축구 경기장

Debrecen 도시에 있는 Nagyerdő는 정사각형 모양의 숲으로, $N \times N$ 개의 셀들의 격자로 모델링 가능하다. 격자의 행은 북쪽에서 남쪽으로 0부터 $N-1$까지 번호가 붙어있고, 격자의 열은 서쪽에서 동쪽으로 0부터 $N-1$까지 번호가 붙어있다. 격자의 r행 c열에 있는 셀은 셀 (r, c)로 부른다.

숲에서, 각 셀은 **빈 칸**이거나 **나무**가 있다. 숲에서 적어도 하나의 셀은 빈 칸이다.

이 도시의 유명한 스포츠 클럽인 DVSC는 숲에 새로운 축구 경기장을 지으려고 한다. 크기 s $(s \geq 1)$인 경기장은 s개의 **서로 다른 빈 칸**인 셀 $(r_0, c_0), \ldots, (r_{s-1}, c_{s-1})$의 집합이다. 엄밀하게 이것이 의미하는 것은:

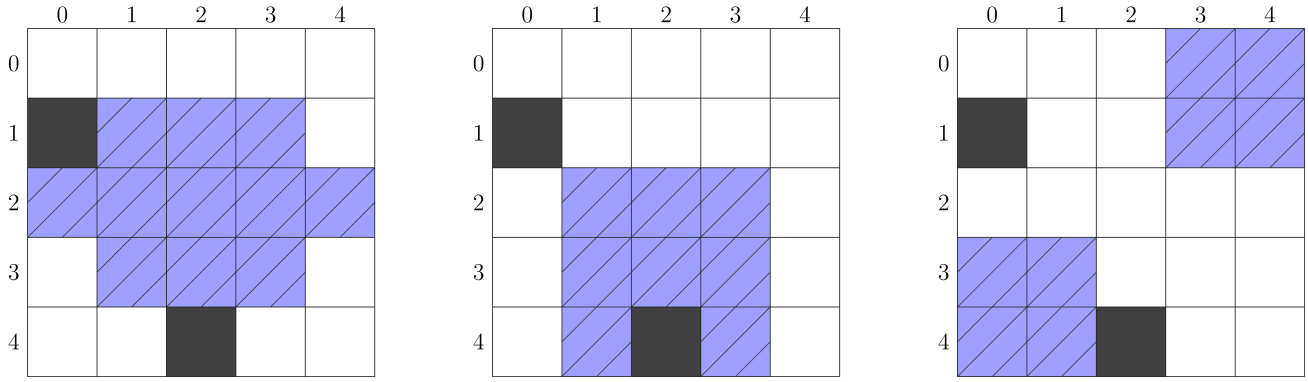
- 0부터 $s-1$까지의 각 i에 대해, 셀 (r_i, c_i)는 빈 칸이어야 하고,
- $0 \leq i < j < s$인 각 i, j에 대해, $r_i \neq r_j$와 $c_i \neq c_j$ 중 최소 하나는 성립해야 한다.

축구는 경기장의 셀들을 오가며 움직이는 공을 이용해서 경기를 한다. **직선 킥**은 다음 두 행동 중 하나로 정의된다:

- 셀 (r, a)에서 셀 (r, b) $(0 \leq r, a, b < N, a \neq b)$로 공을 움직인다. 이때 경기장은 r행의 셀 (r, a)에서 (r, b) 사이의 *모든* 셀을 포함한다. 엄밀하게,
  - 만약 $a < b$이면 경기장은 $a \leq k \leq b$인 각 k에 대해 셀 (r, k)를 포함해야 하고,
  - 만약 $a > b$이면 경기장은 $b \leq k \leq a$인 각 k에 대해 셀 (r, k)를 포함해야 한다.
- 셀 (a, c)에서 셀 (b, c) $(0 \leq c, a, b < N, a \neq b)$로 공을 움직인다. 이때 경기장은 c열의 셀 (a, c)에서 (b, c) 사이의 *모든* 셀을 포함한다. 엄밀하게,
  - 만약 $a < b$이면 경기장은 $a \leq k \leq b$인 각 k에 대해 셀 (k, c)를 포함해야 하고,
  - 만약 $a > b$이면 경기장은 $b \leq k \leq a$인 각 k에 대해 셀 (k, c)를 포함해야 한다.

경기장에 포함되는 임의의 셀에서 경기장에 포함되는 임의의 다른 셀로 최대 2번의 직선 킥으로 공을 움직일 수 있는 경우에 경기장이 **정상적**이라고 한다. 참고로 크기 1인 경기장은 모두 정상적이다.

예를 들어, 셀 $(1, 0)$과 $(4, 2)$에는 나무가 있고 나머지 모든 셀은 빈 칸인 크기 $N = 5$인 숲을 고려하자. 아래 그림은 세 가지의 가능한 경기장을 보여준다. 나무가 있는 셀은 어둡게, 경기장에 포함된 셀은 줄무늬로 표현했다.



