INTELLIGENT SYSTEMS

Report

Programming Project 1

Solving the 8-puzzle problem using A* search algorithm

Group-13:

Geethesh Byreddy gbyreddy@uncc.edu

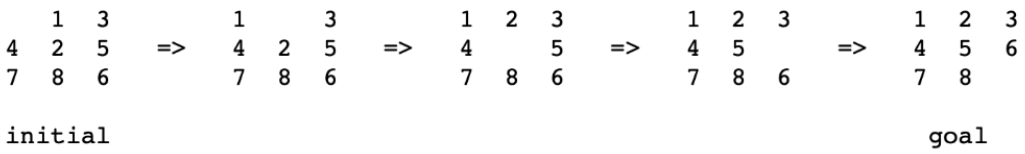
Jyoti Swaroop Naidu Yelaka jyelaka@uncc.edu

Srinivasulu Padigay spadigay@uncc.edu

Index

1. Problem Statement:

The 8-puzzle problem is a puzzle invented by *Noyes Palmer Chapman*. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. The goal is to rearrange the blocks so that they are in order. We are permitted to slide blocks horizontally or vertically into the blank square.The following shows a sequence of legal moves from an initial board position to the goal position.



2. Project:

You are to implement A* search algorithm and apply it to the 8-puzzle problem, using a programming language of your choice. In addition to coding of A* search algorithm, provide problem formulation, operators, g-cost and two heuristic functions (h-cost) of the 8-puzzle problem. Your program should accept initial and goal states from the user and will compute the best path.

3. Language And IDE:

Programming Language: Python.
IDE:PyCharm.

4. Project Details:

The Project folder consists of 5 files namely

1) main.py: This is the main functional file of the project. This further converges into different files to perform different heuristics.
2) misplacedTiles.py: This file helps in implementing the Misplaced tiles heuristic.
3) manhattanDistance.py: This file helps in implementing the Manhattan distance heuristic.
4) Readme.txt: This helps with the contents, their specifications and with execution of the program

5) Report.pdf: This report offers a comprehensive account of the project, furnishing a detailed overview of the procedures employed and the resulting outcomes.

5. Functions and Procedures:

main.py:

check_for_err():

Arguments:list_input

Functionality: This function mainly looks after the input. It checks whether the input entered is in the correct format or not. It has an argument 'list_input' which the function works on, it represents the state of the problem, it can be the goal or initial state.

start():

Arguments: none

Functionality: This function has the functionality of the main. It takes all the input from the user, directs into the specified algorithm and prints the required output.

main():

Arguments: none

Functionality: This helps in starting the program.

misplacedTiles.py:

Class: Node

__init__():

Arguments: self, state, goal_state, parent=None, g_cost=0

Functionality: The main functionality of this function is to initialize a node object for the problem. The state argument states the current state of the list, goal state is the required output state, parent states the parent node in the search tree(default none) and g_cost states the cost of the current node from the parent node(default 0).

calculate_h_cost():

Arguments: self, goal_state

Function: This function mainly focuses on calculating the h cost i.e. calculating the number of misplaced tiles. The argument goal_state is directed to the goal_state given by the user.

__lt__():

Arguments: self, other

Functionality: This helps in comparing the f-cost of the two states.

a_search():

Arguments: case_input, case_goal

Functionality: This is overall the key function of the misplaced tiles, this implements a priority queue, implements the a* search algorithm, returns the nodes and generates them according to the algorithm.

process():

Arguments: case_input, case_goal

Functionality: This function mainly calls the a_search() and prints the returned output accordingly.

manhattan_distance.py:

Class Node:

__init__:

Arguments:self, state, parent, cost, heuristic

Functionality: This function mainly works on initializing all the states,costs and nodes of the tree.

__lt__:

Arguments: self, other

Functionality: It checks the costs of the present state and the previous states.

manhattan_distance:

Arguments:state, goal

Functionality: This function calculates the manhattan distance.

get_neighbors:

Arguments: node

Functionality: This function mainly gets to know about the functionality of the neighboring nodes.

a_star_search:

Arguments: initial_state, goal

Functionality: This is overall the key function of the misplaced tiles, this implements a priority queue, implements the a* search algorithm, returns the nodes and generates them according to the algorithm.

process:

Arguments: case_input, case_goal

Functionality: This function implements the a* algorithm and gives the result accordingly.

