Competencia de Robots

Investigadores de IA en la Universidad de Szeged están llevando a cabo un concurso de programación de robots. Tu amiga, Hanga, ha decidido participar en el concurso. El objetivo es programar al mejor *Pulibot*, admirando la gran inteligencia de la famosa raza de perro húngara, el Puli.

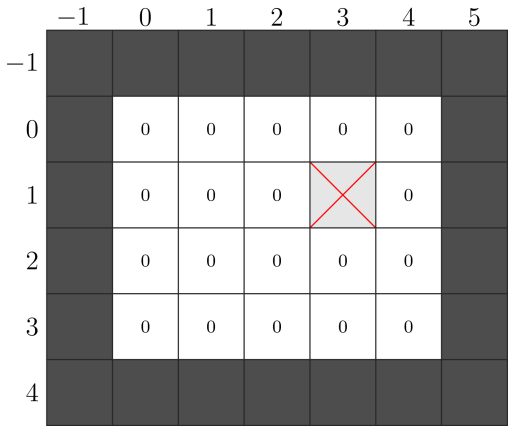
El Pulibot será probado en un laberinto que consiste de una cuadrícula de $(H+2) \times (W+2)$ celdas. Las filas de la cuadrícula son numeradas de -1 a H de norte a sur y las columnas de la cuadrícula son numeradas de -1 a W de oeste a este. Identificamos a la celda ubicada en la fila r y la columna 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 celdas **adyacentes** a la celda (r, c):

- Nos referimos a la celda $(r, c-1)$ como la celda al **oeste** de la celda (r, c);
- Nos referimos a la celda $(r+1, c)$ como la celda al **sur** de la celda (r, c);
- Nos referimos a la celda $(r, c+1)$ como la celda al **este** de la celda (r, c);
- Nos referimos a la celda $(r-1, c)$ como la celda al **norte** de la celda (r, c).

La celda (r, c) es llamada una celda **limite** del laberinto si $r = -1$ o $r = H$ o $c = -1$ o $c = W$. Cada celda que no sea limite en el laberinto es, ó una celda **obstáculo**, ó una celda **vacía**. Adicionalmente, cada celda vacía tiene un **color**, representado por un numero entero no negativo entre 0 y Z_{MAX}, inclusive. Inicialmente, el color de cada celda vacía es 0.

Por ejemplo, considera al laberinto con $H = 4$ y $W = 5$, que contiene una única celda obstáculo $(1, 3)$:



La única celda obstáculo se denota con una equis(X). Las celdas límite del laberinto son marcadas de color negro. Los números escritos en cada celda vacía representan sus colores.

Un **camino** de largo ℓ ($\ell > 0$) desde la celda (r_0, c_0) a la celda (r_ℓ, c_ℓ) es una secuencia de celdas *vacías* distintas $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ en la que, para cada i ($0 \leq i < \ell$), las celdas (r_i, c_i) y (r_{i+1}, c_{i+1}) son adyacentes.

Notese que un camino de largo ℓ contiene exactamente $\ell + 1$ celdas.

En la competencia, los investigadores configuraron un laberinto en el que existe al menos un camino de la celda $(0, 0)$ a la celda $(H - 1, W - 1)$. Se garantiza que las celdas $(0, 0)$ y $(H - 1, W - 1)$ están vacías.

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

Tu tarea es ayudar a Hanga a programar a Pulibot para que sea capaz de encontrar el *camino más corto* (es decir, el camino de tamaño mínimo) desde la celda $(0, 0)$ a la celda $(H - 1, W - 1)$ en el laberinto configurado por los investigadores. La especificación de Pulibot y las reglas de la competencia son descritas a continuación.

Nótese que la última sección de la descripción de este problema describe una herramienta visual que puedes usar para ver a Pulibot.

