Estadio de futbol

Nagyerdő es el nombre de un bosque que tiene forma cuadrada, ubicado en la ciudad de Debrecen, puede ser modelado como una grilla de $N \times N$ casillas. Las filas de la grilla están numeradas de 0 a $N-1$ de norte a sur y las columnas están enumeradas de 0 a $N-1$ de oeste a este. Nos referimos a la casilla localizada en la fila r y columna c de la grilla como (r, c).

En el bosque, cada casilla esta **vacía** o tiene un **árbol**. Al menos una casilla está vacía en el bosque.

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

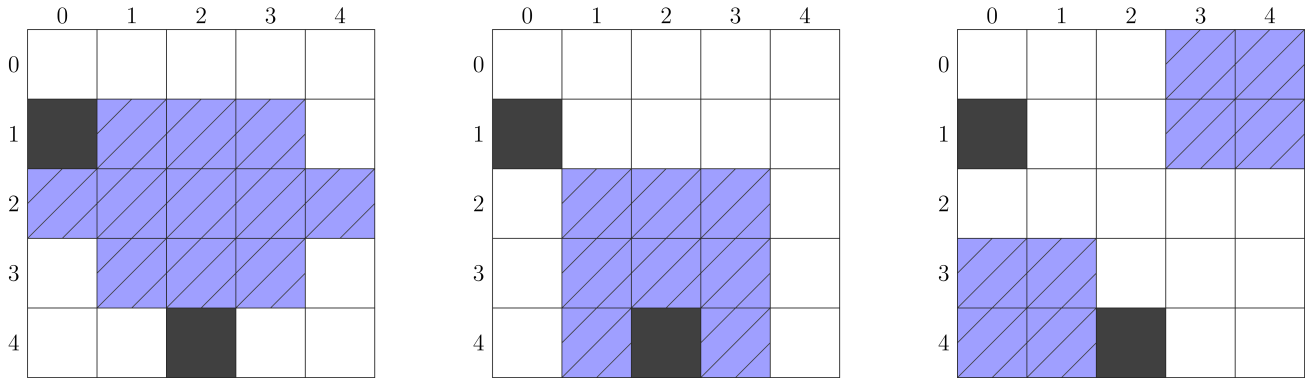
- para todo i desde 0 a $s-1$, inclusive, la casilla (r_i, c_i) está vacía,
- para todo i, j tal que $0 \leq i < j < s$, se cumple al menos una de las siguientes condiciones: $r_i \neq r_j$ o $c_i \neq c_j$.

El futbol se juega usando una pelota que se va moviendo sobre las casillas del estadio. Un **patada recta** está definida como una de las dos siguientes 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 que están entre la casilla (r, a) y la casilla (r, b) en la fila r. Formalmente,
  - si $a < b$ entonces el estadio debe contener las casillas (r, k) para todo k tal que $a \leq k \leq b$,
  - si $a > b$ entonces el estadio debe contener las casillas (r, k) para todo 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 que están entre la casilla (a, c) y la casilla (b, c) en la columna c. Formalmente,
  - si $a < b$ entonces el estadio debe contener las casillas (k, c) para todo k tal que $a \leq k \leq b$,
  - si $a > b$ entonces el estadio debe contener las casillas (k, c) para todo k tal que $b \leq k \leq a$.

Un estadio es **regular** si es posible mover la pelota desde cualquier casilla contenida dentro del estadio a cualquier otra casilla contenida dentro del estadio con a lo sumo 2 patadas rectas. Nótese que cualquier estadio de tamaño 1 es regular.

Por ejemplo, considera un bosque de tamaño $N = 5$, con casillas $(1, 0)$ y $(4, 2)$ que tienen árboles y con las demás casillas vacías. En la imagen de abajo se muestra tres posibles estadios. Las casillas con árboles están marcadas de negro, y las casillas del estadio están marcadas con líneas diagonales.



El estadio de la izquierda es regular. Sin embargo, el estadio del medio es no regular, por que al menos 3 patadas rectas son necesarias para mover la pelota de la casilla $(4, 1)$ a la casilla $(4, 3)$. El estadio de la derecha tampoco es regular, porque es imposible mover la pelota de la casilla $(3, 0)$ a la casilla $(1, 3)$ usando patadas rectas.

El club deportivo quiere construir un estadio regular que sea lo más grande posible. Tu tarea es encontrar el valor máximo de s tal que exista un estadio regular de tamaño s en el bosque.

Detalles de implementación

Debes implementar la siguiente función.

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

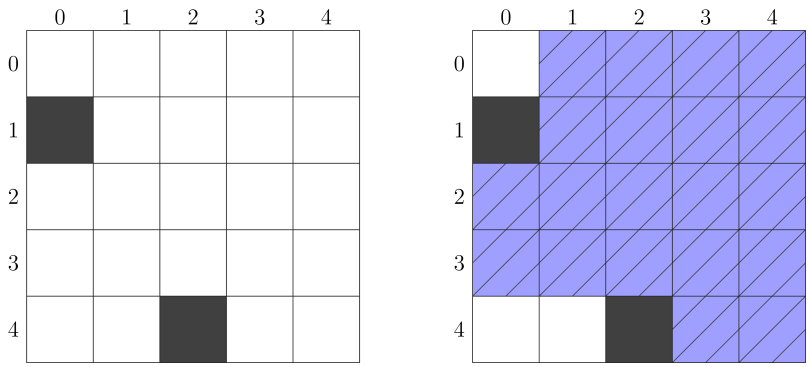
- N: el tamaño del bosque.
- F: un arreglo de tamaño N que contiene arreglos de tamaño N, estos describen las casillas del bosque. Para cada r y c tal que $0 \leq r < N$ y $0 \leq c < N$, $F[r][c] = 0$ significa que (r, c) está vacía, $F[r][c] = 1$ significa que la casilla contiene un árbol.
- Esta función debe retornar el máximo tamaño del estadio que se puede construir en el bosque.
- Esta función es llamada exactamente una vez por cada caso de prueba.

Ejemplo

Considerar el siguiente llamado a la función:

```
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 es mostrado en la izquierda y el estadio regular de tamaño 20 es mostrado en la derecha de la siguiente imagen:



Como no hay algún estadio de tamaño 21 o mayor, la función debe retornar 20.

Límites

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

Subtareas

1. (6 puntos) Hay a lo mucho 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 el 25% del puntaje de la subtarea si tu programa juzga correctamente si el conjunto de *todas* las casillas vacías es un estadio regular.

Más precisamente, para cada caso de prueba donde el conjunto que consiste de todas las casillas vacías es un estadio regular, tu solución:

- obtiene todos los puntos si retorna la respuesta correcta (la cual es el tamaño del conjunto que consiste de todas las casillas vacías).

- obtiene 0 puntos en otro caso.

Para cada caso de prueba donde el conjunto de todas las casillas vacías *no* es un estadio regular, tu solución:

- obtiene todos los puntos si retorna la respuesta correcta.
- obtiene 0 puntos si este retorna el tamaño del conjunto consistente de todas las casillas vacías.
- obtiene 25% de puntaje si retorna otro valor.

El puntaje de cada subtarea es el mínimo de puntos obtenido de entre los casos de prueba en la subtarea.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en 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 evaluador de ejemplo imprime tu respuesta en el siguiente formato:

- línea 1: el valor de retorno de la función `biggest_stadium`