Tecnológico de Monterrey

ITESM CSF
Sistemas inteligentes 7-10 lu
Dr. Víctor de la Cueva
Agosto 30 2018
Adrián Biller A01018940

Documentación Proyecto 1

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

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: funcion recursiva que obtiene los movimientos de cada hijo hasta llegar al padre para obtener el arreglo de movimientos para llegar a la solución.

además, se encuentran las funciones **breadthFirstSearch** y **depthFirstSearch** que implementan el algoritmo y generan la respuesta para la función **busquedaNoInformada**

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 **busquedaNoInformada** cuyos parámetros son (**estado inicial del tablero, estado final del tablero, 0 para uso de breadth first search ó 1 para uso de depth first search**).

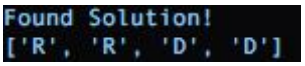
En dado caso se se quieran realizar pruebas en terminal se deberá correr el siguiente comando:

```
Ω python3 Proyecto1.py
```

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

```
Proyecto 1 8-Puzzle Adrian Biller A01018940
0 Para Breadth first search / 1 para Deph first search
```

Al aparecer este texto se debe ingresar ya sea 0 para la ejecución de breadth first search o 1 para depth first search
Una vez ingresado se ejecutará el algoritmo y mostrará el resultado:



Conclusiones

En este proyecto se puede apreciar el diferente performance que tienen diferentes algoritmos como lo son breadth first search y depth first search. En este programa en específico el uso de BFS es mucho más eficiente que DFS ya que al ser un problema donde se generan nodos infinitamente ya que hay un número infinito de movimientos el usar DFS genera una rama infinita, sin embargo con BFS se va revisando nivel por nivel y valga la redundancia, tiene una búsqueda más amplia.
En este problema en específico, el uso de BFS toma 0.008 segundos (depende de la computadora en la que se esté ejecutando) mientras que DFS puede tomar más tiempo, inclusive al punto de no llegar a una solución.
En conclusión, un algoritmo no es bueno o malo, si no que puede variar el problema ciertos algoritmos son más eficientes que otros para encontrar soluciones. En este caso, el BFS es más eficiente para este problema.