Especificación de Pulibot

Se define al **estado** de la celda (r, c) para todo $-1 \leq r \leq H$ y $-1 \leq c \leq W$ como un entero tal que:

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

El programa de Pulibot es ejecutado como una secuencia de pasos. En cada paso, Pulibot reconoce los estados de las celdas cercanas y luego realiza una instrucción. La instrucción realizada es determinada por los estados reconocidos. Una descripción más precisa se muestra a continuación.

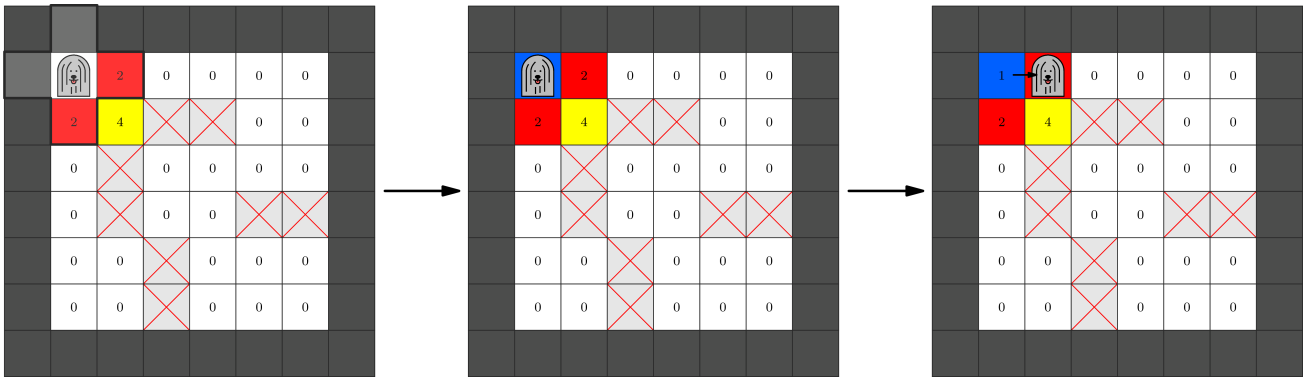
Suponga que al inicio del paso actual, Pulibot se encuentra en la celda (r, c), que es una celda vacía. El paso se realiza de la siguiente manera:

1. Primero, Pulibot reconoce el **arreglo de estado** actual; es decir, el arreglo $S = [S[0], S[1], S[2], S[3], S[4]]$, que consiste del estado de la celda (r, c) y de las todas sus celdas adyacentes:
    - $S[0]$ es el estado de la celda (r, c).
    - $S[1]$ es el estado de la celda al oeste.
    - $S[2]$ es el estado de la celda al sur.
    - $S[3]$ es el estado de la celda al este.

- $S[4]$ es el estado de la celda al norte.

2. Luego, Pulibot determina la **instrucción** (Z, A) que corresponde al arreglo de estado reconocido por el robot.
3. Finalmente, Pulibot realiza la instrucción: ajusta el color de la celda (r, c) al color Z y luego realiza la acción A, que es una de las siguientes acciones:
   - *quedarse* en la celda (r, c)
   - *moverse* a una de las 4 celdas adyacentes
   - *terminar el programa*

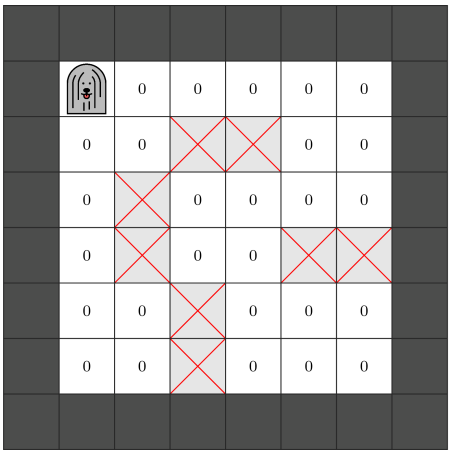
Por ejemplo, considera el escenario mostrado a la izquierda de la siguiente figura. Pulibot se encuentra actualmente en la celda $(0, 0)$ con color 0. Pulibot reconoce el arreglo de estado $S = [0, -2, 2, 2, -2]$. Pulibot puede tener un programa que, al reconocer este arreglo, ajusta el color de la celda actual a $Z = 1$ y luego se mueva hacia el este, como se muestra al medio y a la derecha de la figura:

