Homework 1 - Artificial Intelligence

Gianluca Rea
Matricola: 278722

Introduction

In this homework, agents were implemented to solve the puzzle problem, along with tree search algorithms. In the notebook, there is an example of the 8 & 15 puzzles. Agents also have been implemented to play the game of chess in another notebook.

Puzzle

Two agents with two different heuristics (informed search) and two node search algorithms (uninformed search) have been implemented for the puzzle. That is the sum of the Manhattan distance of each piece from the current position to the correct position and the misplaced tiles which count all the pieces that are in the wrong position.
In addition to these two heuristics, the Best First Search exploration algorithm and the Depth First Search algorithm have been implemented.
The notebook allows you to create a board of square size and takes care of seeing even if the board created randomly is solvent or not. Precisely for this reason, depending on whether N is even or odd, the function in question will have a certain behaviour.
A puzzle is solvent if:

- N is odd and the number of inversions is even on the size of the input;
- N is even and:
    - The blank is in an even position counting from the bottom while the number of reversals is odd;
    - Or the blank is in an odd position counting from the bottom while the number of inversions is even;

(The number of inversions in an array p is represented by the number of values such that: p [j]> p [j + 1] for each position j of the array p)

Optimization

The possible moves that an agent / bfs / dfs can execute are at most m = n-1, i.e. n possible moves minus one move representing the move previously made (by the relative of the current node that generated the m states),
this optimization was carried out within the code to save memory and execution time, in fact without this optimization the state that is removed would generate useless loops.

Broken Loop

In the case of the two heuristics, the greedy algorithm does not generate loops, this is because, whenever the evaluation of the new states is carried out, all the states with lower heuristics than the others (but equal to each other) are taken, after which between them is took a random one.
This move allows for avoiding loops that could be created with the greedy algorithm, thus leading to an infinite non-solution of the board. Certainly, loops can be created, but always taking a random index among the smaller ones (but equal to each other) the loop will always be broken.
As far as the dfs and bfs research is concerned, the nodes are simply visited in order of algorithm until a state equal to that of the solution is reached.

Final analysis

The execution of the algorithm on the 8-puzzle showed that it successfully ends with both heuristics in a reasonable time.
Manhattan distance certainly shows better results than misplaced tiles.
As far as bfs and dfs are concerned, we cannot speak of excellent algorithms.
As the bfs finds a solution after a longer time than the two heuristics. While the dfs can hardly find a solution, the dfs in particular could only find a solution if the board is readily solvent.
All the behaviours of the two search algorithms together with the two heuristics are amplified in the 15-puzzle.
Precisely in the 15-puzzle what finds a solution in finite time is the Manhattan distance heuristic. The bfs finds a solution after a long time. DFS and Misplaced tiles, on the other hand, are unlikely to find a solution in a reasonable time.

Chess

The “Chess” library was used to create the game of chess.
This library allows easy development of the game of chess, as it is possible to create a board, to have the board status and finally it allows to see all the available moves and to execute one depending on who is the turn.
In this regard, three different heuristics have been implemented for this game:

- Material: the sum of the pieces by their value, minus the sum of the opponent's pieces for the costs (n = 0, p = 1, b = 3, k = 3, r = 5, q = 9);
- Greater number of pieces: the number of pieces of the player who has the turn minus the number of pieces of the opponent;
- Most move: ie the move that allows you to have more moves available in the next turn.

N.b:
When a move allows you to checkmate or checkmate the king, that move is considered concerning the heuristics.

Broken Loop

The heuristics can calculate equal global minimum costs, to avoid the creation of cycles for this fact, a random value representing the move is taken between the global minima. Thanks to this it is not possible to have cycles. An example in question is at the beginning of the game, in that case, the heuristics always calculate equal values, and in this case, a random move will always be performed.

Final analysis

Three different tests were sent to execute in which two agents challenge each other for each heuristic and another in which an agent uses the Material heuristic and another agent uses the heuristic greater number of pieces.
From these tests it is not possible to establish which heuristic is better than the other, in fact, the games usually end in a draw for the most part and a few times with the victory of one of the two.