Tecnológico de Monterrey

ITESM CSF
Sistemas inteligentes 7-10 lu
Dr. Víctor de la Cueva
Septiembre 20 2018
Adrián Biller A01018940

Documentación Proyecto 2

Descripción técnica

Este programa fue desarrollado en Python 3.6 con estructura de datos de árboles de decisión. Este árbol de decisiones está formado por clases de nodos llamados *Node* los cuales contienen los siguientes atributos:

- boardState: estado actual del tablero en ese nodo
- moveDone: movimiento realizado por ese nodo en el tablero.
- isVisited: estado del nodo, si fue visitado o no.
- father: nodo padre
- childNodes: arreglo de todos los nodos hijos generados por el nodo padre
- heuristic: contiene el valor de heurística del tablero del nodo.

y también contiene las siguientes funciones:

- printBoardState: imprime el tablero del nodo
- getBlankPosition: obtiene la posición en donde se encuentra el cero del tablero.
- checkPossibleMoves: obtiene un arreglo de todos los posibles movimientos que se puedan realizar con ese tablero.
- createChildren: con base en los movimientos obtenidos por la función pasada se crean nodos hijos con dichos movimientos.
- getNewBoard: se obtiene el tablero después de aplicarle un movimiento.
- returnSolutionMove: función recursiva que obtiene los movimientos de cada hijo hasta llegar al padre para obtener el arreglo de movimientos para llegar a la solución.
- getValuePosition: función que obtiene las coordenadas del valor que se envie dentro de la matriz.
- checkH1: función que por cada elemento fuera de su lugar adiciona un punto en la heurística.
- checkH2: función que obtiene las distancias Manhattan de cada elemento suma todas para asignar la heurística.

además, se encuentran las funciones **astarH1** y **astarH2** que implementan el algoritmo y generan la respuesta para la función **busquedaAstar** con diferentes heurísticas.

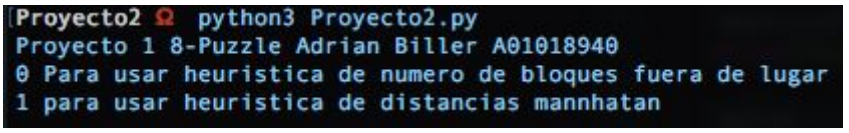
Todo el código está documentado en caso de querer verlo.

Manual de usuario

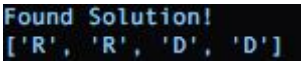
Éste código fue realizado en python 3.6 y utiliza dos paquetes: copy y random.
El código ya cuenta con main para realizar pruebas, sin embargo, para propósitos de pruebas unitarias se creó la función **busquedaAstar** cuyos parámetros son (**estado inicial del tablero, estado final del tablero, 0 para uso de la primera heurística ó 1 para uso de la segunda heurística**).

En dado caso se se quieran realizar pruebas en terminal se deberá correr el siguiente comando:

cuando se corra ese comando se mostrará el siguiente texto en la terminal:



Al aparecer este texto se debe ingresar ya sea 0 para la ejecución de A* con la primera heurística o 1 para la segunda.
Una vez ingresado se ejecutará el algoritmo y mostrará el resultado:



Conclusiones

En este proyecto se pudo observar la diferencia que se tiene entre algoritmos de búsqueda no informada como los realizados en el proyecto anterior a la misma clase de algoritmos pero ahora teniendo un criterio para diferenciar los nodos que tienen una mejor solución o están más cerca de la solución. La diferencia en cuanto a rapidez es considerable, a diferencia del proyecto anterior, en este el número de niveles revisados es menor al mejor resultado obtenido en la búsqueda no informada. En conclusión, si es posible poder crear un criterio para diferenciar cuando hay estados con mejores resultados entonces se puede optimizar la obtención del resultado y reducir los tiempos de búsqueda considerablemente.