8 Puzzle Solution Using A* Algorithm
ITCS 6150

Team Details

Ramanathan Sivaramakrishnan - rsivara1@uncc.edu - 801243565
Siddharth Jayachandra Babu - sjayach3@uncc.edu - 801252829
Sneha Srikanth - ssrikan1@uncc.edu - 801252592

Introduction

Problem Statement:

Noyes Palmer Chapman's 8-puzzle problem is a puzzle that he created. It's played on a 3x3 grid with eight square blocks labeled 1 through 8, as well as a blank square or 0. The objective is to rearrange and provide the 8 block according to the provided goal state using the A* search algorithm and two heuristic functions of the 8-puzzle problem .

A* ALGORITHM:

A* is an informed search method that starts at a certain node in a graph and seeks to discover the cheapest path to the provided goal node. It is a combination of uniform cost search and best first search which avoids expanding the paths that are already expensive.As A* uses admissible heuristics which never overestimates the cost to reach the goal. It decides which path to expand at each iteration based on the following evaluation function:

- ➔ **f(n) = g(n)+h(n)**
- ➔ g(n) = path cost from the start node to node(n)
- ➔ h(n) =estimated cost of cheapest path from n to the goal we have.

Problem Formulation

Given Parameters: We are provided with the Initial state and Goal State.

States: This is a two dimensional array which represents the current state and the heuristics are calculated based on the tile position which represented as (i,j).

Actions: The moves happen in left, up, down and right positions based on the position of the free tile (i.e: 0).

Performance: The total Number of moves required to achieve the Goal state.

Heuristic Functions:

When applied to a state, h(Heuristic) produces a number that represents an evaluation of the state's merit in relation to the goal.
Heuristic functions are divided into two categories.

1. **Misplaced tiles**: By comparing the initial and goal states, it determines how many tiles are misplaced.

2. **Manhattan distance:** It determines how many movements are required to transfer all of the misplaced tiles to the desired or specified spot.Which is the distance between two titles measured along the axes of right angles. Which based on the difference between the coordinates of goals(x1,y1) and the initial state(x2,y2) by using the formula of |x1-x2|+|y1-y2|.

Program Structure

Class used:

- ➔ **Node**: The node class contains the essential functionalities required to complete the problem. Some of the functions involve moving of the tile and also for producing child nodes for the current state that can be used for expanding to reach the goal state.
- ➔ **puzzle_solver** : The puzzle class accepts input from the user for getting the initial state of the puzzle and the goal state of the puzzle.This also contains a function for calculating the heuristic value that is used to attain the goal state.
- ➔ We are stopping the program when the close_list number of nodes expanded exceeds 200.

Global variables

No global variables are used in the implementation.

