

TP1

Grupo 9
Sistemas de Inteligencia Artificial
Instituto Tecnológico de Buenos
Aires

El presente trabajo consiste en proporcionar una explicación del código, así como de las referencias empleadas en su desarrollo.

Estructura del estado

Se elige simular el juego de Zokoban.

Se piensa el problema como una matriz donde cada símbolo representa un elemento diferente. En la tabla 1 se presenta la simbología empleada.

Elemento	Símbolo
Personaje	P
Caja	X
Marca	M
Pared	/
piso	(espacio vacío)

Tabla 1. Lista de elementos y símbolos.

Se muestra en la figura 1 la representación del problema empleando los símbolos mencionados para un estado inicial.

	0	1	2	3	4	5	6	7	8	9
0	/	/	/	/	/	/	/	/	/	/
1	/							M	/	/
2	/							X	/	/
3	/								/	/
4	/					X			/	/
5	/								/	/
6	/					P			/	/
7	/								M	/
8	/								/	/
9	/	/	/	/	/	/	/	/	/	/

Figura 1. Representación simbólica de un estado.

Se puede ver en la tabla 1 que no existe el símbolo de la caja activada (cuando la caja se encuentra en la celda objetivo donde se encuentra la marca). Esto sucede porque no es necesario. Para el problema se plantea un mapa (mapa original) donde se encuentran los objetos inmutables (pared, piso y marca), el mapa de estado es una copia de este con el añadido del personaje y las cajas. Cuando el personaje se mueve, se hace referencia al mapa del escenario para determinar si la celda que deja es de piso o marca.

Para una visualización más amigable a la vista se genera una imagen con los sprites correspondientes para cada símbolo. De esta manera se genera un esquema que simula la interfaz gráfica de un videojuego. El resultado se muestra en la tabla 2 y figura 2.

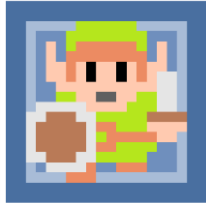
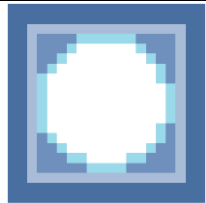
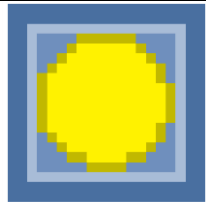
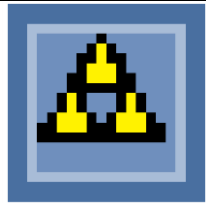

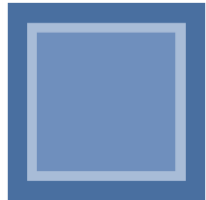
Elemento	Símbolo
Personaje	
Caja	
Caja activada	
Marca	
Pared	
piso	

Tabla 2. Elementos y sprites.

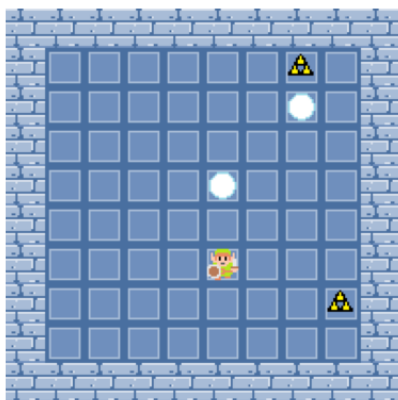


Figura 2. Esquema del estado mediante sprites.

Respecto al código que ejecuta el movimiento, es de invención propia. Se emplearon referencias para definir acciones que, aunque son conocidas, pueden no recordarse debido a la falta de experiencia. Estas son:

- Contar la cantidad de un símbolo en una matriz (<https://stackoverflow.com/questions/8364674/how-to-count-the-number-of-true-elements-in-a-numpy-bool-array>)

- Obtener la posición de los símbolos de una matriz (<https://stackoverflow.com/questions/76237096/how-to-get-the-index-in-the-original-array-when-using-np-argwhere-on-a-slice>)

- Escribir dos condiciones en un if. (https://www.w3schools.com/python/python_conditions.asp)

La función para graficar la imagen es de invención propia.

La función para guardar el conjunto de imágenes a partir de los movimientos obtenidos por los algoritmos se obtuvo mediante chat GPT. Al necesitarse solamente la acción de guardar las imágenes en un zip, se decidió por esa forma de obtener información.

- El método de diagonal cero es de invención propia.

La función encontrar_indices para usar en los modelos (<https://stackoverflow.com/questions/73292182/finding-indices-of-array-where-values-meet-condition-python-numpy>)

Para BFS y DFS se usó como referencia adicional: Herráez Jiménez, R. (2015). Estudio de técnicas de resolución para el juego del Sokoban.

El código de los algoritmos base se obtuvo de juntar diversas fuentes:

- <https://github.com/maheshreddykukunooru/Sokoban-Game/blob/master/bfs.py> (se obtiene la forma de generar la lista y de definir si un estado fue visitado).

- <https://github.com/xbandrade/sokoban-solver-generator/blob/main/src/astar.py> (permite generar la forma de la lista para greedy y A* y la estructura de evitar duplicados)

El contador de estados expandidos y la evaluación de dominancia es de invención propia.

Para graficar los histogramas se empleó chat GPT.