AI Assignment

Team Name: Matrix Agents

Team Members: Shubham Singh (18MCMC52)
Vikas (18MCMC58)

Game: 8 Puzzle

Problem Statement:

The 8-puzzle is a sliding tile puzzle that is made up of a square structured frame area containing tiles in random/irregular order with one tile missing. It is a smaller version of the 15-puzzle (also called Gem Puzzle, Boss Puzzle, Game of Fifteen, Mystic Square and numerous other names) . 8-puzzle is basically a frame area separated into 3x3 grids containing 8 tiles and one void grid. The tiles are marked in some way so as they can be identified. The tiles are mostly numbered from 1 to 8. We are given with an initial configuration of the tiles. A desired final configuration is also given. We have to reach the final state by sliding the tiles using the empty grid present.

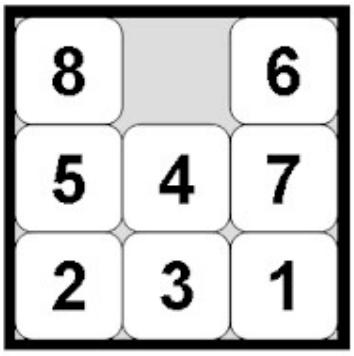
Language: JAVA

Algorithm: A* Algorithm

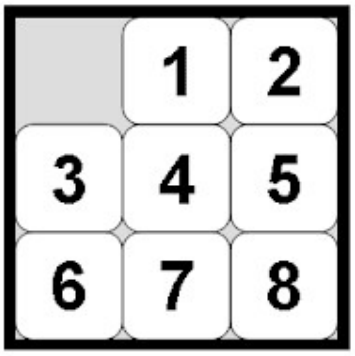
Work done: Front End (GUI) –Shubham Singh
Back End –Vikas

Rules:

1. Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one 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.



Initial state



Goal State

Environment set up:

- Install Eclipse IDE (Oxygen)
- Here we have used java JDK 8.1
- The game can be run on any machine that has java installed by using a java compiler.
- Put the source code in Eclipse workspace and execute it.

Algorithm:

A star(A*) Algorithm:

It is a state space search algorithm. It is an algorithm that is regularly used searching of path and traversing of a graph, to plot an optimal path traversable between states (nodes). As it is efficient and accurate, it revels in boundless utilization. The A* algorithm integrates characteristics of uniform-cost search and heuristic based search to proficiently find optimally efficient path. A*algorithm is a

best-first search algorithm in which the cost linked to a state is f(n) = g(n) + h(n),where

- g(n) is the cost of the path traversed from the initial state to node n.
- h(n) is the estimated path-cost or the heuristic function cost from node to the goal node.
- Thus, f(n) shows the lowest total cost possible for any path leading through node n to goal state when h(n) is the estimated remaining path-cost.
- If f value of various nodes of equal, such a stalemate situation is settled by taking the node with inferior heuristic estimate h(n) values(node ordering).
- The procedure continues till the node to be expanded is a goal node.

Efficiency of A*:

A* is the fastest search algorithm. There is no algorithm that can find a solution expanding lesser number of nodes than a* for a given heuristic.

Heuristics:

Number of Misplaced Tiles:

In this heuristic, a tile from any position may be taken out and moved to any required position. The evident algorithm for finding a solution is basically moving each tile from its present spot to the spot in its goal configuration. Thus, the path length of the least cost-path is the count of tiles that are not present in its desired positions.

1		3
4	2	5
7	8	6

1	2	3
4	5	6
7	8	

Number of misplaced tiles

- Here tiles 2,5,6 are not in correct positions . so h(n) =3

Sum of Manhattan Distances:

Manhattan Distance-: The optimal solution to this puzzle is found by moving each tile along a *shortest path* between its initial and goal state. For anyone tile, the length of this shortest path is the grid distance (horizontal plus vertical distance) between its current and goal positions. Therefore, the total solution length is merely the summation of these grid distances for each tile.

Example:

In the figure given below, only the "3"," 8" and "1" numbered tiles are, away from their goal state by 2, 3, and 3 squares respectively. So the heuristic function evaluates to 8(2 + 3 + 3). It means the heuristic signals that the goal state can be reached in just 8 moves.

3	2	8
4	5	6
7	1	

Initial state

1	2	3
4	5	6
7	8	

Goal state

Implementation

1. **The A* Algorithm have been Implemented.**
2. Heuristics used are number of misplaced (out of order) tiles and sum of Manhattan distance.
3. The flaws of the heuristics are discussed and an improvement in Manhattan heuristic is done below.
4. All the implementations are done in JAVA programming language.

Development process:

1. As this is the AI based project we know that we have to spend lots of time on this project as we did it.
2. First few days we understand how it work which algorithm is best for it we did lots of researches.
3. Understand the working of algorithm
4. While implement in coding environment we face lots of challenge to pick which data structure we should pick.
5. We implement stack data structure and Queue data structure.

GUI:

1. For GUI board we have using java library javax.swing.*; and java.awt.*;
2. We have create initial state board in which we enter the input value of our choice from 0 to 8 and after that we have to press ok for getting the optimal way to solve the given problem.
3. After getting the input from bored first we have to check is there any repeat value or more than 8 in case their is unsolvable message will be Occured.
4. We have create simple GUI board.

Note:- As we both are work on different module and we are living in remote area we faced lots of communication barrier we did not integrated the module as we think but we will complete it soon. We are sending our complete project and also our GUI part that is incomplete.