Concurso de robots

Los investigadores de inteligencia artificial de la Universidad de Szeged organizan un concurso de programación de robots. Tu amiga Hanga ha decidido participar en el concurso. El objetivo es programar el Pulibot definitivo *Pulibot*, admirando la gran inteligencia de la famosa raza de perro pastor húngaro, el Puli.

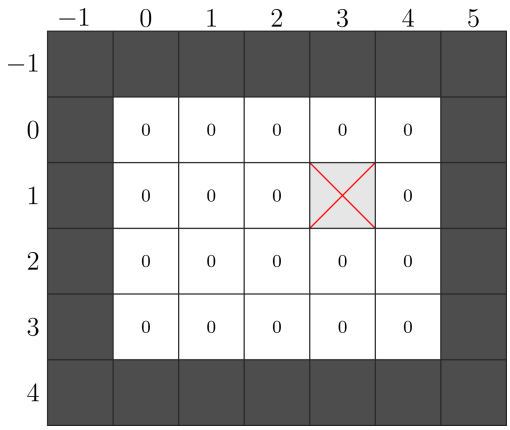
Pulibot será probado en un laberinto que consta de un $(H+2) \times (W+2)$ cuadrícula de celdas. Las filas de la cuadrícula están numeradas desde. -1 hasta H de norte a sur y las columnas de la cuadrícula están numeradas de -1 to W de oeste a este. Nos referimos a la celda ubicada en la fila.r y columna c de la cuadrícula ($-1 \leq r \leq H$, $-1 \leq c \leq W$) como celda (r, c).

Considere una celda (r, c) tal que $0 \leq r < H$ y $0 \leq c < W$. Hay 4 celda **adyacentes** a la celda (r, c):

- celda $(r, c-1)$ es referida como la celda **oeste** de la celda (r, c);
- celda $(r+1, c)$ es referida como la celda **sur** de la celda (r, c);
- celda $(r, c+1)$ es referida como la celda **este** de la celdal (r, c);
- celda $(r-1, c)$ es referida como la celda **norte** de la celda (r, c).

La celda (r, c) es llamada como celda **límite** del laberinto si $r = -1$ o $r = H$ o $c = -1$ o $c = W$ sostiene . Cada celda que no es una celda límite del laberinto es bien una celda de **obstáculo** o una celda **vacía**. Además, cada celda vacía tiene un **color**, representado por un número entero no negativo entre 0 y Z_{MAX}, inclusive. Inicialmente, el color de cada celda vacía es 0.

Por ejemplo, considere un laberinto con $H = 4$ y $W = 5$, que contiene una sola celda de obstáculo $(1, 3)$:



La única celda de obstáculo está marcada con una cruz. Las celdas límite del laberinto están sombreadas. Los números escritos en cada celda vacía representan sus respectivos colores.

Un **camino** de longitud ℓ ($\ell > 0$) desde la celda (r_0, c_0) hasta la celda (r_ℓ, c_ℓ) es una secuencia de celdas *empty* distintas por pares $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ en el que para cada i ($0 \leq i < \ell$) la celda (r_i, c_i) y (r_{i+1}, c_{i+1}) son adjacentes.

Obsérvese que un camino de longitud ℓ ccontiene exactamente $\ell + 1$ celdas.

En el concurso, los investigadores establecen un laberinto en el que existe al menos un camino desde la celda $(0, 0)$ hasta la celda $(H - 1, W - 1)$. Nótese que esto implica que las celdas $(0, 0)$ y $(H - 1, W - 1)$ están garantizadas para estar vacías.

Hanga no sabe qué celdas del laberinto están vacías y qué celdas son obstáculos.

Tu tarea es ayudar a Hanga a programar Pulibot para que sea capaz de encontrar un *camino más corto* (es decir, un camino de longitud mínima) desde la celda $(0, 0)$ hasta la celda $(H - 1, W - 1)$ en el laberinto desconocido que han preparado los investigadores.

La especificación de Pulibot y las reglas del concurso se describen a continuación.

Especificación de Pulibot

Definir el **estado** de una celda (r, c) para cada $-1 \leq r \leq H$ and $-1 \leq c \leq W$ como un número entero de modo que:

- si la celda (r, c) es una celda límite entonces su estado es -2;
- si la celda (r, c) es una celda obstáculo entonces su estado es -1;
- si la celda (r, c) es una celda vacía entonces su estado es el color de la celda.

