Proyecto Aprendizaje por Refuerzo

Entrega Parcial: Ambiente, Estados y Acciones

Antonio Miguel Porcelli Sposaro - Dubán Leandro Jiménez Puerta

1. Introducción

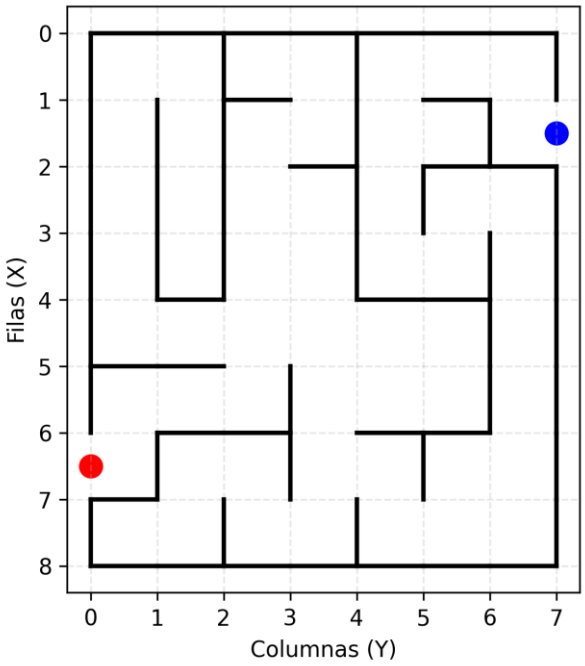
El Aprendizaje por Refuerzo (*Reinforcement Learning*) constituye una de las áreas más relevantes del campo del Aprendizaje Automático, al centrarse en la **interacción entre un agente y su entorno** mediante un proceso de **toma de decisiones secuencial**. A diferencia de los enfoques supervisados, donde el aprendizaje se basa en ejemplos con etiquetas, en *RL* el conocimiento se adquiere a través de la experiencia directa: el agente ejecuta acciones, observa las consecuencias y ajusta su comportamiento para **maximizar** la recompensa acumulada.

En este contexto, los entornos *GridWorld* proporcionan un **escenario simplificado** pero ilustrativo para estudiar los fundamentos del *Reinforcement Learning*. Estos entornos discretos permiten modelar conceptos de **estado**, **acción** y **recompensa**, así como analizar la evolución del aprendizaje mediante **políticas** y **funciones de valor**.

El presente trabajo implementa un agente que interactúa dentro de un entorno laberinto de 8×7 celdas, en el cual debe aprender a alcanzar una meta optimizando su comportamiento mediante la definición de una función de recompensas y un conjunto de acciones posibles.

2. Propósito del Agente

El propósito del agente consiste en aprender una política óptima que maximice la recompensa acumulada al desplazarse dentro del entorno laberinto. Desde un **estado inicial** (*punto rojo*), el agente puede ejecutar acciones que modifican su posición dentro de la cuadrícula de 8×7 . El desafío consiste en descubrir, mediante exploración y aprendizaje, la secuencia de acciones (política) que maximiza la recompensa final. A continuación, se presenta el *GridWorld* a utilizar:



3. Objetivo

El objetivo en este punto intermedio del desarrollo del proyecto es **caracterizar los elementos que componen la solución del problema de navegación del agente en el entorno *GridWorld***. Esto implica definir los elementos del entorno de *Reinforcement Learning:* el conjunto de estados, las acciones posibles y la función de recompensa; siendo estos los que determinan el comportamiento del agente y la forma en que aprende su tarea: alcanzar la meta.

4. Desarrollo

4.1. Definición de las acciones

El agente puede moverse en este entorno discreto ejecutando una de las 4 acciones posibles:

$$\mathcal{A}: \{'up', 'down', 'left', 'right'\}$$

Donde cada acción representa un **desplazamiento unitario** en la cuadrícula. Dado el entorno, este conjunto es suficiente para que el agente explore y aprenda mediante la interacción.