왼쪽 경기장은 정상적이다. 하지만, 가운데 경기장은 정상적이 아닌데, 이유는 셀 $(4, 1)$에서 $(4, 3)$으로 공을 움직이려면 최소한 3번의 직선 킥이 필요하기 때문이다. 오른쪽 경기장도 정상적이 아닌데, 이유는 셀 $(3, 0)$에서 $(1, 3)$으로 직선 킥으로 공을 움직이는 것이 불가능하기 때문이다.

스포츠 클럽은 가능한한 큰 정상적인 경기장을 짓고 싶어한다. 당신은 숲에 존재할 수 있는 정상적인 경기장의 크기 s의 최댓값을 구해야 한다.

Implementation Details

다음 함수를 구현해야 한다.

```
int biggest_stadium(int N, int[][] F)
```

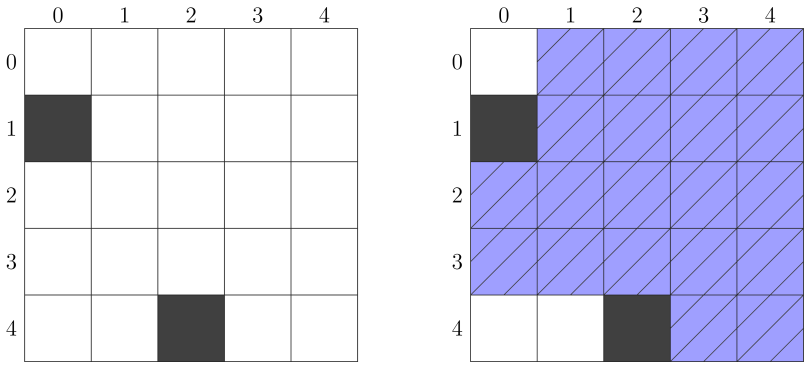
- N: 숲의 크기
- F: 숲의 셀을 나타내는, 길이 N인 배열들을 갖는 길이 N인 배열. $0 \leq r < N$이고 $0 \leq c < N$인 각각의 r과 c에 대해, $F[r][c] = 0$는 셀 (r, c)가 빈 칸임을 의미하고, $F[r][c] = 1$은 그 셀에 나무가 있는 것을 의미한다.
- 이 함수는 숲에 지을 수 있는 정상적인 경기장의 최대 크기를 리턴해야 한다.
- 이 함수는 각 테스트 케이스에 대해 정확히 한번만 호출된다.

Example

다음 호출을 생각해보자:

```
biggest_stadium(5, [[0, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 0, 0, 0],
                    [0, 0, 1, 0, 0]])
```

이 예에서, 다음 그림의 왼쪽은 숲을 나타내고 오른쪽은 크기 20인 정상적인 경기장을 나타낸다.



크기가 21 이상인 정상적인 경기장이 존재하지 않기 때문에, 이 함수는 20을 리턴해야 한다.

Constraints

- $1 \leq N \leq 2\,000$
- $0 \leq F[i][j] \leq 1$ ($0 \leq i < N$이고 $0 \leq j < N$인 모든 i와 j에 대해)
- 숲에는 최소 하나의 빈 칸이 존재한다. 바꿔 얘기하면, 어떤 $0 \leq i < N$와 $0 \leq j < N$에 대해 $F[i][j] = 0$이 성립한다.

Subtasks

1. (6 points) 나무가 있는 셀은 최대 한개이다.
2. (8 points) $N \leq 3$
3. (22 points) $N \leq 7$
4. (18 points) $N \leq 30$
5. (16 points) $N \leq 500$
6. (30 points) 추가적인 제한이 없다.

각 서브태스크에 대해, *모든* 빈 칸 셀로 구성되는 집합이 정상적인 경기장인지 올바르게 판정한다면 서브태스크 점수의 25%를 얻을 수 있다.

보다 정확하게는, 모든 빈 칸 셀로 구성되는 집합이 정상적인 경기장인 각 테스트 케이스에 대해, 당신의 점수는:

- 정확한 답(빈 칸 셀의 총 개수)을 리턴한 경우, 만점을 받는다.
- 그 외의 경우, 0점을 받는다.

모든 빈 칸 셀로 구성되는 집합이 정상적인 경기장이 *아닌* 각 테스트 케이스에 대해, 당신의 점수는:

- 정확한 답을 리턴한 경우, 만점을 받는다.
- 빈 칸 셀의 총 개수를 리턴한 경우, 0점을 받는다.
- 위 두 경우와 다른 값을 리턴한 경우, 25%의 점수를 받는다.

각 서브태스크의 최종 점수는 그 서브태스크에 속한 테스트 케이스들의 점수의 최솟값으로 정해진다.

Sample Grader

샘플 그레이더의 입력 형식은 다음과 같다:

- line 1: N
- line $2 + i$ ($0 \leq i < N$): $F[i][0]$ $F[i][1]$ $\ldots$ $F[i][N-1]$

샘플 그레이더는 다음 형식으로 답을 출력한다:

- line 1: `biggest_stadium`의 리턴값