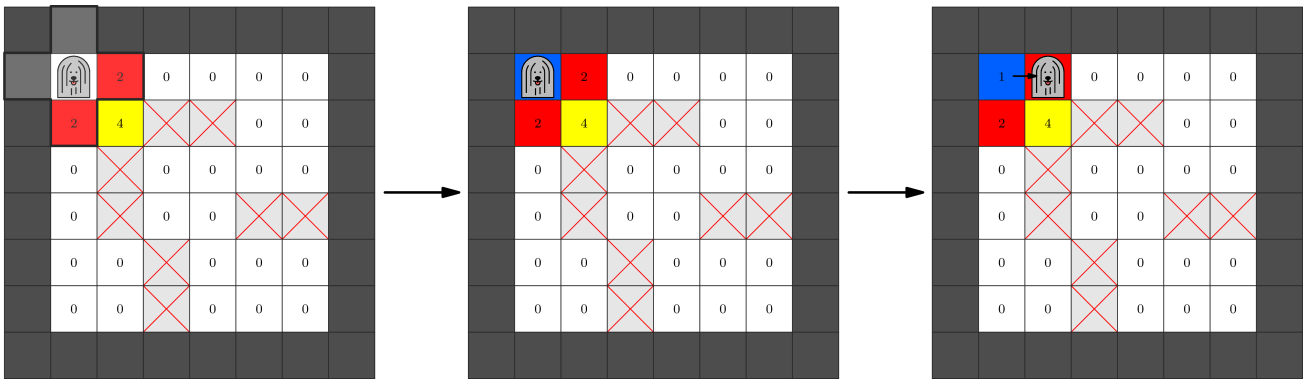
El programa de Pulibot se ejecuta como una secuencia de pasos. En cada paso, Pulibot reconoce los estados de las celdas cercanas y luego realiza una instrucción. La instrucción que ejecuta viene determinada por los estados reconocidos. A continuación se ofrece una descripción más precisa.

Supongamos que al comienzo del paso actual, Pulibot se encuentra en la celda (r, c), que es una celda vacía. El paso se realiza de la siguiente manera

1. En primer lugar, Pulibot reconoce la matriz de estado actual **matriz de estado**, es decir, la matriz $S = [S[0], S[1], S[2], S[3], S[4]]$, consistente en el estado de la celda (r, c) y de todas las celdas adyacentes:
    - $S[0]$ es el estado de la celda (r, c).
    - $S[1]$ es el estado de la celda situada al oeste.
    - $S[2]$ es el estado de la celda situada al sur.
    - $S[3]$ es el estado de la celda situada al este.
    - $S[4]$ es el estado de la celda situada al n

2. A continuación, Pulibot determina la **instrucción** (Z, A) que corresponde a la matriz de estados reconocida.
3. Finalmente, Pulibot ejecuta esa instrucción: establece el color de la celda (r, c) al color Z y luego realiza la acción A, que es una de las siguientes acciones:
    - *stay* en la celda (r, c);
    - *move* a una de las 4 celdas adyacentes;
    - *terminate the program*.

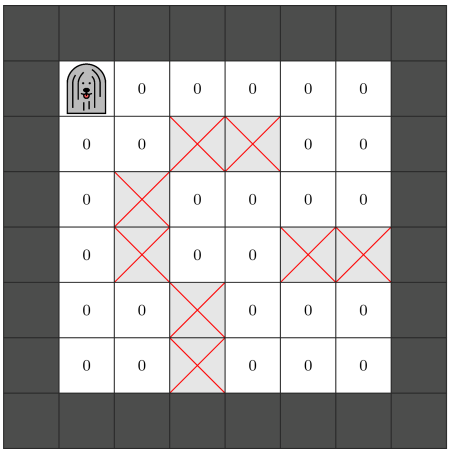
Por ejemplo, considere el escenario mostrado a la izquierda de la siguiente figura. Pulibot está actualmente en la celda $(0, 0)$ con el color 0. Pulibot reconoce la matriz de estados $S = [0, -2, 2, 2, -2]$. Pulibot puede tener un programa que, al reconocer esta matriz, establece el color de la celda actual a $Z = 1$ y luego se mueve hacia el este, como se muestra en el centro y a la derecha de la figura:

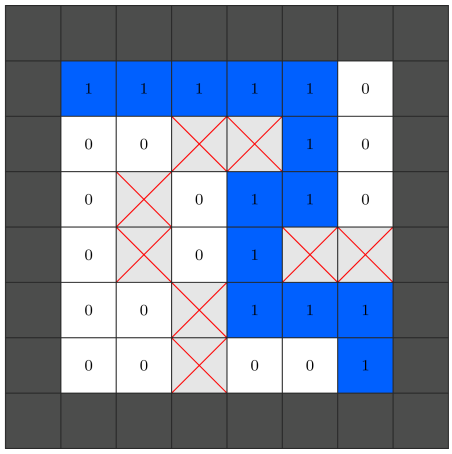


Reglas del Concurso de Robots

- Al inicio, Pulibot se coloca en la celda $(0, 0)$ y comienza a ejecutar su programa.
- Pulibot no puede moverse a una celda que no esté vacía.
- El programa de Pulibot debe terminar después de un máximo de 500.000 pasos.
- Después de la terminación del programa de Pulibot, las celdas vacías en el laberinto deben ser de color tal que:
    - Existe un camino más corto desde $(0, 0)$ hasta $(H - 1, W - 1)$ para el cual el color de cada celda incluida en el camino es 1.
    - El color de cualquier otra celda vacía es 0.
- Pulibot puede terminar su programa en cualquier celda vacía.

Por ejemplo, la siguiente figura muestra un laberinto posible con $H = W = 6$. La configuración inicial se muestra a la izquierda y la coloración esperada de las celdas vacías después de la terminación se muestra a la derecha:





Detalles de implementación

Debes implementar el siguiente procedimiento.

```
void program_pulibot()
```

- Este procedimiento debería producir el programa de Pulibot. Este programa debería funcionar correctamente para todos los valores de H y W y cualquier laberinto que cumpla las restricciones de la tarea.
- Este procedimiento es llamado exactamente una vez para cada caso de prueba.

Este procedimiento puede hacer llamadas al siguiente procedimiento para producir el programa de Pulibot:

```
void set_instruction(int[] S, int Z, char A)
```

- S: matriz de longitud 5 que describe una matriz de estados.
- Z: un entero no negativo que representa un color.
- A: un único carácter que representa una acción de Pulibot como sigue:
  - `H`: quedarse;
  - `W`: moverse hacia el oeste;
  - `S`: moverse hacia el sur;
  - `E`: moverse hacia el este;
  - `N`: moverse hacia el norte;
  - `T`: terminar el programa.
- Llamar a este procedimiento instruye a Pulibot que al reconocer el estado S debe realizar la instrucción (Z, A).

Llamar a este procedimiento múltiples veces con el mismo estado S resultará en un veredicto de `Output isn't correct`.

No es necesario llamar a `set_instruction` con cada posible matriz de estados S. Sin embargo, si Pulibot reconoce más tarde un estado para el que no se estableció una instrucción, obtendrá un veredicto de `Output isn't correct`.

Después de que `program_pulibot`se complete, el calificador invoca el programa de Pulibot sobre uno o más laberintos. Estas invocaciones *no* cuentan para el límite de tiempo de su solución. El calificador es *no* adaptativo, es decir, el conjunto de laberintos está predefinido en cada caso de prueba.

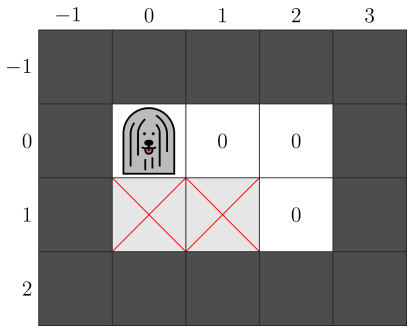
Si Pulibot viola alguna de las Reglas del Concurso de Robots antes de terminar su programa, obtendrá un veredicto de `Output isn't correct` .