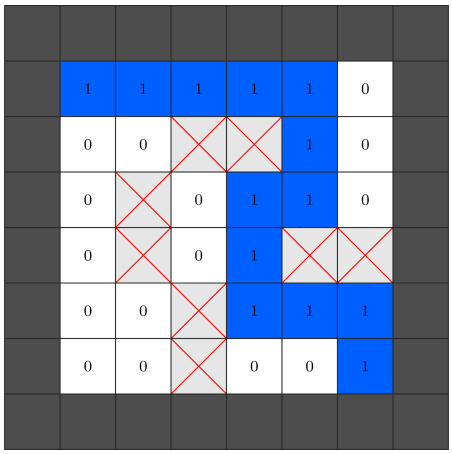


Reglas de la Competencia de Robots

- Al inicio, Pulibot es ubicado en la celda $(0, 0)$ e inicia la ejecución de su programa.
- No se permite que Pulibot se mueva a una celda que no esté vacía.
- El programa de Pulibot debe terminar después de a lo sumo $500\,000$ pasos.
- Después de la finalización del programa de Pulibot, las celdas vacías en el laberinto deben estar coloreadas tal que:
  - Existe un camino más corto de $(0, 0)$ a $(H - 1, W - 1)$ para el cual el color de cada celda incluida en el camino es 1.
  - El color de todas las demás celdas vacías 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 una forma aceptable de colorear las celdas vacías después de la finalización del programa se muestra a la derecha.





Detalles de implementación

Debes implementar la siguiente función:

```
void program_pulibot()
```

- Esta función debe producir el programa de Pulibot. Este programa debe funcionar correctamente para todos los valores de H y W y para cualquier laberinto que cumpla las restricciones del problema.
- Esta función es llamada exactamente una vez para cada caso de prueba.

Esta función puede hacer llamadas a la siguiente función para producir el programa de Pulibot:

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

- S: arreglo de tamaño 5 describiendo el arreglo de estado.
- Z: un entero no negativo representando un color.
- A: un único carácter representando la acción de Pulibot de la siguiente manera:
  - `H`: quedarse;
  - `W`: moverse al oeste;
  - `S`: moverse al sur;
  - `E`: moverse al este;
  - `N`: moverse al norte;
  - `T`: terminar el programa.
- Llamar a esta función indica a Pulibot que al reconocer el estado S, debe realizar la instrucción (Z, A)

Si llamas a esta función múltiples veces con el mismo estado S obtendras el veredicto `Output isn't correct`.

No es requerido llamar a `set_instruction` para cada posible arreglo de estado S. Sin embargo, si mas tarde Pulibot reconoce un estado para el cual una instrucción no ha sido indicada,

obtendrás el veredicto `Output isn't correct`.

Después que `program_pulibot` termine, el evaluador de ejemplo ejecutará el programa de Pulibot para uno o más laberintos. Estas ejecuciones *no* son contadas para el tiempo límite de tu solución. El evaluador de ejemplo *no es* adaptativo; es decir, el conjunto de laberintos esta predefinido para cada caso de prueba.

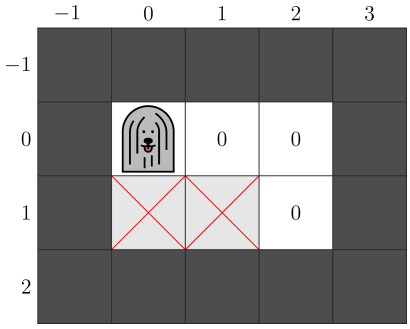
Si Pulibot viola alguna de las Reglas de la Competencia de Robots antes de terminar su programa, obtendrás el veredicto `Output isn't correct`.

