Soccer Stadium

Nagyerdő is a square-shaped forest located in the city of Debrecen, which can be modeled as an $N \times N$ grid of cells. The rows of the grid are numbered from 0 to $N-1$ from north to south, and the columns are numbered from 0 to $N-1$ from west to east. We refer to the cell located at row r and column c of the grid as cell (r, c).

In the forest, each cell is either **empty** or contains a **tree**. At least one cell in the forest is empty.

DVSC, the famous sports club of the city, is planning to build a new soccer stadium in the forest. A stadium of size s (where $s \geq 1$) is a set of s *distinct (එකිනෙකට වෙනස්) empty* cells $(r_0, c_0), \ldots, (r_{s-1}, c_{s-1})$. Formally this means:

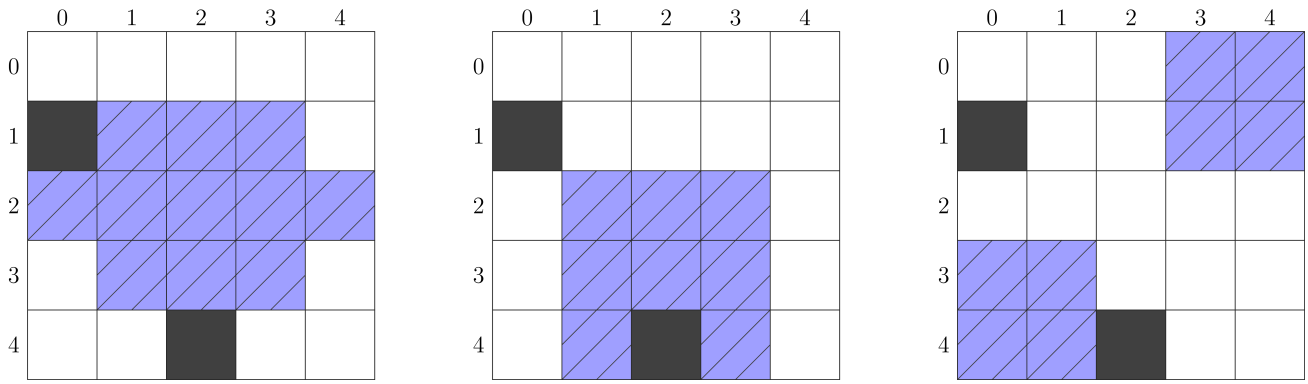
- for each i from 0 to $s-1$, inclusive (0 සහ $s-1$ ඇතුළුව), cell (r_i, c_i) is empty,
- for each i, j such that $0 \leq i < j < s$, at least one of $r_i \neq r_j$ and $c_i \neq c_j$ holds.

Soccer is played using a ball that is moved around the cells of the stadium. A **straight kick** is defined to be either of the following two actions:

- Move the ball from cell (r, a) to cell (r, b) ($0 \leq r, a, b < N, a \neq b$), where the stadium contains *all* cells between cell (r, a) and (r, b) in row r. Formally,
  - if $a < b$ then the stadium should contain cell (r, k) for each k such that $a \leq k \leq b$,
  - if $a > b$ then the stadium should contain cell (r, k) for each k such that $b \leq k \leq a$.
- Move the ball from cell (a, c) to cell (b, c) ($0 \leq c, a, b < N, a \neq b$), where the stadium contains *all* cells between cell (a, c) and (b, c) in column c. Formally,
  - if $a < b$ then the stadium should contain cell (k, c) for each k such that $a \leq k \leq b$,
  - if $a > b$ then the stadium should contain cell (k, c) for each k such that $b \leq k \leq a$.

A stadium is **regular** if it is possible to move the ball from any cell contained by the stadium to any other cell contained by the stadium with at most (උපරිම) 2 straight kicks. Note that any stadium of size 1 is regular.

For example, consider a forest of size $N = 5$, with cells $(1, 0)$ and $(4, 2)$ containing trees and every other cell being empty. The figure below shows three possible stadiums. Cells with trees are darkened, and cells contained by the stadium are striped.



The stadium on the left is regular. However, the stadium in the middle is not regular, because at least 3 straight kicks are needed to move the ball from cell $(4, 1)$ to $(4, 3)$. The stadium on the right is also not regular, because it is impossible to move the ball from cell $(3, 0)$ to $(1, 3)$ using straight kicks.

The sports club wants to build a regular stadium that is as big as possible. Your task is to find the maximum value of s such that there exists a regular stadium of size s in the forest.

Implementation Details

You should implement the following procedure.

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

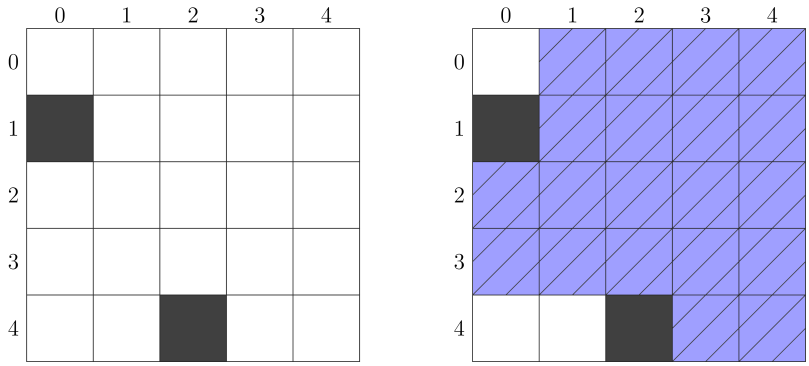
- N: the size of the forest.
- F: an array of length N containing arrays of length N, describing cells in the forest. For each r and c such that $0 \leq r < N$ and $0 \leq c < N$, $F[r][c] = 0$ means that cell (r, c) is empty, and $F[r][c] = 1$ means that it contains a tree.
- This procedure should return the maximum size of a regular stadium that can be built in the forest.
- This procedure is called exactly once for each test case.

Example

Consider the following call:

```
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]])
```

In this example, the forest is displayed on the left and a regular stadium of size 20 is displayed on the right of the following figure:



Since there is no regular stadium of size 21 or greater, the procedure should return 20.

Constraints

- $1 \leq N \leq 2\,000$
- $0 \leq F[i][j] \leq 1$ (for each i and j such that $0 \leq i < N$ and $0 \leq j < N$)
- There is at least one empty cell in the forest. In other words, $F[i][j] = 0$ for some $0 \leq i < N$ and $0 \leq j < N$.

Subtasks

1. (6 points) There is at most one cell containing a tree.
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) No additional constraints.

In each subtask, you can obtain 25% of the subtask score if your program judges correctly whether the set consisting of *all* the empty cells is a regular stadium.

More precisely, for each test case in which the set consisting of all the empty cells is a regular stadium, your solution:

- gets full points if it returns the correct answer (which is the size of the set consisting of all the empty cells).
- gets 0 points otherwise.

For each test case in which the set consisting of all the empty cells is *not* a regular stadium, your solution:

- gets full points if it returns the correct answer.
- gets 0 points if it returns the size of the set consisting of all the empty cells.
- gets 25% of the points if it returns any other value.

The score for each subtask is the minimum of the points for the test cases in the subtask.

Sample Grader

The sample grader reads the input in the following format:

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

The sample grader prints your answer in the following format:

- line 1: the return value of `biggest_stadium`