Ejemplo

El procedimiento `program_pulibot` puede hacer llamadas a `set_instruction` como sigue:

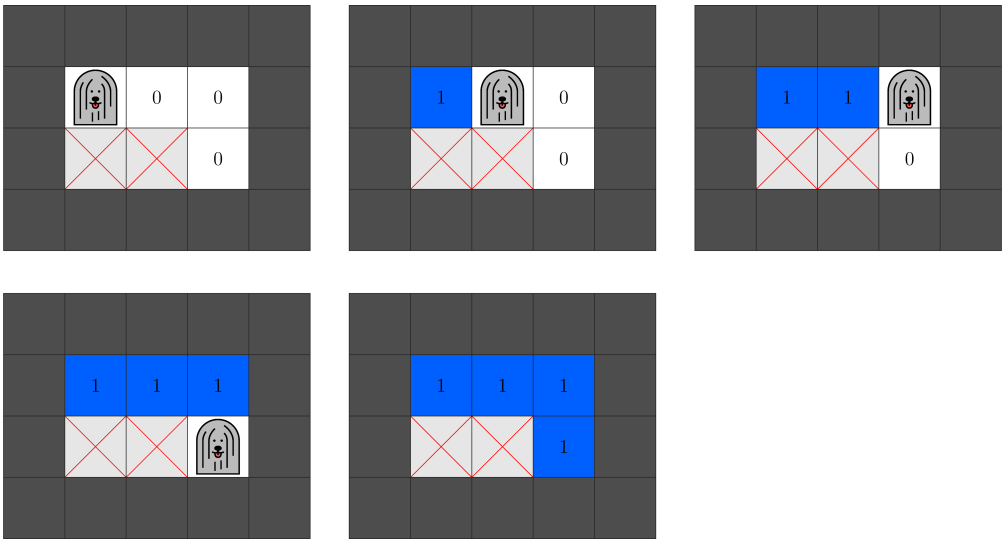
Llamada a	para la matriz de estados S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Poner color a 1 y mover al este
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Poner color a 1 y mover al este
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Poner color a 1 y mover al sur
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Poner color a 1 y terminar el programa

Consideremos un escenario donde $H = 2$ y $W = 3$, y el laberinto se muestra en la siguiente figura.



Para este laberinto en particular, el programa de Pulibot se ejecuta en cuatro pasos. Las matrices de estado que reconoce Pulibot las instrucciones que ejecuta corresponden exactamente a las cuatro llamadas a `set_instruction` hechas arriba, en orden. La última de estas instrucciones termina el programa.

La siguiente figura muestra el estado del laberinto antes de cada uno de los cuatro pasos y su estado final después de la terminación.



Sin embargo, tenga en cuenta que este programa de 4 instrucciones podría no encontrar el camino más corto en otros laberintos válidos. Por lo tanto, si se envía, recibirá un veredicto de `Output isn't correct` verdict.

Restriciones

$Z_{MAX} = 19$. Por lo tanto, Pulibot puede utilizar colores de 0 a 19, ambos inclusive.

Para cada laberinto utilizado para probar Pulibot:

- $2 \leq H, W \leq 15$
- Hay al menos un camino desde la celda $(0, 0)$ a la celda $(H - 1, W - 1)$.

Subtareas

1. (6 puntos) No hay ninguna celda obstáculo en el laberinto.
2. (10 puntos) $H = 2$
3. (18 puntos) Hay exactamente un camino entre cada par de celdas vacías.
4. (20 puntos) El camino más corto desde la celda $(0, 0)$ a la celda $(H - 1, W - 1)$ tiene longitud $H + W - 2$.
5. (46 puntos) No hay restricciones adicionales.

Si, en alguno de los casos de prueba, las llamadas al procedimiento `set_instruction` o el programa de Pulibot sobre su ejecución no se ajustan a las restricciones descritas en Detalles de la implementación, la puntuación de tu solución para esa subtarea será de 0.

En cada subtarea, puedes obtener una puntuación parcial produciendo una coloración que sea casi correcta.

Formalmente:

- La solución de un caso de prueba está **completa** si la coloración final de las celdas vacías cumple las reglas del concurso de robots.
- La solución de un caso de prueba es **parcial** si la coloración final tiene el siguiente aspecto:
  - Existe un camino más corto de $(0, 0)$ a $(H - 1, W - 1)$ para el que el color de cada celda incluida en el camino es 1.
  - No hay ninguna otra celda vacía en la rejilla con color 1.
  - Alguna celda vacía de la rejilla tiene un color distinto de 0 y 1.

Si su solución a un caso de prueba no es completa ni parcial, su puntuación para el caso de prueba correspondiente será de 0. correspondiente será de 0.

En las subtareas 1-4, obtendrá el 50% de los puntos si su solución es parcial.