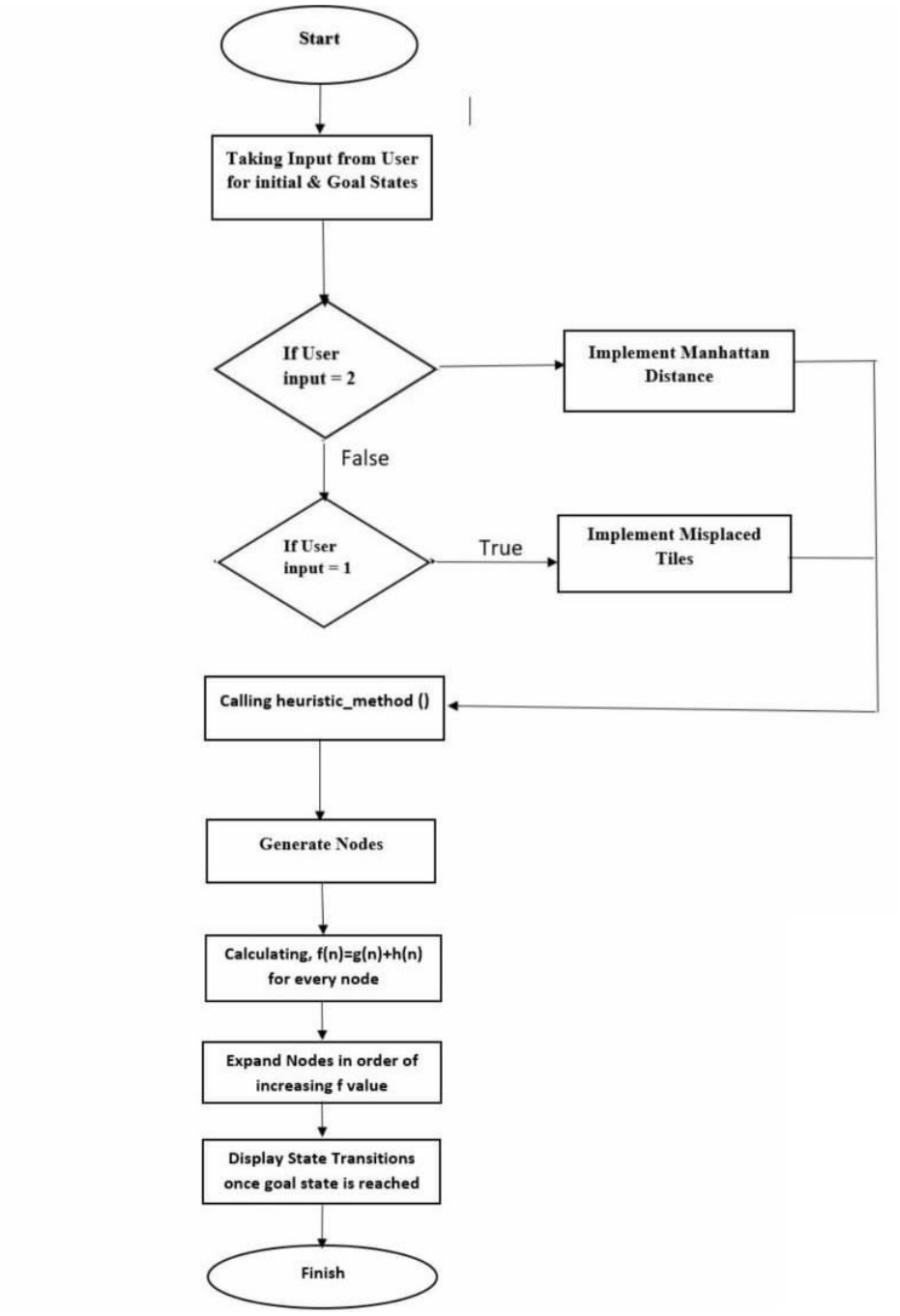
Functions Used

- ➔ **__init__** :Used to set up the class's attributes.This function is used in both Node and puzzle_solver class.In the Node class it is used to set up the node's state,path_cost,heuristic function,total_cost,parent_node and the corresponding node's children list.
- ➔ **get_zero_index** :This function helps to get the free tile (i.e 0) inside the board.
- ➔ **create_childnodes** : All the possible outcomes are generated for a particular state by rearranging the tiles into the free tile positions. (i.e: 0 positions)
- ➔ **move_tile** :The tile is positioned in the available space with the help of this function .
- ➔ **input_matrix** :This function is used to get the initial state of the puzzle and the goal state of the puzzle.
- ➔ **read states** :Reading the current state and setting up the node count to zero is the purpose of this function.
- ➔ **get_heuristic** :This function helps the user to inquire about the preferred heuristic.
- ➔ **heuristic_function**:Based on the user's preference this function helps to calculate the heuristic value.
- ➔ **node_selected** :This function is used to display the current node state, its path cost g(n) and the heuristic value h(n).
- ➔ **solver_function**:This function is the main function that is used to create a node and generate all it's possible states and also checks whether the goal state has been reached or not.

Data Structure Used:

We have used List Data Structure from python.We have declared two lists **open_list** and **close_list.** The **open_list** is used to maintain the possible outcomes for a node that aren't yet visited and the **closed_list** contains those that are visited.

Flowchart:


Start
Taking Input from User for initial & Goal States
If User input = 2
Implement Manhattan Distance
False
If User input = 1
True
Implement Misplaced Tiles
Calling heuristic_method ()
Generate Nodes
Calculating, f(n)=g(n)+h(n) for every node
Expand Nodes in order of increasing f value
Display State Transitions once goal state is reached
Finish

