INTELIGENCIA ARTIFICIAL - BUSQUEDAS

PROYECTO

Manuel Benítez Sánchez

TABLA DE CONTENIDO

Contenido

Busquedas

FORMULACIÓN DEL PROBLEMA

Nombre: Movimiento restringido del caballo de ajedrez

Estado: Cada estado estará representado con posiciones en un matriz 5x5, los estados serán todas las combinaciones posibles entre filas y columnas exceptuando el estado (0,4).

Estado inicial: Se indicará de donde partimos y hacia donde nos dirigimos, Es decir, de donde parte el caballo y hasta que posición tienes que ir.

Acciones: El caballo solo podrá hacer los movimientos correspondientes a la figura del caballo en el ajedrez, hasta 8 posibles movimientos, siguiendo el formato L.

- 2 arriba + 1 derecha
- 2 arriba + 1 izquierda
- 2 derecha + 1 arriba
- 2 derecha + 1 abajo
- 2 abajo + 1 derecha
- 2 abajo + 1 izquierda
- 2 izquierda + 1 arriba
- 2 izquierda + 1 abajo

Modelo de transición: En cada movimiento se actualiza la posición de nuestro caballo en el tablero

Función objetivo: El caballo debe llegar a una posición en concreto teniendo en cuenta sus movimientos posibles, y las restricciones del tablero

Coste: Se supone que es 1

Observaciones: Para la implementación hemos pensado en no usar una matriz, pues solo necesitamos tener la posición objetivo y la posición del caballo que se irá actualizando en cada movimiento

SOLUCIÓN AL PROBLEMA

La solución consta de 6 archivos *.java

- ModificatedBoard.java
- HorseFunctionFactory.java
- HorseGoalTest.java
- HorseHeuristica1.java
- HorseHeuristica2.java
- HorseDemo.java

NÚMERO DE NODOS EXPANDIDOS Y COSTE DE LA SOLUCIÓN

Por cada tipo de búsqueda realizada nos devuelve a la información de las acciones realizadas (movimientos), el coste de esta búsqueda (pathCode), nodos expandidos y la cola de memoria usada.

BÚSQUEDA EN ANCHURA

ACTION[NAME==1D2AR]
ACTION[NAME==2I1AB]
ACTION[NAME==1D2AR]
PATHCOST : 3.0
NODESEXPANDED : 8
QUEUESIZE : 9
MAXQUEUESIZE : 10

BÚSQUEDA EN PROFUNDIDAD

ACTION[NAME==2I1AR]
ACTION[NAME==2D1AR]
ACTION[NAME==2I1AR]
ACTION[NAME==1D2AB]
ACTION[NAME==1D2AR]
PATHCOST : 5.0
NODESEXPANDED : 6
QUEUESIZE : 9
MAXQUEUESIZE : 11

FUNCIÓN HEURÍSTICA 1

ACTION[NAME==1D2AR]
ACTION[NAME==2I1AB]
ACTION[NAME==1D2AR]
PATHCOST : 3.0
NODESEXPANDED : 11
QUEUESIZE : 12
MAXQUEUESIZE : 13

FUNCIÓN HEURÍSTICA 2

ACTION[NAME==1I2AR]
ACTION[NAME==2D1AB]
ACTION[NAME==1I2AR]
PATHCOST : 3.0
NODESEXPANDED : 8
QUEUESIZE : 11
MAXQUEUESIZE : 12

CONCLUSIÓN

Analizando los resultados obtenidos de la solución vemos que con la implementación A* es igual que si usamos la heurística 2, si no tenemos en cuenta el tamaño en memoria, vemos que la heurística 2 mejor la heurística 1 en cuanto a nodos expandidos.

La implementación de la búsqueda en profundidad es la que mayor costo tiene.