Ejemplo

La función `program_pulibot` podria realizar llamadas a `set_instruction` de la siguiente manera:

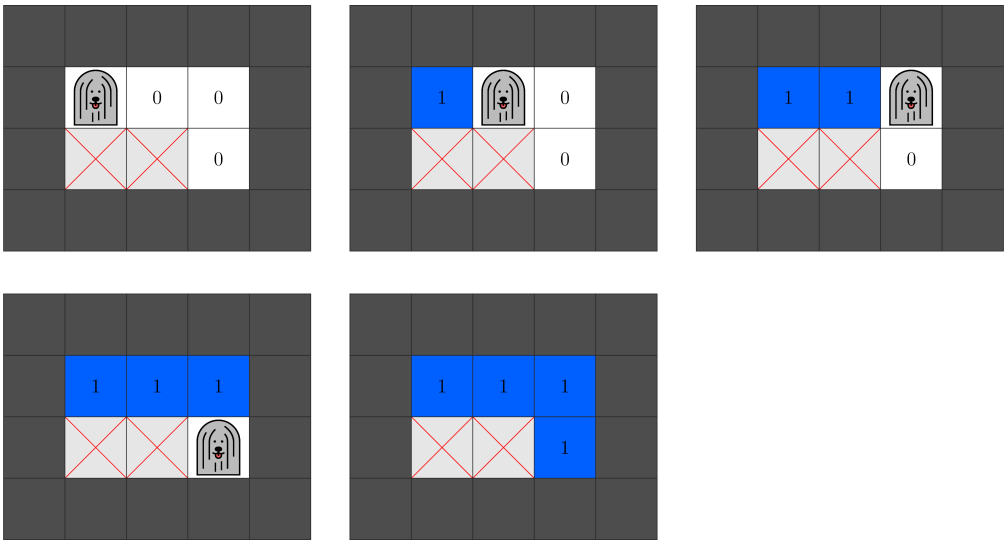
Llamada	Instrucción para el estado S
`set_instruction([0, -2, -1, 0, -2], 1, E)`	Ajustar el color a 1 y moverse al este
`set_instruction([0, 1, -1, 0, -2], 1, E)`	Ajustar el color a 1 y moverse al este
`set_instruction([0, 1, 0, -2, -2], 1, S)`	Ajustar el color a 1 y moverse al sur
`set_instruction([0, -1, -2, -2, 1], 1, T)`	Ajustar el color a 1 y terminar el programa

Considera el escenario donde $H = 2$ y $W = 3$, y el laberinto que se muestra en la siguiente figura:



Para este laberinto en particular, el programa de Pulibot se ejecuta en cuatro pasos. Los arreglos de estado que Pulibot reconoce y las instrucciones que realiza corresponden exactamente a las cuatro llamadas a `set_instruction` realizadas anteriormente, en orden. La última de dichas 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 terminar.



Sin embargo, nota que este programa de 4 instrucciones puede no encontrar el camino más corto en otros laberintos válidos. Por lo tanto, de ser enviado, recibirá un veredicto `Output isn't correct`.

Restricciones

$Z_{MAX} = 19$. Por lo tanto, Pulibot puede usar los colores del 0 al 19, inclusive.

Para cada laberinto usado para evaluar a Pulibot:

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

Subtareas

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

Si en cualquiera de los casos de prueba, las llamadas a la función `set_instruction` o el programa de Pulibot al ejecutarse no cumplen las restricciones descritas en los Detalles de Implementación, el puntaje de tu solución para esa subtarea será 0.

En cada subtarea, puedes obtener una puntuación parcial si produces una forma de colorear que es casi correcta.

Formalmente:

- La solución de un caso de prueba es **completa** si la forma de colorear final de las celdas vacías satisface las Reglas de la Competencia de Robots.
- La solución de un caso de prueba es **parcial** si la forma de colorear final cumple con:
  - Existe un camino más corto desde $(0, 0)$ a $(H - 1, W - 1)$ para el cual el color de cada celda incluida en el camino es 1.
  - No existe otra celda vacía en la cuadrícula con color 1.
  - Alguna celda vacía en la cuadrícula tiene un color diferente a 0 y 1.