4.2. Definición de los estados

El conjunto de **estados** representa todas las posiciones posibles que el agente puede ocupar dentro del entrono discreto del laberinto. Cada estado se modela como un par $(x,\ y)$, donde x es la fila e y la columna dentro de la cuadrícula $m \times n$ (8×7).

Formalmente, el conjunto de **estados** se define como:

$$\mathcal{S}\colon \{(x,\ y) \mid x \in [0,\ 7],\ y \in [0,\ 6]\}$$

Lo que equivale a:

$\mathcal{S}$: {(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 0), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (3, 6), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6), (7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6)}

4.3. Definición de las acciones aplicables a cada estado y Función de recompensa

La dinámica del entorno se representa mediante tuplas $(s,\ a,\ s',\ r)$ que describen cómo cambia la situación del agente después de ejecutar una acción y qué retroalimentación obtiene del entorno.

Este caso, todas las acciones – *up, down, left, right* – están siempre disponibles, pero la recompensa depende del **resultado** de la acción:

- -5 Si el movimiento intenta atravesar una pared o salir del entorno (*ej. Moverse a la izquierda desde la posición inicial*)
- -1 Para movimientos válidos dentro del laberinto, buscando reforzar la completación del laberinto en la menor cantidad de pasos posibles.
- $+100$ Al alcanzar el estado final o meta.

La lógica detrás de esta definición busca lograr dos objetivos:

1. **Dar una señal inmediata y clara** cuando el agente comete un error. Si una acción es ilegal, el agente permanece en el mismo estado y recibe una penalización. Esto le ayuda a aprender rápidamente qué acciones debe evitar.
2. **Mantener la simplicidad del espacio de acciones.** El propio proceso de aprendizaje ajusta las probabilidades de acción: las que llevan a paredes recibirán penalizaciones y serán descartadas con el tiempo.

A continuación, definen explícitamente las acciones que puede ejecutar el agente, así como las recompensas asociadas:

Rewards

S	action	S'	reward
(0, 0)	up	(0, 0)	-5
(0, 0)	down	(1, 0)	-1
(0, 0)	left	(0, 0)	-5
(0, 0)	right	(0, 1)	-1
(0, 1)	up	(0, 1)	-5
(0, 1)	down	(1, 1)	-1
(0, 1)	left	(0, 0)	-1
(0, 1)	right	(0, 1)	-5
(0, 2)	up	(0, 2)	-5
(0, 2)	down	(0, 2)	-5
(0, 2)	left	(0, 2)	-5
(0, 2)	right	(0, 3)	-1
(0, 3)	up	(0, 3)	-5
(0, 3)	down	(1, 3)	-1
(0, 3)	left	(0, 2)	-1
(0, 3)	right	(0, 3)	-5
(0, 4)	up	(0, 4)	-5
(0, 4)	down	(1, 4)	-1
(0, 4)	left	(0, 4)	-5
(0, 4)	right	(0, 5)	-1
(0, 5)	up	(0, 5)	-5
(0, 5)	down	(0, 5)	-5
(0, 5)	left	(0, 4)	-1
(0, 5)	right	(0, 6)	-1
(0, 6)	up	(0, 6)	-5
(0, 6)	down	(1, 6)	-1
(0, 6)	left	(0, 5)	-1
(0, 6)	right	(0, 6)	-5
(1, 0)	up	(0, 0)	-1
(1, 0)	down	(2, 0)	-1
(1, 0)	left	(1, 0)	-5
(1, 0)	right	(1, 0)	-5
(1, 1)	up	(0, 1)	-1
(1, 1)	down	(2, 1)	-1
(1, 1)	left	(1, 1)	-5
(1, 1)	right	(1, 1)	-5
(1, 2)	up	(1, 2)	-5
(1, 2)	down	(2, 2)	-1
(1, 2)	left	(1, 2)	-5
(1, 2)	right	(1, 3)	-1
(1, 3)	up	(0, 3)	-1
(1, 3)	down	(1, 3)	-5
(1, 3)	left	(1, 2)	-1
(1, 3)	right	(1, 3)	-5

