Team Members: 1. Amit Patil (801077053)

2. Raj Jani (801077311)

**Solving 8 Puzzle problem with A\* Algorithm**

**8 Puzzle Problem Formulation:** A board with 3x3 boxes is given where 8 boxes are numbered with random integers (1-8) and 1 empty box numbered ‘0’. The boxes can be moved in four directions (Up, Down, Left and Right) and the aim is to move the boxes in a way that the mentioned goal state is reached.

Possible States: List of locations of 9 boxes which means 9! Configurations/possible states

Actions: Boxes can move up, down, left and right

Performance measure: Minimum number of moves made to reach goal state.

Goal Test: Check if boxes reached the goal configuration.

Path Cost: Number of moves performed to reach goal state

**Program Structure:** The classes and functions used in the program are:

Main: This is the main class of the project. It is where the program execution begins.

Node: This class creates and holds information of the state of a node using \_\_init\_\_() function.

eightPuzzle: This class has the main heuristics implemented and it contains the following functions defined:

* Init: Defines the problem state (input, goal, heuristic chosen)
* isGoalState: Checks if the current state matches the goal state.
* calculateHvalueMisplacedTiles: This method calculates the Misplaced Heuristic of nodes using their index value
* calculateHvalueManhattan: This method calculates the Manhattan Heuristic of nodes using their index value
* generatechilds: Generates the valid options of moves possible for a particular state (up, down, left, right)
* checkifSameStateExistswithLowerFvalue: This function returns true or false based on whether there exists a state with lower/higher value of f. Thus, checking previous states that are either in fringe or in the expanded list.
* createNode: Function to generate new node
* calG: calculates the G value
* calculateH: calculates the H heuristic value
* isTwoStatesEqual: checks if two states are equal
* findIndex: returns the value of index of the number in the current state
* printstate: print the current state
* printNode: prints the values of g, h, f and state
* printPath: print the path from input to goal.

Astar: This is the class where A\* algorithm is implemented and from here calls are made to functions of eightPuzzle

* AstarSearch: Is the main class for A\* Algorithm where it searches for the best path to follow and defines the expanded Queue which counts the number of nodes expanded and reached.

**ALGORITHM APPROACH:**

* We first convert input and goal states provide by user into a list that contains 3 lists
* Next, we check if the provided input and goal states are proper or not.
* Then we select the heuristic for solving the puzzle
* Now, Apply the A\* algorithm

1. Check if initial input is equal to goal state else,
2. While (fringe not empty)
3. Keep generating the children
4. And for each child now we check

If the child is the goal state, check if the same state exists in the fringe with lower f value, check if it exists in the expanded list

1. Add the parent to expanded list
2. Finally print the nodes to reach goal state.

**Sample Code Run:**

**CASE 1:**

C:\Users\RAJ\Desktop\IS project>python main.py

**Enter input:**

1 2 3

7 4 5

6 8 0

**Enter goal:**

1 2 3

8 6 4

7 5 0

**Select Heuristic Function**

**1.Manhattan Distance**

**2.Misplaced Tiles**

**1**

Goal Found [[1, 2, 3], [8, 6, 4], [7, 5, 0]]

The Path length = 9

Path Trace

1 2 3

7 4 5

6 8

1 2 3

7 4

6 8 5

1 2 3

7 4

6 8 5

1 2 3

7 8 4

6 5

1 2 3

7 8 4

6 5

1 2 3

8 4

7 6 5

1 2 3

8 4

7 6 5

1 2 3

8 6 4

7 5

1 2 3

8 6 4

7 5

**The number of nodes generated = 63**

**The number of nodes expanded = 21**

**CASE 2:**

C:\Users\RAJ\Desktop\IS project>python main.py

**Enter input:**

1 2 3

4 0 5

6 7 8

**Enter goal:**

1 2 3

4 5 6

7 8 0

**Select Heuristic Function**

**1.Manhattan Distance**

**2.Misplaced Tiles**

**1**

Goal Found [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

The Path length = 15

Path Trace

1 2 3

4 5

6 7 8

1 2 3

4 5

6 7 8

1 2 3

4 5 8

6 7

1 2 3

4 5 8

6 7

1 2 3

4 5 8

6 7

1 2 3

5 8

4 6 7

1 2 3

5 8

4 6 7

1 2 3

5 6 8

4 7

1 2 3

5 6 8

4 7

1 2 3

5 6

4 7 8

1 2 3

5 6

4 7 8

1 2 3

5 6

4 7 8

1 2 3

4 5 6

7 8

1 2 3

4 5 6

7 8

1 2 3

4 5 6

7 8

**The number of nodes generated = 1600**

**The number of nodes expanded = 583**

**CASE 3:**

C:\Users\RAJ\Desktop\IS project>python main.py

**Enter input:**

1 2 3

0 4 5

6 7 8

**Enter goal:**

1 2 3

4 5 0

6 7 8

**Select Heuristic Function**

**1.Manhattan Distance**

**2.Misplaced Tiles**

**2**

Goal Found [[1, 2, 3], [4, 5, 0], [6, 7, 8]]

The Path length = 3

Path Trace

1 2 3

4 5

6 7 8

1 2 3

4 5

6 7 8

1 2 3

4 5

6 7 8

**The number of nodes generated = 8**

**The number of nodes expanded = 2**

**CASE 4:**

C:\Users\RAJ\Desktop\IS project>python main.py

**Enter input:**

1 2 3

0 4 5

6 7 8

**Enter goal:**

1 2 3

4 5 6

7 8 0

**Select Heuristic Function**

**1.Manhattan Distance**

**2.Misplaced Tiles**

**2**

Goal Found [[1, 2, 3], [4, 5, 6], [7, 8, 0]]

The Path length = 16

Path Trace

1 2 3

4 5

6 7 8

1 2 3

4 5

6 7 8

1 2 3

4 5

6 7 8

1 2 3

4 5 8

6 7

1 2 3

4 5 8

6 7

1 2 3

4 5 8

6 7

1 2 3

5 8

4 6 7

1 2 3

5 8

4 6 7

1 2 3

5 6 8

4 7

1 2 3

5 6 8

4 7

1 2 3

5 6

4 7 8

1 2 3

5 6

4 7 8

1 2 3

5 6

4 7 8

1 2 3

4 5 6

7 8

1 2 3

4 5 6

7 8

1 2 3

4 5 6

7 8

**The number of nodes generated = 1968**

**The number of nodes expanded = 708**