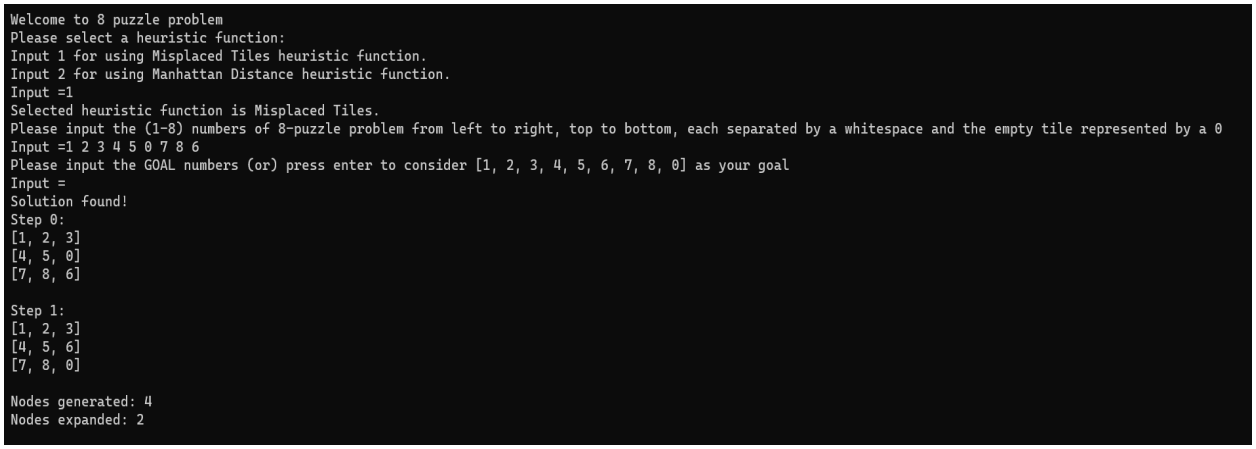
6. Input and Cases:

These cases compare the Manhattan Distance and Misplaced Tiles heuristics on the 8 puzzle programs.
Here are 6 different cases

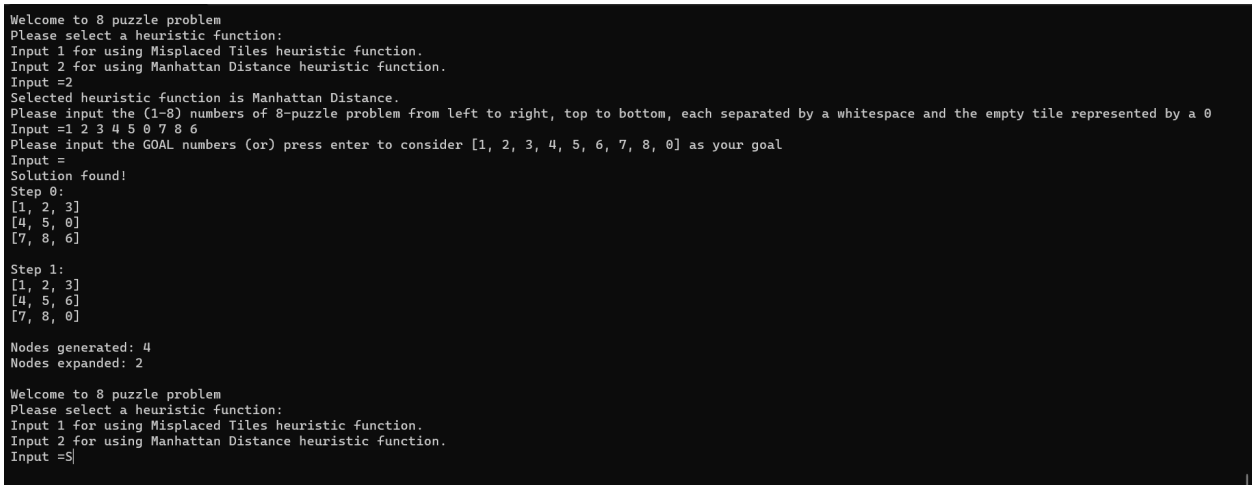
Case-1:

Input State: 1 2 3 4 5 0 7 8 6
Goal State : 1 2 3 4 5 6 7 8 0

Misplaced Tiles:



Manhattan Distance:



Case-2:

Input state:1 4 2 3 5 6 7 8 0

Output state:1 2 3 4 5 6 7 8 0

Misplaced Tiles:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input = 1
Selected heuristic function is Misplaced Tiles.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input =1 4 2 3 5 6 7 8 0
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =
Solution found!
Step 0:
[1, 4, 2]
[3, 5, 6]
[7, 8, 0]

Step 1:
[1, 4, 2]
[3, 5, 6]
[7, 0, 8]

Step 2:
[1, 4, 2]
[3, 0, 6]
[7, 5, 8]

Step 3:
[1, 4, 2]
[0, 3, 6]
[7, 5, 8]

Step 4:
[1, 4, 2]
[7, 3, 6]
[0, 5, 8]

Step 5:
[1, 4, 2]
[7, 3, 6]
[5, 0, 8]
```

```
Step 6:
[1, 4, 2]
[7, 3, 6]
[5, 8, 0]

Step 7:
[1, 4, 2]
[7, 3, 0]
[5, 8, 6]

Step 8:
[1, 4, 2]
[7, 0, 3]
[5, 8, 6]

Step 9:
[1, 0, 2]
[7, 4, 3]
[5, 8, 6]

Step 10:
[1, 2, 0]
[7, 4, 3]
[5, 8, 6]

Step 11:
[1, 2, 3]
[7, 4, 0]
[5, 8, 6]

Step 12:
[1, 2, 3]
[7, 4, 6]
[5, 8, 0]
```

```
Step 13:
[1, 2, 3]
[7, 4, 6]
[5, 0, 8]

Step 14:
[1, 2, 3]
[7, 4, 6]
[0, 5, 8]

Step 15:
[1, 2, 3]
[0, 4, 6]
[7, 5, 8]

Step 16:
[1, 2, 3]
[4, 0, 6]
[7, 5, 8]

Step 17:
[1, 2, 3]
[4, 5, 6]
[7, 0, 8]

Step 18:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]

Nodes generated: 3610
Nodes expanded: 1330
```

Manhattan Distance:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =2
Selected heuristic function is Manhattan Distance.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input =1 4 2 3 5 6 7 8 0
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =
Solution found!
Step 0:
[1, 4, 2]
[3, 5, 6]
[7, 8, 0]

Step 1:
[1, 4, 2]
[3, 5, 0]
[7, 8, 6]

Step 2:
[1, 4, 2]
[3, 0, 5]
[7, 8, 6]

Step 3:
[1, 4, 2]
[0, 3, 5]
[7, 8, 6]

Step 4:
[1, 4, 2]
[7, 3, 5]
[0, 8, 6]

Step 5:
[1, 4, 2]
[7, 3, 5]
[8, 0, 6]
```

```
Step 6:
[1, 4, 2]
[7, 3, 5]
[8, 6, 0]

Step 7:
[1, 4, 2]
[7, 3, 0]
[8, 6, 5]

Step 8:
[1, 4, 2]
[7, 0, 3]
[8, 6, 5]

Step 9:
[1, 0, 2]
[7, 4, 3]
[8, 6, 5]

Step 10:
[1, 2, 0]
[7, 4, 3]
[8, 6, 5]

Step 11:
[1, 2, 3]
[7, 4, 0]
[8, 6, 5]

Step 12:
[1, 2, 3]
[7, 4, 5]
[8, 6, 0]

Step 13:
[1, 2, 3]
[7, 4, 5]
[8, 0, 6]
```

```
Step 14:
[1, 2, 3]
[7, 4, 5]
[0, 8, 6]

Step 15:
[1, 2, 3]
[0, 4, 5]
[7, 8, 6]

Step 16:
[1, 2, 3]
[4, 0, 5]
[7, 8, 6]

Step 17:
[1, 2, 3]
[4, 5, 0]
[7, 8, 6]

Step 18:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]

Nodes generated: 511
Nodes expanded: 313
```

Case-3:

Input state :3 4 7 8 1 5 6 0 2

Output state:0 1 2 3 4 5 6 7 8

Misplaced Tiles:

```
Step 14:
[1, 4, 2]
[3, 8, 0]
[6, 5, 7]

Step 15:
[1, 4, 2]
[3, 0, 8]
[6, 5, 7]

Step 16:
[1, 4, 2]
[3, 5, 8]
[6, 0, 7]

Step 17:
[1, 4, 2]
[3, 5, 8]
[6, 7, 0]

Step 18:
[1, 4, 2]
[3, 5, 0]
[6, 7, 8]

Step 19:
[1, 4, 2]
[3, 0, 5]
[6, 7, 8]

Step 20:
[1, 0, 2]
[3, 4, 5]
[6, 7, 8]

Step 21:
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]

Nodes generated: 15462
Nodes expanded: 5739
```

