Estadio de fútbol

Nagyerdő es un bosque de forma cuadrada que se encuentra en la ciudad de Debrecen, y que se puede modelar por una cuadrícula de $N \times N$ casillas. Las filas de la cuadrícula están numeradas de 0 a $N-1$ de norte a sur y las columnas numeradas de 0 a $N-1$ de oeste a este. Nos referimos a la casilla que se encuentra en la fila r y columna c de la cuadrícula como casilla (r, c).

En el bosque, cada casilla está **vacía** o contiene un **árbol**. Como mínimo una casilla del bosque está vacía.

DVSC, el famoso club deportivo de la ciudad, está planeando construir un estadio de fútbol en el bosque. Un estadio de tamaño s (donde $s \geq 1$) es un conjunto de s casillas *vacías y distintas* $(r_0, c_0), \ldots, (r_{s-1}, c_{s-1})$. Formalmente significa que:

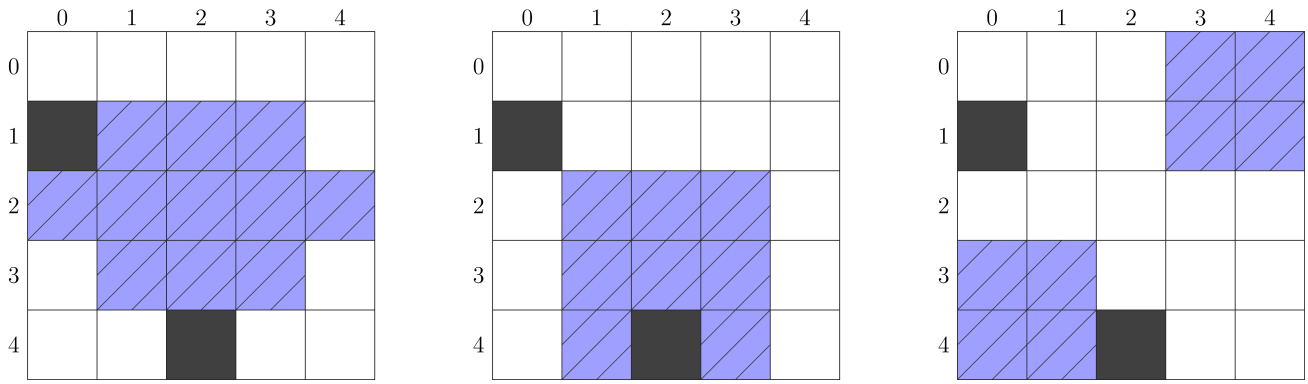
- para cada i de 0 a $s-1$, inclusive, la casilla (r_i, c_i) está vacía.
- para cada i, j tal que $0 \leq i < j < s$, como mínimo uno entre $r_i \neq r_j$ y $c_i \neq c_j$ se satisface.

El fútbol se juega con una pelota que se mueve por las casillas de estadio. Un **disparo recto** se define como una de las siguientes dos acciones:

- Mover la pelota de la casilla (r, a) a la casilla (r, b) ($0 \leq r, a, b < N, a \neq b$), donde el estadio contiene *todas* las casillas entre (r, a) y (r, b) en la fila r. Formalmente,
  - si $a < b$ entonces el estadio contiene las casillas (r, k) para cada k tal que $a \leq k \leq b$,
  - si $a > b$ entonces el estadio contiene la casilla (r, k) para cada k tal que $b \leq k \leq a$.
- Mover la pelota de la casilla (a, c) a la casilla (b, c) ($0 \leq c, a, b < N, a \neq b$), donde el estadio contiene *todas* las casillas entre (a, c) y (b, c) en la columna c. Formalmente,
  - si $a < b$ entonces el estadio contiene las casillas (k, c) para cada k tal que $a \leq k \leq b$,
  - si $a > b$ entonces el estadio contiene las casillas (k, c) para cada k tal que $b \leq k \leq a$.

Un estadio es **regular** si es posible mover la pelota de cualquier casilla contenida en el estadio a cualquier otra casilla contenida en el estadio con como mucho 2 disparos rectos. Fíjate que cualquier estadio de tamaño 1 es regular.