(1, 4)	up	(0, 4)	-1
(1, 4)	down	(2, 4)	-1
(1, 4)	left	(1, 4)	-5
(1, 4)	right	(1, 5)	-1
(1, 5)	up	(1, 5)	-5
(1, 5)	down	(1, 5)	-5
(1, 5)	left	(1, 4)	-1
(1, 5)	right	(1, 5)	-5
(1, 6)	up	(0, 6)	-1
(1, 6)	down	(1, 6)	-5
(1, 6)	left	(1, 6)	-5
(1, 6)	right	(1, 7)	100
(2, 0)	up	(1, 0)	-1
(2, 0)	down	(3, 0)	-1
(2, 0)	left	(2, 0)	-5
(2, 0)	right	(2, 0)	-5
(2, 1)	up	(1, 1)	-1
(2, 1)	down	(3, 1)	-1
(2, 1)	left	(2, 1)	-5
(2, 1)	right	(2, 1)	-5
(2, 2)	up	(1, 2)	-1
(2, 2)	down	(3, 2)	-1
(2, 2)	left	(2, 2)	-5
(2, 2)	right	(2, 3)	-1
(2, 3)	up	(2, 3)	-5
(2, 3)	down	(3, 3)	-1
(2, 3)	left	(2, 2)	-1
(2, 3)	right	(2, 3)	-5
(2, 4)	up	(1, 4)	-1
(2, 4)	down	(3, 4)	-1
(2, 4)	left	(2, 4)	-5
(2, 4)	right	(2, 4)	-5
(2, 5)	up	(2, 5)	-5
(2, 5)	down	(3, 5)	-1
(2, 5)	left	(2, 5)	-5
(2, 5)	right	(2, 6)	-1
(2, 6)	up	(2, 6)	-5
(2, 6)	down	(3, 6)	-1
(2, 6)	left	(2, 5)	-1
(2, 6)	right	(2, 6)	-5
(3, 0)	up	(2, 0)	-1
(3, 0)	down	(4, 0)	-1
(3, 0)	left	(3, 0)	-5
(3, 0)	right	(3, 0)	-5
(3, 1)	up	(2, 1)	-1
(3, 1)	down	(3, 1)	-5
(3, 1)	left	(3, 1)	-5
(3, 1)	right	(3, 1)	-5
(3, 2)	up	(2, 2)	-1
(3, 2)	down	(4, 2)	-1

(3, 2)	left	(3, 2)	-5
(3, 2)	right	(3, 3)	-1
(3, 3)	up	(2, 3)	-1
(3, 3)	down	(4, 3)	-1
(3, 3)	left	(3, 2)	-1
(3, 3)	right	(3, 3)	-5
(3, 4)	up	(2, 4)	-1
(3, 4)	down	(3, 4)	-5
(3, 4)	left	(3, 4)	-5
(3, 4)	right	(3, 5)	-1
(3, 5)	up	(2, 5)	-1
(3, 5)	down	(3, 5)	-5
(3, 5)	left	(3, 4)	-1
(3, 5)	right	(3, 5)	-5
(3, 6)	up	(2, 6)	-1
(3, 6)	down	(4, 6)	-1
(3, 6)	left	(3, 6)	-5
(3, 6)	right	(3, 6)	-5
(4, 0)	up	(3, 0)	-1
(4, 0)	down	(4, 0)	-5
(4, 0)	left	(4, 0)	-5
(4, 0)	right	(4, 1)	-1
(4, 1)	up	(4, 1)	-5
(4, 1)	down	(4, 1)	-5
(4, 1)	left	(4, 0)	-1
(4, 1)	right	(4, 2)	-1
(4, 2)	up	(3, 2)	-1
(4, 2)	down	(5, 2)	-1
(4, 2)	left	(4, 1)	-1
(4, 2)	right	(4, 3)	-1
(4, 3)	up	(3, 3)	-1
(4, 3)	down	(5, 3)	-1
(4, 3)	left	(4, 2)	-1
(4, 3)	right	(4, 4)	-1
(4, 4)	up	(4, 4)	-5
(4, 4)	down	(5, 4)	-1
(4, 4)	left	(4, 3)	-1
(4, 4)	right	(4, 5)	-1
(4, 5)	up	(4, 5)	-5
(4, 5)	down	(5, 5)	-1
(4, 5)	left	(4, 4)	-1
(4, 5)	right	(4, 5)	-5
(4, 6)	up	(3, 6)	-1
(4, 6)	down	(5, 6)	-1
(4, 6)	left	(4, 6)	-5
(4, 6)	right	(4, 6)	-5
(5, 0)	up	(5, 0)	-5
(5, 0)	down	(6, 0)	-1
(5, 0)	left	(5, 0)	-5
(5, 0)	right	(5, 1)	-1