Manhattan Distance:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =2
Selected heuristic function is Manhattan Distance.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input =3 4 7 8 1 5 6 0 2
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =0 1 2 3 4 5 6 7 8
Solution found!
Step 0:
[3, 4, 7]
[8, 1, 5]
[6, 0, 2]

Step 1:
[3, 4, 7]
[8, 1, 5]
[6, 2, 0]

Step 2:
[3, 4, 7]
[8, 1, 0]
[6, 2, 5]

Step 3:
[3, 4, 0]
[8, 1, 7]
[6, 2, 5]

Step 4:
[3, 0, 4]
[8, 1, 7]
[6, 2, 5]

Step 5:
[3, 1, 4]
[8, 0, 7]
[6, 2, 5]

Step 6:
[3, 1, 4]
[8, 2, 7]
[6, 0, 5]
```

```
Step 7:
[3, 1, 4]
[8, 2, 7]
[6, 5, 0]

Step 8:
[3, 1, 4]
[8, 2, 0]
[6, 5, 7]

Step 9:
[3, 1, 4]
[8, 0, 2]
[6, 5, 7]

Step 10:
[3, 1, 4]
[0, 8, 2]
[6, 5, 7]

Step 11:
[0, 1, 4]
[3, 8, 2]
[6, 5, 7]

Step 12:
[1, 0, 4]
[3, 8, 2]
[6, 5, 7]

Step 13:
[1, 4, 0]
[3, 8, 2]
[6, 5, 7]

Step 14:
[1, 4, 2]
[3, 8, 0]
[6, 5, 7]

Step 15:
[1, 4, 2]
[3, 0, 8]
[6, 5, 7]
```

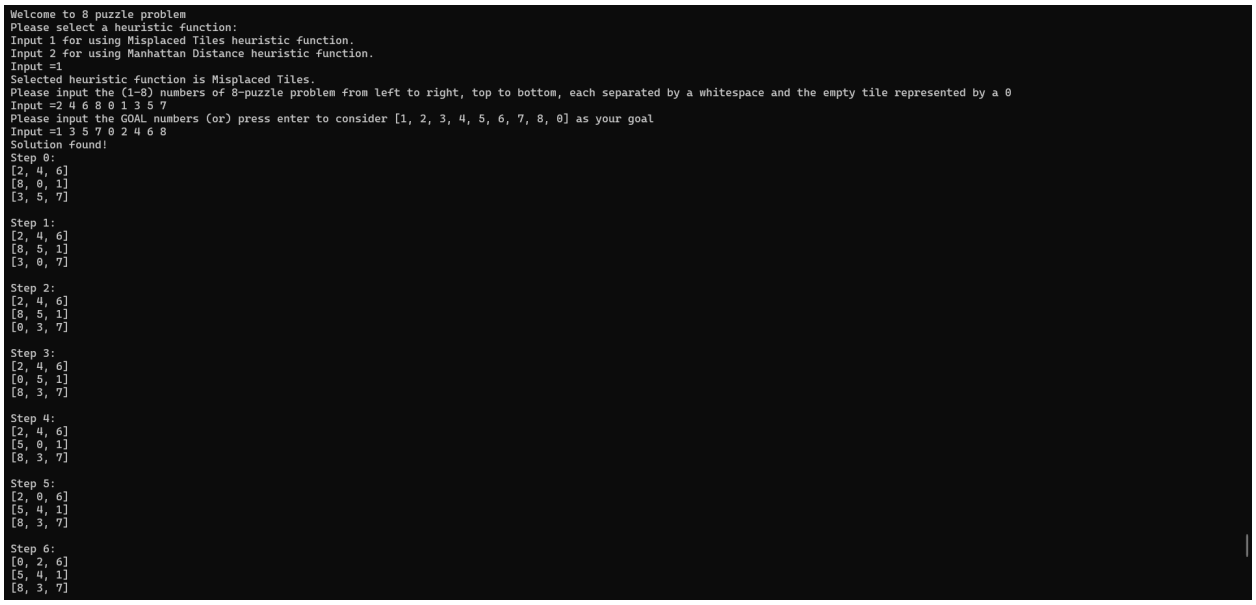
```
Step 16:
[1, 4, 2]
[3, 5, 8]
[6, 0, 7]

