호, 다른 하나가 자식 노드의 번호이다. 입력되는 값의 제한조건은 아래와 같다.

- `1 <= n1, n2 <= 100`
- 노드 번호는 각각 `1`부터 `n1`, `n2`까지의 값이 사용되며, 각 트리의 `1`번 노드가 루트이다.
- 입력되는 데이터는 항상 올바른 트리임이 보장된다.

출력 형식

두 트리의 최대 공통부분의 노드 수를 리턴한다.

예제 입출력

변수명	값
n1	8
g1	[[3, 1], [5, 7], [8, 7], [2, 3], [3, 6], [1, 5], [4, 3]]
n2	9
g2	[[1, 5], [5, 6], [3, 7], [3, 1], [7, 4], [7, 2], [9, 8], [5, 9]]
answer	7

예제에 대한 설명

본문의 그림과 같은 예이다. 첫 번째 트리에서 1, 2, 3, 4, 5, 7, 8번의 사용자를 포함하는 트리는 두 번째 트리에서 1, 3, 4, 5, 6, 7, 9번 사용자를 포함하는 트리와 모양이 동일하며, 이것이 가장 큰 경우이다.