(5, 1)	up	(5, 1)	-5
(5, 1)	down	(5, 1)	-5
(5, 1)	left	(5, 0)	-1
(5, 1)	right	(5, 2)	-1
(5, 2)	up	(4, 2)	-1
(5, 2)	down	(5, 2)	-5
(5, 2)	left	(5, 1)	-1
(5, 2)	right	(5, 2)	-5
(5, 3)	up	(4, 3)	-1
(5, 3)	down	(6, 3)	-1
(5, 3)	left	(5, 3)	-5
(5, 3)	right	(5, 4)	-1
(5, 4)	up	(4, 4)	-1
(5, 4)	down	(5, 4)	-5
(5, 4)	left	(5, 3)	-1
(5, 4)	right	(5, 5)	-1
(5, 5)	up	(4, 5)	-1
(5, 5)	down	(5, 5)	-5
(5, 5)	left	(5, 4)	-1
(5, 5)	right	(5, 5)	-5
(5, 6)	up	(4, 6)	-1
(5, 6)	down	(6, 6)	-1
(5, 6)	left	(5, 6)	-5
(5, 6)	right	(5, 6)	-5
(6, 0)	up	(5, 0)	-1
(6, 0)	down	(6, 0)	-5
(6, 0)	left	(6, 0)	-5
(6, 0)	right	(6, 0)	-5
(6, 1)	up	(6, 1)	-5
(6, 1)	down	(7, 1)	-1
(6, 1)	left	(6, 1)	-5
(6, 1)	right	(6, 2)	-1
(6, 2)	up	(6, 2)	-5
(6, 2)	down	(7, 2)	-1
(6, 2)	left	(6, 1)	-1
(6, 2)	right	(6, 2)	-5
(6, 3)	up	(5, 3)	-1
(6, 3)	down	(7, 3)	-1
(6, 3)	left	(6, 3)	-5
(6, 3)	right	(6, 4)	-1
(6, 4)	up	(6, 4)	-5
(6, 4)	down	(7, 4)	-1
(6, 4)	left	(6, 3)	-1
(6, 4)	right	(6, 4)	-5
(6, 5)	up	(6, 5)	-5
(6, 5)	down	(7, 5)	-1
(6, 5)	left	(6, 5)	-5
(6, 5)	right	(6, 6)	-1
(6, 6)	up	(5, 6)	-1
(6, 6)	down	(7, 6)	-1

(6, 6)	left	(6, 5)	-1
(6, 6)	right	(6, 6)	-5
(7, 0)	up	(7, 0)	-5
(7, 0)	down	(7, 0)	-5
(7, 0)	left	(7, 0)	-5
(7, 0)	right	(7, 1)	-1
(7, 1)	up	(6, 1)	-1
(7, 1)	down	(7, 1)	-5
(7, 1)	left	(7, 0)	-1
(7, 1)	right	(7, 1)	-5
(7, 2)	up	(6, 2)	-1
(7, 2)	down	(7, 2)	-5
(7, 2)	left	(7, 2)	-5
(7, 2)	right	(7, 3)	-1
(7, 3)	up	(6, 3)	-1
(7, 3)	down	(7, 3)	-5
(7, 3)	left	(7, 2)	-1
(7, 3)	right	(7, 3)	-5
(7, 4)	up	(6, 4)	-1
(7, 4)	down	(7, 4)	-5
(7, 4)	left	(7, 4)	-5
(7, 4)	right	(7, 5)	-1
(7, 5)	up	(6, 5)	-1
(7, 5)	down	(7, 5)	-5
(7, 5)	left	(7, 4)	-1
(7, 5)	right	(7, 6)	-1
(7, 6)	up	(6, 6)	-1
(7, 6)	down	(7, 6)	-5
(7, 6)	left	(7, 5)	-1
(7, 6)	right	(7, 6)	-5