Step 17:
[1, 4, 2]
[3, 5, 8]
[6, 7, 0]

Step 18:
[1, 4, 2]
[3, 5, 0]
[6, 7, 8]

Step 19:
[1, 4, 2]
[3, 0, 5]
[6, 7, 8]

Step 20:
[1, 0, 2]
[3, 4, 5]
[6, 7, 8]

Step 21:
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]

Nodes generated: 851
Nodes expanded: 517

Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =
```

Case-4:

Input state: 2 4 6 8 0 1 3 5 7

Output state:1 3 5 7 0 2 4 6 8

Misplaced Tiles:



```
Step 16:
[0, 5, 2]
[3, 1, 6]
[4, 8, 7]

Step 17:
[3, 5, 2]
[0, 1, 6]
[4, 8, 7]

Step 18:
[3, 5, 2]
[1, 0, 6]
[4, 8, 7]

Step 19:
[3, 5, 2]
[1, 6, 0]
[4, 8, 7]

Step 20:
[3, 5, 2]
[1, 6, 7]
[4, 8, 0]

Step 21:
[3, 5, 2]
[1, 6, 7]
[4, 0, 8]

Step 22:
[3, 5, 2]
[1, 0, 7]
[4, 6, 8]

Step 23:
[3, 5, 2]
[1, 7, 0]
[4, 6, 8]

Step 24:
[3, 5, 0]
[1, 7, 2]
[4, 6, 8]
```

```
Step 23:
[3, 5, 2]
[1, 7, 0]
[4, 6, 8]

Step 24:
[3, 5, 0]
[1, 7, 2]
[4, 6, 8]

Step 25:
[3, 0, 5]
[1, 7, 2]
[4, 6, 8]

Step 26:
[0, 3, 5]
[1, 7, 2]
[4, 6, 8]

Step 27:
[1, 3, 5]
[0, 7, 2]
[4, 6, 8]

Step 28:
[1, 3, 5]
[7, 0, 2]
[4, 6, 8]

Nodes generated: 213136
Nodes expanded: 77952
```

Manhattan Distance:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =2
Selected heuristic function is Manhattan Distance.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input =2 4 6 8 0 1 3 5 7
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =1 3 5 7 0 2 4 6 8
Solution found!
Step 0:
[2, 4, 6]
[8, 0, 1]
[3, 5, 7]

Step 1:
[2, 4, 6]
[8, 1, 0]
[3, 5, 7]

Step 2:
[2, 4, 6]
[8, 1, 7]
[3, 5, 0]

Step 3:
[2, 4, 6]
[8, 1, 7]
[3, 0, 5]

Step 4:
[2, 4, 6]
[8, 0, 7]
[3, 1, 5]

Step 5:
[2, 4, 6]
[0, 8, 7]
[3, 1, 5]

Step 6:
[2, 4, 6]
[3, 8, 7]
[0, 1, 5]
```

```
Step 7:
[2, 4, 6]
[3, 8, 7]
[1, 0, 5]

Step 8:
[2, 4, 6]
[3, 0, 7]
[1, 8, 5]

Step 9:
[2, 4, 6]
[3, 7, 0]
[1, 8, 5]

Step 10:
[2, 4, 6]
[3, 7, 5]
[1, 8, 0]

Step 11:
[2, 4, 6]
[3, 7, 5]
[1, 0, 8]

Step 12:
[2, 4, 6]
[3, 0, 5]
[1, 7, 8]

Step 13:
[2, 0, 6]
[3, 4, 5]
[1, 7, 8]

Step 14:
[0, 2, 6]
[3, 4, 5]
[1, 7, 8]

Step 15:
[3, 2, 6]
[0, 4, 5]
[1, 7, 8]
```


```
Step 7:
[2, 4, 6]
[3, 8, 7]
[1, 0, 5]

Step 8:
[2, 4, 6]
[3, 0, 7]
[1, 8, 5]

Step 9:
[2, 4, 6]
[3, 7, 0]
[1, 8, 5]

Step 10:
[2, 4, 6]
[3, 7, 5]
[1, 8, 0]

Step 11:
[2, 4, 6]
[3, 7, 5]
[1, 0, 8]

Step 12:
[2, 4, 6]
[3, 0, 5]
[1, 7, 8]

Step 13:
[2, 0, 6]
[3, 4, 5]
[1, 7, 8]

Step 14:
[0, 2, 6]
[3, 4, 5]
[1, 7, 8]

Step 15:
[3, 2, 6]
[0, 4, 5]
[1, 7, 8]
```