Por ejemplo, considera el bosque de tamaño $N = 5$, con casillas $(1, 0)$ y $(4, 2)$ que contienen árboles y el resto son casillas vacías. La figura muestra tres estadios posibles. Las casillas negras representan los árboles y las rayadas son posibles estadios.



El estadio a la izquierda es regular. Por lo contrario, el estadio en el medio no es regular ya que como mínimo se necesitan 3 disparos rectos para mover la pelota desde $(4, 1)$ a $(4, 3)$. El estadio a la derecha tampoco es regular ya que no es posible mover la pelota desde la casilla $(3, 0)$ a $(1, 3)$ usando disparos rectos.

El club deportivo quiere construir el estadio regular más grande posible. Tu tarea es dar el valor s que cumpla que pueda existir un estadio regular de tamaño s en el bosque.

Detalles de implementación

Tienes que implementar la siguiente función:

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

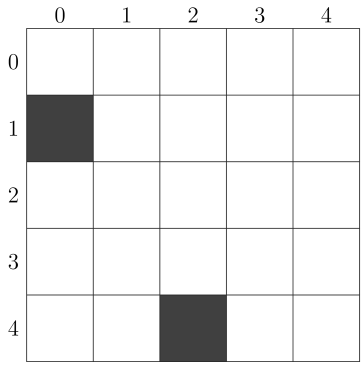
- N: el tamaño del bosque.
- F: un array de longitud N conteniendo arrays de longitud N, describiendo las casillas del bosque. Para cada r y c tal que $0 \leq r < N$ y $0 \leq c < N$, $F[r][c] = 0$ significa que la casilla (r, c) está vacía, y $F[r][c] = 1$ significa que contiene un árbol.
- Esta función debe devolver el tamaño máximo de un estadio regular que se pueda construir en el bosque.
- La función se llama exactamente una vez en cada juego de pruebas.

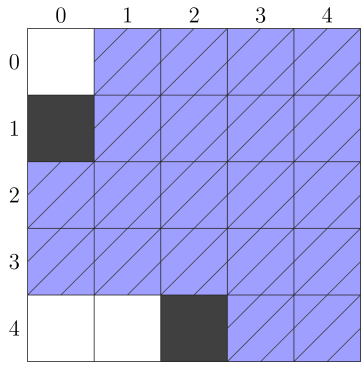
Ejemplo

Considera la siguiente llamada:

```
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]])
```

En este ejemplo, el bosque se muestra a la izquierda y un estadio de tamaño 20 se muestra en la figura de la derecha:





Como no hay un estadio regular de tamaño 21 o superior, la función debe devolver 20.

Restricciones

- $1 \leq N \leq 2\,000$
- $0 \leq F[i][j] \leq 1$ (para cada i y j tal que $0 \leq i < N$ y $0 \leq j < N$)
- Como mínimo hay una casilla vacía en el bosque. En otras palabras, $F[i][j] = 0$ para alguna $0 \leq i < N$ y $0 \leq j < N$.

Subtareas

1. (6 puntos) Como mucho hay una casilla que contiene un árbol.
2. (8 puntos) $N \leq 3$
3. (22 puntos) $N \leq 7$
4. (18 puntos) $N \leq 30$
5. (16 puntos) $N \leq 500$
6. (30 puntos) Sin restricciones adicionales.

En cada subtarea, puedes obtener un 25% de la puntuación de la subtarea si tu programa puede juzgar correctamente si el conjunto de *todas* las casillas vacías son un estadio regular.

Más específicamente, para cada caso de prueba en que el conjunto de las casillas vacías es un estadio regular, tu solución:

- obtiene todos los puntos si devuelve la respuesta correcta (que es el tamaño del conjunto de todas las casillas vacías).
- obtiene 0 puntos alternativamente.

Para cada caso de prueba en que el conjunto de casillas vacías *no* sea un estadio regular, tu solución:

- obtiene todos los puntos si devuelve la respuesta correcta.
- obtiene 0 puntos si devuelve el tamaño del conjunto formado por todas las casillas vacías.
- obtiene 25% de los puntos si devuelve cualquier otro valor.

La puntuación para cada subtarea es el mínimo número de puntos de los juegos de prueba de la subtarea.

Grader de ejemplo

El grader de ejemplo lee el input con el siguiente formato:

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

El grader de ejemplo escribe tu respuesta con el siguiente formato:

- línea 1: el valor devuelto por `biggest_stadium`