Rewards = [((0, 0), 'up', (0, 0), -5), ((0, 0), 'down', (1, 0), -1), ((0, 0), 'left', (0, 0), -5), ((0, 0), 'right', (0, 1), -1), ((0, 1), 'up', (0, 1), -5), ((0, 1), 'down', (1, 1), -1), ((0, 1), 'left', (0, 0), -1), ((0, 1), 'right', (0, 1), -5), ((0, 2), 'up', (0, 2), -5), ((0, 2), 'down', (0, 2), -5), ((0, 2), 'left', (0, 2), -5), ((0, 2), 'right', (0, 3), -1), ((0, 3), 'up', (0, 3), -5), ((0, 3), 'down', (1, 3), -1), ((0, 3), 'left', (0, 2), -1), ((0, 3), 'right', (0, 3), -5), ((0, 4), 'up', (0, 4), -5), ((0, 4), 'down', (1, 4), -1), ((0, 4), 'left', (0, 4), -5), ((0, 4), 'right', (0, 5), -1), ((0, 5), 'up', (0, 5), -5), ((0, 5), 'down', (0, 5), -5), ((0, 5), 'left', (0, 4), -1), ((0, 5), 'right', (0, 6), -1), ((0, 6), 'up', (0, 6), -5), ((0, 6), 'down', (1, 6), -1), ((0, 6), 'left', (0, 5), -1), ((0, 6), 'right', (0, 6), -5), ((1, 0), 'up', (0, 0), -1), ((1, 0), 'down', (2, 0), -1), ((1, 0), 'left', (1, 0), -5), ((1, 0), 'right', (1, 0), -5), ((1, 1), 'up', (0, 1), -1), ((1, 1), 'down', (2, 1), -1), ((1, 1), 'left', (1, 1), -5), ((1, 1), 'right', (1, 1), -5), ((1, 2), 'up', (1, 2), -5), ((1, 2), 'down', (2, 2), -1), ((1, 2), 'left', (1, 2), -5), ((1, 2), 'right', (1, 3), -1), ((1, 3), 'up', (0, 3), -1), ((1, 3), 'down', (1, 3), -5), ((1, 3), 'left', (1, 2), -1), ((1, 3), 'right', (1, 3), -5), ((1, 4), 'up', (0, 4), -1), ((1, 4), 'down', (2, 4), -1), ((1, 4), 'left', (1, 4), -5), ((1, 4), 'right', (1, 5), -1), ((1, 5), 'up', (1, 5), -5), ((1, 5), 'down', (1, 5), -5), ((1, 5), 'left', (1, 4), -1), ((1, 5), 'right', (1, 5), -5), ((1, 6), 'up', (0, 6), -1), ((1, 6), 'down',