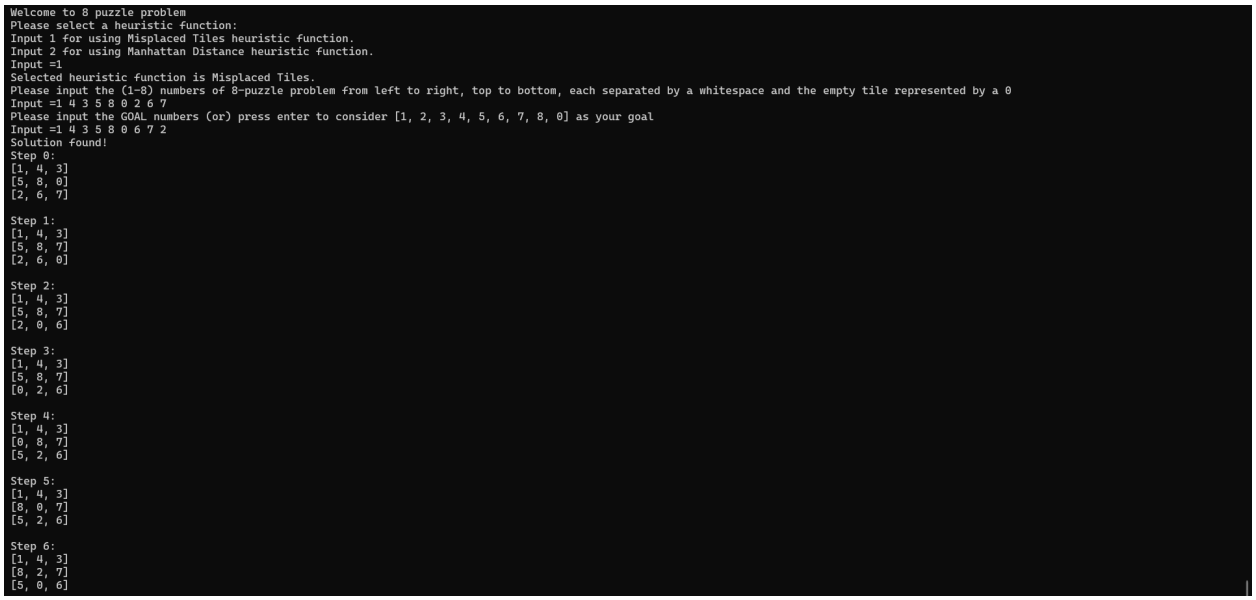

```
Step 25:
[1, 3, 5]
[0, 6, 2]
[7, 4, 8]

Step 26:
[1, 3, 5]
[7, 6, 2]
[0, 4, 8]

Step 27:
[1, 3, 5]
[7, 6, 2]
[4, 0, 8]

Step 28:
[1, 3, 5]
[7, 0, 2]
[4, 6, 8]

Nodes generated: 3910
Nodes expanded: 2391
```

Case-5:

Input state:1 4 3 5 8 0 2 6 7

Output state:1 4 3 5 8 0 6 7 2

Misplaced Tiles:



Manhattan Distance:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =2
Selected heuristic function is Manhattan Distance.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input =1 4 3 5 8 0 2 6 7
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =1 4 3 5 8 0 6 7 2
Solution found!
Step 0:
[1, 4, 3]
[5, 8, 0]
[2, 6, 7]

Step 1:
[1, 4, 3]
[5, 8, 7]
[2, 6, 0]

Step 2:
[1, 4, 3]
[5, 8, 7]
[2, 0, 6]

Step 3:
[1, 4, 3]
[5, 8, 7]
[0, 2, 6]

Step 4:
[1, 4, 3]
[0, 8, 7]
[5, 2, 6]

Step 5:
[1, 4, 3]
[8, 0, 7]
[5, 2, 6]

Step 6:
[1, 4, 3]
[8, 2, 7]
[5, 0, 6]
```

```
Step 7:
[1, 4, 3]
[8, 2, 7]
[5, 6, 0]

Step 8:
[1, 4, 3]
[8, 2, 0]
[5, 6, 7]

