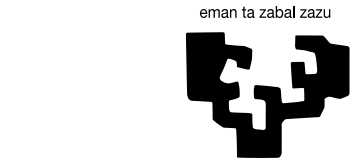

Inteligencia Artificial

Silvia Arenales Muñoz y Judith Antelo Lacalle

February 15, 2023

Abstract

Pacman. Algoritmos de búsqueda: Depth First Search, Breadth First Search y Uniform Cost Search.

1 Introducción

En este proyecto de la asignatura de Inteligencia Artificial, se nos ha pedido desarrollar el juego *PacMan*, en el cual nuestro agente (*PacMan*) deberá ser capaz de comer todas las bolitas mientras esquiva los fantasmas en un escenario.

Concretamente, en esta primera fase, en ***PL1a***, se deben implementar tres algoritmos de búsqueda, que permitan buscar un camino a través del laberinto para poder comer un punto de comida situado en el escenario.

Los algoritmos desarrollados han sido los siguientes:

1. Búsqueda en profundidad - Depth First Search
2. Búsqueda en anchura - Breadth First Search
3. Búsqueda en grado de coste uniforme - Uniform Cost Search

2 Pregunta 1: Buscando un punto de comida fijo usando Depth First Search.

Búsqueda en profundidad o Depth-First Search (DFS) consiste en ir visitando los estados de un camino en concreto hasta llegar al estado final. Si no encuentra el estado final, comienza a expandir de nuevo una nueva rama, así hasta visitar el nodo final buscado. Una vez se haya encontrado dicho nodo, se devolverá el camino que se debe recorrer desde el estado inicial hasta el final mediante un array de movimientos que tiene que hacer el agente (*PacMan* para llegar a dicho estado.

En términos de implementación del algoritmo, hemos utilizado una pila (**Stack**) como estructura de datos, para almacenar junto a su camino.
La pila nos permite apilar(**push**) y desapilar(**pop**) aquel estado que está en la parte superior de la pila. Así, la pila nos ayuda a recordar aquellos estados que hemos visto que son adyacentes pero que todavía no hemos visitado. Se hace esto por si se va a tener que tener que ir por otro camino.

Al final, una pila es una lista ordenada o estructura de datos que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo LIFO (Last In, First Out) lo que viene a ser lo mismo, último en entrar,

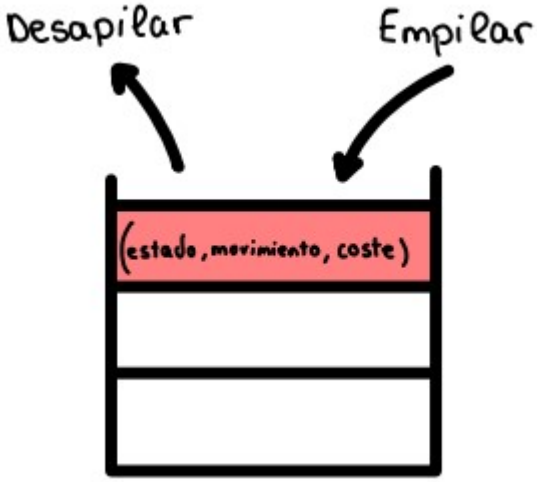

Figure 1: STACK

primero en salir.

3 Pregunta 2: Breadth First Search.

Este algoritmo, algoritmo de búsqueda en anchura (BFS), permite recorrer los estados del grafo primero todos los estados sucesivos del estado de inicio. Es decir, primero comienza por el estado inicial y visita todos sus adyacentes a la vez, y posteriormente recorre todos los adyacentes de los adyacentes y así sucesivamente hasta ellegar al estado final. Por tanto, nuestro algoritmo nos devolverá el camino desde el estado inicial hasta el estado final.

Teniendo en cuenta la implementación, hemos hecho uso de una cola (**Queue**), que funciona de manera similar a una pila pero con la diferencia de que al insertar un elemento se coloca al final, al sacar un elemento se sacará el primero de la cola. De esta manera, podemos recorrer primero todos los adyacentes de un estado, sin la necesidad de tener que profundizar en toda una rama como se hacía con la pila.

Al final, una cola es un tipo de dato abstracto, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pull por el otro. También se le llama estructura FIFO y el acceso es (Last In, First Out) lo que viene a ser lo mismo, primer elemeto en entrar será también el primero en salir.

Figure 2: QUEUE

4 Pregunta 3: Cambiando la función de coste.

Este último algoritmo nos permite hallar un camino en el grafo desde el estado inicial hasta el estado final con el mínimo coste posible. El fincionamiento es similar al del algoritmo de búsqueda, pero aquí se ha utilizado una cola de prioridad (**priority queue**). Creemos que ha sido acertada ya que a la hora de desencolar un estado de la cola, saldrá aquel con el valor de coste menor, puesto que los elementos se ordenan en base al coste de movimientos. Por tanto, de esta manera nos aseguramos que el camino que lleva del estado inicial al final será el que menor coste posible tenga.

5 Aspectos Reseñables

En cuanto a la hora de implementación de los tres métodos de búsqueda, hemos seguido el patrón y la estructura proporcionada en las diapositivas de algoritmos de búsqueda exhaustiva (diapositiva 32). Una de las cosas que nos ha costado un poco más ha sido entender bien el mecanismo de la función getSuccessors() ya que nos devolvuelve tres cosas que son, cual es el nodo sucesor, el movimiento del sucesor (coordenadas) y el coste del sucesor. Una vez que entendimos bien la función y las tres cosas que devolvía, nos resultó más fácil acceder a esos datos y poder devolver el resultado, sumándole el coste nuevo del nodo adyacente, al anterior y así sucesivamente.