(1, 6), -5), ((1, 6), 'left', (1, 6), -5), ((1, 6), 'right', (1, 7), 100), ((2, 0), 'up', (1, 0), -1), ((2, 0), 'down', (3, 0), -1), ((2, 0), 'left', (2, 0), -5), ((2, 0), 'right', (2, 0), -5), ((2, 1), 'up', (1, 1), -1), ((2, 1), 'down', (3, 1), -1), ((2, 1), 'left', (2, 1), -5), ((2, 1), 'right', (2, 1), -5), ((2, 2), 'up', (1, 2), -1), ((2, 2), 'down', (3, 2), -1), ((2, 2), 'left', (2, 2), -5), ((2, 2), 'right', (2, 3), -1), ((2, 3), 'up', (2, 3), -5), ((2, 3), 'down', (3, 3), -1), ((2, 3), 'left', (2, 2), -1), ((2, 3), 'right', (2, 3), -5), ((2, 4), 'up', (1, 4), -1), ((2, 4), 'down', (3, 4), -1), ((2, 4), 'left', (2, 4), -5), ((2, 4), 'right', (2, 4), -5), ((2, 5), 'up', (2, 5), -5), ((2, 5), 'down', (3, 5), -1), ((2, 5), 'left', (2, 5), -5), ((2, 5), 'right', (2, 6), -1), ((2, 6), 'up', (2, 6), -5), ((2, 6), 'down', (3, 6), -1), ((2, 6), 'left', (2, 5), -1), ((2, 6), 'right', (2, 6), -5), ((3, 0), 'up', (2, 0), -1), ((3, 0), 'down', (4, 0), -1), ((3, 0), 'left', (3, 0), -5), ((3, 0), 'right', (3, 0), -5), ((3, 1), 'up', (2, 1), -1), ((3, 1), 'down', (3, 1), -5), ((3, 1), 'left', (3, 1), -5), ((3, 1), 'right', (3, 1), -5), ((3, 2), 'up', (2, 2), -1), ((3, 2), 'down', (4, 2), -1), ((3, 2), 'left', (3, 2), -5), ((3, 2), 'right', (3, 3), -1), ((3, 3), 'up', (2, 3), -1), ((3, 3), 'down', (4, 3), -1), ((3, 3), 'left', (3, 2), -1), ((3, 3), 'right', (3, 3), -5), ((3, 4), 'up', (2, 4), -1), ((3, 4), 'down', (3, 4), -5), ((3, 4), 'left', (3, 4), -5), ((3, 4), 'right', (3, 5), -1), ((3, 5), 'up', (2, 5), -1), ((3, 5), 'down', (3, 5), -5), ((3, 5), 'left', (3, 4), -1), ((3, 5), 'right', (3, 5), -5), ((3, 6), 'up', (2, 6), -1), ((3, 6), 'down', (4, 6), -1), ((3, 6), 'left', (3, 6), -5), ((3, 6), 'right', (3, 6), -5), ((4, 0), 'up', (3, 0), -1), ((4, 0), 'down', (4, 0), -5), ((4, 0), 'left', (4, 0), -5), ((4, 0), 'right', (4, 1), -1), ((4, 1), 'up', (4, 1), -5), ((4, 1), 'down', (4, 1), -5), ((4, 1), 'left', (4, 0), -1), ((4, 1), 'right', (4, 2), -1), ((4, 2), 'up', (3, 2), -1), ((4, 2), 'down', (5, 2), -1), ((4, 2), 'left', (4, 1), -1), ((4, 2), 'right', (4, 3), -1), ((4, 3), 'up', (3, 3), -1), ((4, 3), 'down', (5, 3), -1), ((4, 3), 'left', (4, 2), -1), ((4, 3), 'right', (4, 4), -1), ((4, 4), 'up', (4, 4), -5), ((4, 4), 'down', (5, 4), -1), ((4, 4), 'left', (4, 3), -1), ((4, 4), 'right', (4, 5), -1), ((4, 5), 'up', (4, 5), -5), ((4, 5), 'down', (5, 5), -1), ((4, 5), 'left', (4, 4), -1), ((4, 5), 'right', (4, 5), -5), ((4, 6), 'up', (3, 6), -1), ((4, 6), 'down', (5, 6), -1), ((4, 6), 'left', (4, 