Step 9:
[1, 4, 3]
[8, 0, 2]
[5, 6, 7]

Step 10:
[1, 4, 3]
[0, 8, 2]
[5, 6, 7]

Step 11:
[1, 4, 3]
[5, 8, 2]
[0, 6, 7]

Step 12:
[1, 4, 3]
[5, 8, 2]
[6, 0, 7]

Step 13:
[1, 4, 3]
[5, 8, 2]
[6, 7, 0]

Step 14:
[1, 4, 3]
[5, 8, 0]
[6, 7, 2]

Nodes generated: 210
Nodes expanded: 125
```

Case-6:

Input state: 0 8 7 6 5 1 2 3 4

Output state:1 2 3 4 5 6 7 8 0

Misplaced Tiles:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =1
Selected heuristic function is Misplaced Tiles.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input = 0 8 7 6 5 1 2 3 4
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =
Solution found!
Step 0:
[0, 8, 7]
[6, 5, 1]
[2, 3, 4]

Step 1:
[6, 8, 7]
[0, 5, 1]
[2, 3, 4]

Step 2:
[6, 8, 7]
[5, 0, 1]
[2, 3, 4]

Step 3:
[6, 8, 7]
[5, 1, 0]
[2, 3, 4]

Step 4:
[6, 8, 0]
[5, 1, 7]
[2, 3, 4]

Step 5:
[6, 0, 8]
[5, 1, 7]
[2, 3, 4]

Step 6:
[6, 1, 8]
[5, 0, 7]
[2, 3, 4]
```

```
Step 7:
[6, 1, 8]
[5, 3, 7]
[2, 0, 4]

Step 8:
[6, 1, 8]
[5, 3, 7]
[0, 2, 4]

Step 9:
[6, 1, 8]
[0, 3, 7]
[5, 2, 4]

Step 10:
[0, 1, 8]
[6, 3, 7]
[5, 2, 4]

Step 11:
[1, 0, 8]
[6, 3, 7]
[5, 2, 4]

Step 12:
[1, 3, 8]
[6, 0, 7]
[5, 2, 4]

Step 13:
[1, 3, 8]
[6, 2, 7]
[5, 0, 4]

Step 14:
[1, 3, 8]
[6, 2, 7]
[5, 4, 0]

Step 15:
[1, 3, 8]
[6, 2, 0]
[5, 4, 7]
```

```
Step 16:
[1, 3, 0]
[6, 2, 8]
[5, 4, 7]

Step 17:
[1, 0, 3]
[6, 2, 8]
[5, 4, 7]

Step 18:
[1, 2, 3]
[6, 0, 8]
[5, 4, 7]

Step 19:
[1, 2, 3]
[0, 6, 8]
[5, 4, 7]

Step 20:
[1, 2, 3]
[5, 6, 8]
[0, 4, 7]

Step 21:
[1, 2, 3]
[5, 6, 8]
[4, 0, 7]

Step 22:
[1, 2, 3]
[5, 6, 8]
[4, 7, 0]

Step 23:
[1, 2, 3]
[5, 6, 0]
[4, 7, 8]

Step 24:
[1, 2, 3]
[5, 0, 6]
[4, 7, 8]
```

```
Step 25:
[1, 2, 3]
[0, 5, 6]
[4, 7, 8]

Step 26:
[1, 2, 3]
[4, 5, 6]
[0, 7, 8]

Step 27:
[1, 2, 3]
[4, 5, 6]
[7, 0, 8]

Step 28:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]

Nodes generated: 186113
Nodes expanded: 68936
```

