슈퍼트리 잇기 (supertrees)

Gardens by the Bay는 싱가포르에 있는 공원이다. 이 공원에는 슈퍼트리라는 이름의 n개의 탑이 있다. 이 탑은 0부터 $n-1$까지 번호가 매겨져 있다. 우리는 **0개 이상**의 다리를 지으려고 한다. 각각의 다리는 서로 다른 두 탑을 잇고, **양방향으로** 건널 수 있다. 한 쌍의 탑을 잇는 다리는 최대 하나이다.

탑 x에서 탑 y로의 경로는 다음 조건을 만족하는 하나 또는 그 이상 탑의 서열이다.

- 서열의 첫번째 원소는 x이다.
- 서열의 마지막 원소는 y이다.
- 동일한 원소가 서열에서 **두 번 이상 나오는 경우는 없다.**
- 서열에서 인접한 두 원소 (탑)은 다리 하나로 연결되어 있다.

정의에 의해서 탑 하나에서 자기 자신으로 가는 경로의 수는 정확하게 하나이며, 탑 i에서 탑 j로 가는 경로의 가짓수는 탑 j에서 탑 i로 가는 경로의 가짓수와 같다는데 유의하라.

설계를 담당한 책임자는, 다음 조건을 만족하게 다리를 지으려고 한다. 모든 $0 \leq i, j \leq n-1$에 대해서, 탑 i에서 탑 j로 가는 서로 다른 경로가 정확하게 $p[i][j]$ 가지이다. $0 \leq p[i][j] \leq 3$이다.

설계 요구 사항을 만족하게 다리를 짓거나, 이것이 불가능하다는 것을 판정하는 프로그램을 작성하시오.

Implementation details

다음 함수를 구현해야 한다.

```
int construct(int[][] p)
```

- p: $n \times n$ 크기 배열로 설계 요구 사항을 나타낸다.
- 만약 요구 사항을 만족하게 다리를 짓는 것이 가능하다면, 이 함수는 아래에 설명하는 `build` 함수를 정확히 한 번 호출하여 어떻게 다리를 짓는지 제출한다. 그 다음 1을 리턴해야 한다.
- 그렇지 않다면, 이 함수는 `build` 함수를 호출하지 않고 0을 리턴해야 한다.
- 이 함수는 정확히 한 번 호출된다.

`build` 함수는 다음과 같이 정의된다.

```
void build(int[][] b)
```

- b: $n \times n$ 크기 배열로 만약 탑 i와 탑 j를 잇는 다리가 있다면 $b[i][j] = 1$이고, 그렇지 않다면 $b[i][j] = 0$이다.
- 이 배열은 모든 $0 \leq i, j \leq n-1$에 대해서 $b[i][j] = b[j][i]$이고 모든 $0 \leq i \leq n-1$에 대해서

$b[i][i] = 0$이어야 한다.

Examples

Example 1

다음 호출을 고려해보자.

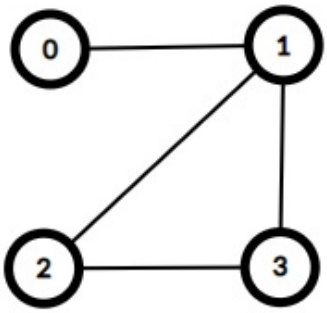
```
construct([[1, 1, 2, 2], [1, 1, 2, 2], [2, 2, 1, 2], [2, 2, 2, 1]])
```

이는 탑 0에서 탑 1로 정확히 하나의 경로가 있어야 한다는 뜻이다. 그 외 모든 다른 탑 $0 \leq x < y \leq 3$의 쌍 (x, y)에는 탑 x에서 탑 y로 가는 경로가 정확히 두 개 있어야 한다.

4개의 다리를 지어서 다음 탑의 쌍 $(0, 1)$, $(1, 2)$, $(1, 3)$, $(2, 3)$을 이으면 조건을 만족한다.

이 답을 제출하기 위해서 `construct` 함수는 다음과 같이 함수 호출을 해야 한다.

- `build([[0, 1, 0, 0], [1, 0, 1, 1], [0, 1, 0, 1], [0, 1, 1, 0]])`



리턴값은 1이어야 한다.

이 경우에는, 요구 사항을 만족하게 다리를 놓는 방법이 여러 가지 존재하고, 그 중 어느 것도 정답으로 인정된다.

Example 2

다음 호출을 고려해보자.

```
construct([[1, 0], [0, 1]])
```

이는 어떤 두 탑에 대해서도 경로가 없어야 한다는 뜻이다. 이는 다리를 짓지 않는 것으로 해결할 수 있다.

따라서 `construct` 함수는 다음과 같이 함수 호출을 해야 한다.

- `build([[0, 0], [0, 0]])`

그 다음, `construct`는 1을 리턴해야 한다.

Example 3

다음 호출을 고려해보자.

```
construct([[1, 3], [3, 1]])
```

이는 탑 0에서 탑 1로 정확히 3 개의 경로가 있어야 한다는 뜻이다. 이 제약 조건을 맞추는 방법은 없다. 따라서, `construct` 함수는 `build` 함수를 호출하지 않고 0을 리턴해야 한다.

Constraints

- $1 \le n \le 1000$
- $p[i][i] = 1$ (for all $0 \le i \le n-1$)
- $p[i][j] = p[j][i]$ (for all $0 \le i, j \le n-1$)
- $0 \le p[i][j] \le 3$ (for all $0 \le i, j \le n-1$)

Subtasks

1. (11 points) $p[i][j] = 1$ (for all $0 \le i, j \le n-1$)
2. (10 points) $p[i][j] = 0$ or 1 (for all $0 \le i, j \le n-1$)
3. (19 points) $p[i][j] = 0$ or 2 (for all $i \ne j$, $0 \le i, j \le n-1$)
4. (35 points) $0 \le p[i][j] \le 2$ (for all $0 \le i, j \le n-1$)이고 요구사항을 만족하게 다리를 놓는 방법이 최소 하나가 존재한다.
5. (21 points) $0 \le p[i][j] \le 2$ (for all $0 \le i, j \le n-1$)
6. (4 points) 추가적인 제약 조건이 없다.

Sample grader

샘플 그레이더는 다음 양식으로 입력을 읽는다.

- line 1: n
- line $2+i$ ($0 \le i \le n-1$): $p[i][0]$ $p[i][1]$ $\ldots$ $p[i][n-1]$

샘플 그레이더는 다음 양식으로 출력한다.

- line 1: `construct`의 리턴값.

만약 `construct`의 리턴값이 1이면, 샘플 그레이더는 추가로 다음을 출력한다.

- line $2+i$ ($0 \le i \le n-1$): $b[i][0]$ $b[i][1]$ $\ldots$ $b[i][n-1]$