6), -5), ((4, 6), 'right', (4, 6), -5), ((5, 0), 'up', (5, 0), -5), ((5, 0), 'down', (6, 0), -1), ((5, 0), 'left', (5, 0), -5), ((5, 0), 'right', (5, 1), -1), ((5, 1), 'up', (5, 1), -5), ((5, 1), 'down', (5, 1), -5), ((5, 1), 'left', (5, 0), -1), ((5, 1), 'right', (5, 2), -1), ((5, 2), 'up', (4, 2), -1), ((5, 2), 'down', (5, 2), -5), ((5, 2), 'left', (5, 1), -1), ((5, 2), 'right', (5, 2), -5), ((5, 3), 'up', (4, 3), -1), ((5, 3), 'down', (6, 3), -1), ((5, 3), 'left', (5, 3), -5), ((5, 3), 'right', (5, 4), -1), ((5, 4), 'up', (4, 4), -1), ((5, 4), 'down', (5, 4), -5), ((5, 4), 'left', (5, 3), -1), ((5, 4), 'right', (5, 5), -1), ((5, 5), 'up', (4, 5), -1), ((5, 5), 'down', (5, 5), -5), ((5, 5), 'left', (5, 4), -1), ((5, 5), 'right', (5, 5), -5), ((5, 6), 'up', (4, 6), -1), ((5, 6), 'down', (6, 6), -1), ((5, 6), 'left', (5, 6), -5), ((5, 6), 'right', (5, 6), -5), ((6, 0), 'up', (5, 0), -1), ((6, 0), 'down', (6, 0), -5), ((6, 0), 'left', (6, 0), -5), ((6, 0), 'right', (6, 0), -5), ((6, 1), 'up', (6, 1), -5), ((6, 1), 'down', (7, 1), -1), ((6, 1), 'left', (6, 1), -5), ((6, 1), 'right', (6, 2), -1), ((6, 2), 'up', (6, 2), -5), ((6, 2), 'down', (7, 2), -1), ((6, 2), 'left', (6, 1), -1), ((6, 2), 'right', (6, 2), -5), ((6, 3), 'up', (5, 3), -1), ((6, 3), 'down', (7, 3), -1), ((6, 3), 'left', (6, 3), -5), ((6, 3), 'right', (6, 4), -1), ((6, 4), 'up', (6, 4), -5), ((6, 4), 'down', (7, 4), -1), ((6, 4), 'left', (6, 3), -1), ((6, 4), 'right', (6, 4), -5), ((6, 5), 'up', (6, 5), -5), ((6, 5), 'down', (7, 5), -1), ((6, 5), 'left', (6, 5), -5), ((6, 5), 'right', (6, 6), -1), ((6, 6), 'up', (5, 6), -1), ((6, 6), 'down', (7, 6), -1), ((6, 6), 'left', (6, 5), -1), ((6, 6), 'right', (6, 6), -5), ((7, 0), 'up', (7, 0), -5), ((7, 0), 'down', (7, 0), -5), ((7, 0), 'left', (7, 0), -5), ((7, 0), 'right', (7, 1), -1),

((7, 1), 'up', (6, 1), -1), ((7, 1), 'down', (7, 1), -5), ((7, 1), 'left', (7, 0), -1), ((7, 1), 'right', (7, 1), -5), ((7, 2), 'up', (6, 2), -1), ((7, 2), 'down', (7, 2), -5), ((7, 2), 'left', (7, 2), -5), ((7, 2), 'right', (7, 3), -1), ((7, 3), 'up', (6, 3), -1), ((7, 3), 'down', (7, 3), -5), ((7, 3), 'left', (7, 2), -1), ((7, 3), 'right', (7, 3), -5), ((7, 4), 'up', (6, 4), -1), ((7, 4), 'down', (7, 4), -5), ((7, 4), 'left', (7, 4), -5), ((7, 4), 'right', (7, 5), -1), ((7, 5), 'up', (6, 5), -1), ((7, 5), 'down', (7, 5), -5), ((7, 5), 'left', (7, 4), -1), ((7, 5), 'right', (7, 6), -1), ((7, 6), 'up', (6, 6), -1), ((7, 6), 'down', (7, 6), -5), ((7, 6), 'left', (7, 5), -1), ((7, 6), 'right', (7, 6), -5)]