Programming Project 1

Solving the 8-puzzle using A* algorithm

Problem Formulation: Given a 3x3 board with 8 tiles and an empty space. The objective is to place the numbers on tiles to match final configuration using the empty space. We can slide four adjacent (left, right, above and below) tiles into the empty space.

States: List of 9 locations

Configurations of the puzzle (9! configurations/possible states)

Not all states are reachable from a given state. In fact, exactly half of them are reachable from a given state.

Actions: Move one of the movable pieces up, down, left, right (<=4)

Performance measure: Minimize total moves

Goal Test: Check if goal configuration is reached

Path Cost: Number of actions to reach goal

Operators:

24 operators of the form $Op_{(r,\ c,\ d)}$ where (r, c) belongs to {0, 1, 2}, d belongs to {Left, Right, Up, Down}

Program Structure:

The program contains three main classes

a) **A_Star_Algorithm**
b) **Tile**
c) **BoardGame**

A_Star_Algorithm: This class contains the main function. It accepts the initial state and goal state as input from the user. It prompts the user to enter the type of heuristic he/she wants to apply. The program will get redirected to the BoardGame class through board.search(inputTiles,goalTiles,heuristic);

BoardGame: This class contains the main implementation of the heuristics. Based on the user input, the code gets redirected to manhattanHeuristic(input123,goal) or misplacedHeuristic(input123,goal).

In the misplaced tiles heuristic, we compare the x-position and y-position of each of the generated tiles with the goal state tiles and compute the number of misplaced tiles present. findTile() return the tile if there is a matching tile.

In the manhattan distance heuristic, we will take the difference between the x positions and y positions of each tile, of initial and final state and add them.

Tile: Tile class stores the x position, y position and the value of each cell of the 8-puzzle.
The class has getters and setters for x position, y position and value.

Variables and Methods:

A_Star_Algorithm:

InputTiles: It contains the information of the tiles of the initial state. This information includes x-position, y-position and the value of the tile.

goalTiles: It contains the information of the tiles of the goal state. This information includes x-position, y-position and the value of the tile.

add(inputTiles): This method adds inputTiles to the generated boards list.

board.search(inputtiles,goalTiles): Calls the main A star search.

Tile:

clone( ): This method is used to eliminate shallow copying and implement deep copying.

BoardGame:

exploredNodes: This array list contains all the nodes that are generated.

generatedBoardsList: This array list contains all the nodes that are opened.

generatedBoards: This hash map contains the fringe.

gValue: This contains the g(n) value.

neighbourTiles: Stores the neighbouring tiles for the given tile.

neighbourTiles1( ): This method finds the neighbouring tiles for the given tile by comparing the x and y positions of the tiles in it's vicinity and return them.

findTile( ): This method returns the Tile (it's x position, y position and value) by taking the number and input array list.

findTilewithXY( ): Returns the tile by taking x,y positions as input.

sortbyKey(): This method uses a Treemap. Using this method, we can get the least heuristic board from the list.

manhattanHeuristic(): Compute the manhattan heuristic by taking generated tiles and goal state.

swapTiles(): This method is used to swap the given tile with it's neighbouring tiles. It checks the x and y positions of the input tile with each tile of the generatedBoardsList. If the input tile is not present in the fringe, this method will add it to the fringe.

printBoard( ): Print the states(board).

Submitted by

Manideep Reddy Nukala

Sai Charan Reddy Vallapureddy