En la subtarea 5, tu puntuación depende del número de colores utilizados en el programa de Pulibot. Más concretamente, denotemos por $Z^\star$ el valor máximo de Z sobre todas las llamadas realizadas a `set_instruction`. La puntuación del caso de prueba se calcula de acuerdo con la siguiente tabla:

Condición	Puntuación (completa)	Puntuación (parcial)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

La puntuación de cada subtarea es el mínimo de los puntos de los casos de prueba de la subtarea.

Ejemplo del calificador Grader

El ejemplo del calificador lee la entrada en el siguiente formato:

- línea 1: H W
- línea $2 + r$ ($0 \leq r < H$): $m[r][0]$ $m[r][1]$ $\ldots$ $m[r][W - 1]$

Aquí, m es una matriz de matrices H de enteros W, que describen las celdas no limítrofes del laberinto. $m[r][c] = 0$ si la celda (r, c) es una celda vacía y $m[r][c] = 1$ si la celda (r, c) es una celda obstáculo.

El calificador de ejemplo primero llama a `program_pulibot()`. Si el calificador de ejemplo detecta una violación del protocolo, el calificador de ejemplo imprime `Protocol Violation: <MSG>` y termina, donde `<MSG>` es uno de los siguientes mensajes de error:

- `Invalid array`: $-2 \leq S[i] \leq Z_{MAX}$ no se cumple para algunos i o la longitud de Sno es 5.
- `Invalid color`: $0 \leq Z \leq Z_{MAX}$no se cumple.
- `Invalid action`: el carácter A no es uno de los siguientes: `H`, `W`, `S`, `E`, `N` or `T`.
- `Same state array`: `set_instruction` Se ha llamado con el mimo array S al menos dos veces.

De lo contrario, cuando `program_pulibot` se completa, el calificador de ejemplo ejecuta el programa de Pulibot en el laberinto descrito por la entrada.

El calificador de ejemplo produce dos salidas.

En primer lugar, el calificador de muestra escribe un registro de las acciones de Pulibot en el archivo `robot.bin` del directorio de trabajo. Este archivo sirve como entrada de la herramienta de visualización descrita en la siguiente sección.

En segundo lugar, si el programa de Pulibot no termina con éxito, el clasificador de muestras imprime uno de los siguientes mensajes de error:

- `Unexpected state`: Pulibot reconoció un array de estados con el que no se llamó a `set_instruction`.
- `Invalid move`: al realizar una acción, Pulibot se ha movido a una celda no vacía.
- `Too many steps`: Pulibot realizó $500\,000$ pasos sin terminar su programa.

En otro caso, sea $e[r][c]$ el estado de la celda (r, c) después de terminar el programa de Pulibot. El clasificador de ejemplo imprime H llíneas en el siguiente formato:

- Línea $1 + r$ ($0 \leq r < H$): $e[r][0]$ $e[r][1]$ $\ldots$ $e[r][W-1]$

Herramienta de visualización

El paquete adjunto para esta tarea contiene un archivo llamado `display.py`. Cuando se invoca, este script Python muestra las acciones de Pulibot en el laberinto descrito por la entrada del clasificador de ejemplo. Para ello, el archivo binario `robot.bin` debe estar presente en el directorio de trabajo.

Para invocar el script, ejecute el siguiente comando.

```
python3 display.py
```

Aparecerá una sencilla interfaz gráfica. Las principales características son las siguientes:

- Se puede observar el estado del laberinto completo. La ubicación actual de Pulibot se resalta con un rectángulo.
- Puedes navegar a través de los pasos de Pulibot haciendo clic en los botones de flecha o pulsando sus teclas de acceso rápido. También puedes saltar a un paso específico.

- El próximo paso en el programa de Pulibot se muestra en la parte inferior. Muestra la matriz de estado actual y la instrucción que realizará. Después del paso final, muestra uno de los mensajes de error del calificador, o `Terminated` si el programa termina con éxito.
- A cada número que representa un color, se le puede asignar un color visual de fondo, así como un texto. El texto de visualización es una cadena corta que se escribirá en cada celda que tenga el mismo color. Puede asignar colores de fondo y textos de visualización de cualquiera de las siguientes maneras:
    - Estableciéndolos en una ventana de diálogo tras pulsar el botón `Colors`.
    - Editar el contenido del fichero `colors.txt`.
- Para recargar `robot.bin`, utilice el botón `Reload`. Es útil si el contenido de `robot.bin` ha cambiado.