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

En las subtareas 1-4, el puntaje para una solucion completa es 100% y el puntaje para una solucion parcial es el 50% de los puntos para el subtask.

En la subtarea 5, tu puntuación dependerá del número de colores utilizados en el programa de Pulibot. Más precisamente, denotamos por $Z^\star$ al máximo valor de Z sobre todas las llamadas hechas a `set_instruction`. La puntuación del caso de prueba es calculada de acuerdo a 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 puntos obtenidos en los casos de prueba de la subtarea.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada siguiendo 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 un arreglo de H arreglos de W enteros, describiendo a las celdas no límite 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 evaluador de ejemplo primero llama a `program_pulibot()`. Si el evaluador de ejemplo detecta una violación al protocolo, el evaluador de ejemplo imprimirá `Protocol Violation: <MSG>` y terminará, donde `<MSG>` es uno de los siguientes mensajes de error:

- `Invalid array`: no se cumple $-2 \leq S[i] \leq Z_{MAX}$ para alguna i o el largo de S no es 5.
- `Invalid color`: no se cumple $0 \leq Z \leq Z_{MAX}$.
- `Invalid action`: El carácter A no es alguno de `H`, `W`, `S`, `E`, `N` o `T`.
- `Same state array`: `set_instruction` fue llamado con el mismo arreglo S al menos dos veces.

De lo contrario, cuando `program_pulibot` termina, el evaluador de ejemplo ejecutará el programa de Pulibot en el laberinto descrito por la entrada.

El evaluador de ejemplo producirá dos salidas.

Primero, el evaluador de ejemplo escribirá un registro de las acciones de Pulibot al archivo `robot.bin` en la carpeta actual. Este archivo sirve como entrada para la herramienta de visualización descrita en la siguiente sección.

Segundo, si el programa de Pulibot no termina satisfactoriamente, el evaluador de ejemplo imprime uno de los siguientes mensajes de error:

- `Unexpected state`: Pulibot reconoció un arreglo de estado para el cual `set_instruction` no fue llamada.
- `Invalid move`: realizar una acción resultó que Pulibot se moviera a una celda no vacía.
- `Too many steps`: Pulibot realizó $500\,000$ pasos sin terminar su programa.

De lo contrario, sea $e[r][c]$ el estado de la celda (r, c) después de la finalización del programa de Pulibot. El evaluador de ejemplo imprime H líneas en el formato siguiente:

- 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 es invocado, este script de Python muestra las acciones de Pulibot en el laberinto descrito por la entrada del evaluador de ejemplo. Para esto, el archivo `robot.bin` debe estar presente en la carpeta actual.

Para invocar al script, ejecuta el siguiente comando.

```
python3 display.py
```

Una interfaz gráfica simple aparece. Las características principales son:

- Puedes observar el estatus del laberinto completo. La ubicación actual de Pulibot es marcada por un rectángulo.
- Puedes navegar a través de los pasos de Pulibot haciendo click en los botones con flecha o presionando sus teclas correspondientes. También puedes saltar a un paso en específico.
- El siguiente paso en el programa de Pulibot se muestra en la parte inferior. Muestra el arreglo de estado actual y la instrucción que realizará. Después del paso final, muestra ya sea uno de los mensajes de error del evaluador de ejemplo, o `Terminated` si el programa termina satisfactoriamente.
- Para cada número que representa un color, puedes asignar un color de fondo visual, así como texto desplegado. El texto es una cadena corta que debe ser escrita en cada celda que tenga el mismo color. Puedes asignar colores de fondo y textos desplegados en una de las siguientes maneras:
    - Ajustarlos en una ventana de diálogo después de hacer click en el botón `Colors`.
    - Editar el contenido del archivo `colors.txt`.
- Para recargar `robot.bin`, utiliza el botón `Reload`. Esto es útil si el contenido de `robot.bin` ha sido actualizado.