Manhattan Distance:

```
Welcome to 8 puzzle problem
Please select a heuristic function:
Input 1 for using Misplaced Tiles heuristic function.
Input 2 for using Manhattan Distance heuristic function.
Input =2
Selected heuristic function is Manhattan Distance.
Please input the (1-8) numbers of 8-puzzle problem from left to right, top to bottom, each separated by a whitespace and the empty tile represented by a 0
Input = 0 8 7 6 5 1 2 3 4
Please input the GOAL numbers (or) press enter to consider [1, 2, 3, 4, 5, 6, 7, 8, 0] as your goal
Input =
Solution found!
Step 0:
[0, 8, 7]
[6, 5, 1]
[2, 3, 4]

Step 1:
[6, 8, 7]
[0, 5, 1]
[2, 3, 4]

Step 2:
[6, 8, 7]
[5, 0, 1]
[2, 3, 4]

Step 3:
[6, 0, 7]
[5, 8, 1]
[2, 3, 4]

Step 4:
[6, 7, 0]
[5, 8, 1]
[2, 3, 4]

Step 5:
[6, 7, 1]
[5, 8, 0]
[2, 3, 4]

Step 6:
[6, 7, 1]
[5, 0, 8]
[2, 3, 4]
```

```
Step 7:
[6, 7, 1]
[5, 3, 8]
[2, 0, 4]

Step 8:
[6, 7, 1]
[5, 3, 8]
[2, 4, 0]

Step 9:
[6, 7, 1]
[5, 3, 0]
[2, 4, 8]

Step 10:
[6, 7, 1]
[5, 0, 3]
[2, 4, 8]

Step 11:
[6, 0, 1]
[5, 7, 3]
[2, 4, 8]

Step 12:
[0, 6, 1]
[5, 7, 3]
[2, 4, 8]

Step 13:
[5, 6, 1]
[0, 7, 3]
[2, 4, 8]

Step 14:
[5, 6, 1]
[2, 7, 3]
[0, 4, 8]

Step 15:
[5, 6, 1]
[2, 7, 3]
[4, 0, 8]
```

```
Step 16:
[5, 6, 1]
[2, 0, 3]
[4, 7, 8]

Step 17:
[5, 0, 1]
[2, 6, 3]
[4, 7, 8]

Step 18:
[5, 1, 0]
[2, 6, 3]
[4, 7, 8]

Step 19:
[5, 1, 3]
[2, 6, 0]
[4, 7, 8]

Step 20:
[5, 1, 3]
[2, 0, 6]
[4, 7, 8]

Step 21:
[5, 1, 3]
[0, 2, 6]
[4, 7, 8]

Step 22:
[0, 1, 3]
[5, 2, 6]
[4, 7, 8]

Step 23:
[1, 0, 3]
[5, 2, 6]
[4, 7, 8]

Step 24:
[1, 2, 3]
[5, 0, 6]
[4, 7, 8]
```

```
Step 25:
[1, 2, 3]
[0, 5, 6]
[4, 7, 8]

Step 26:
[1, 2, 3]
[4, 5, 6]
[0, 7, 8]

Step 27:
[1, 2, 3]
[4, 5, 6]
[7, 0, 8]

Step 28:
[1, 2, 3]
[4, 5, 6]
[7, 8, 0]

Nodes generated: 2006
Nodes expanded: 1250
```

7.Summary:

Case	Initial State	Goal State	Misplaced Tiles		Manhattan Distance	
			Generated Nodes	Explored Nodes	Generated Nodes	Explored Nodes
1	1 2 3 4 5 0 7 8 6	1 2 3 4 5 6 7 8 0	4	2	4	2
2	1 4 2 3 5 6 7 8 0	1 2 3 4 5 6 7 8 0	3618	1330	511	313
3	3 4 7 8 1 5 6 0 2	0 1 2 3 4 5 6 7 8	15462	5739	851	517
4	2 4 6 8 0 1 3 5 7	1 3 5 7 0 2 4 6 8	213136	77952	3910	2391
5	1 4 3 5 8 0 2 6 7	1 4 3 5 8 0 6 7 2	819	297	210	125
6	0 8 7 6 5 1 2 3 4	1 2 3 4 5 6 7 8 0	186113	68936	2006	1250

8.Conclusion:

The intricacy of potential moves leading to the end goal in games like the 8-puzzle is vast. For many computer systems, identifying the right sequence for a specific 8-puzzle layout can be time-consuming, especially when using an undirected search method. To address this, we equipped the computer to prioritize moves in a way that it quickly identifies the most efficient route to victory. We employed the A* search algorithm, a guided search technique, to tackle this puzzle challenge. Our approach incorporates two distinct "Guided strategies", namely, Manhattan distance and Misplaced tile heuristic functions to ensure an optimal solution pathway. Both strategies determine heuristic values for each state, known as ‘h’ and ‘g’ values, which represent step counts. By combining ‘h’ and ‘g’, we get the ‘f’ value. During each move, the state with the lowest ‘f’ value is selected for further exploration. This systematic approach ensures we reach the desired outcome with utmost efficiency, avoiding a purely trial-and-error method.

9. References:

- https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/
- https://www.geeksforgeeks.org/a-search-algorithm/
- https://en.wikipedia.org/wiki/A*_search_algorithm