Source Code:

```
import copy

class Node:

    # We initialize the new node with the data which is required.

    def __init__(self,data,path_cost,heuristic,parent):

        self.state = data

        self.path_cost = path_cost# path cost is the level.

        # heuristic: heuristic value(1 or 2 which is misplaced tiles or manhattan distance).

        self.heuristic_func = heuristic

        self.Tcost = path_cost+ heuristic # Tcost : total cost.

        self.parent_node = parent#parent of the node that is getting initialized.

        self.child_nodes = []#The possible outcomes for the current node_state is stored in this list.

# index function returns position of 0 in the **8 puzzle in the intial_state or in the goal_state.

    def get_zero_index(self,matrix,k):

        for i in range(0,len(self.state)):

            for j in range(0,len(self.state)):

                if matrix[i][j] == k:

                    return i,j
```

```
# Move the 0 in the given direction and if the position is out of limits the
return None.

    def move_title(self,first,second):

        posi_1 = first[0]

        posj_1 = first[1]

        posi_2 = second[0]

        posj_2= second[1]

#copy library of python is used to create a deepcopy of the current state that is
being changed based on the free title which is 0.

        temp_state = copy.deepcopy(self.state)

        temp = temp_state[posi_1][posj_1]

        temp_state[posi_1][posj_1] = temp_state[posi_2][posj_2]

        temp_state[posi_2][posj_2] = temp

        return temp_state

# Moving the 0 in either left, right, up or down directions.

    def create_childnodes(self):

        self.zero = self.get_zero_index(self.state,0)

        ith_pos,jth_pos = self.get_zero_index(self.state,0)

        left = jth_pos - 1

        down = ith_pos + 1

        up = ith_pos - 1

        right = jth_pos + 1

        self.generate_child_nodes = []

        no_rows = 2
```

```
        no_cols = 2
        if(right <= no_cols):
            self.generate_child_nodes.append((ith_pos, right))
            puzzle.generated_count+=1
        if(up >= 0):
            self.generate_child_nodes.append((up, jth_pos))
            puzzle.generated_count+=1
        if(left >= 0):
            self.generate_child_nodes.append((ith_pos, left))
            puzzle.generated_count+=1
        if(down <= no_rows):
            self.generate_child_nodes.append((down, jth_pos))
            puzzle.generated_count+=1
class puzzle_solver:
    def __init__(self):
        self.open_list = []
        self.close_list = []
# takes the input for the puzzle from the user and returns the input state as a
3*3 matrix
#And it is also used to get the goal state from the user.
    def input_matrix(self,name_state):
            print("Enter the "+name_state+" state matrix values:")
            state=[]
```

```
        for i in range(0,9):
            n=int(input("Enter the values:"+name_state+":"))
            state.append(n)
        return [state[0:3],state[3:6],state[6:9]]
# Read the states {start,goal} and initialize node count to 0
    def initaliasing_states(self):
        self.start_state = self.input_matrix("initial")
        self.goal_state = self.input_matrix("Goal")
        self.node_count = 0
# get the heuristic {Misplaced tiles, Manhatten distance} from the user
    def get_heuristic(self):
        while True:
            self.heuristic_func = input("\n Choose a heuristic approach:\n 1.
Misplaced Tiles or 2. Manhattan Distance:\n")
            if self.heuristic_func == '1' or self.heuristic_func == '2':
                break
            else:print("Select a Valid Input")
# get_zeroth_index function returns position of tile with 0 in the given puzzle.
    def get_zeroth_index(self,matrix,num):
        for i in range(0,len(matrix)):
            for j in range(0,len(matrix)):
                if matrix[i][j] == num:
                    return i,j
```

```
# calculate the h(n) based on the user selected
    def heuristic_functions(self,current_state,goal_state):
        if self.heuristic_func == '1':
            misplaced_tiles = 0
            for i in range(0,9):
                current_state_position = self.get_zeroth_index(current_state,i)
                goal_state_position = self.get_zeroth_index(goal_state,i)
                if current_state_position != goal_state_position:
                    misplaced_tiles += 1
            return misplaced_tiles
        else:
            manhatten_value = 0
            for num in range(0,9):
                current_state_position = self.get_zeroth_index(current_state,num)
                goal_state_position = self.get_zeroth_index(goal_state,num)
                total_distance = abs(current_state_position[0] - goal_state_position[0])
+ abs(current_state_position[1] - goal_state_position[1])
                manhatten_value += total_distance
            return manhatten_value
# selected node with f,g and h values
    def node_selected(self,node,info = True):
        current_state = node
        if info == True:
```

```
    print("The path_cost g(n) = ", node.path_cost, ", heuristic value h(n) =
",node.heuristic_func, "And the total cost to reach the goal state f(n) =
g(n)+h(n) = ",node.Tcost)

            current_state = node.state

            print(current_state)

# this is main function solving of the puzzle starts from here

    def solver_function(self):

        self.generated_count = 0

        self.initaliasing_states()

        self.get_heuristic()

        initial_hueristic = self.heuristic_functions(self.start_state,self.goal_state)

        # initialize start state and append the start node to opened list

        start_state = Node(self.start_state,0,initial_hueristic,None)

        self.open_list.append(start_state)

        while True:

            current_node = self.open_list[0]

            self.node_count += 1

            self.node_selected(current_node)

            if current_node.heuristic_func == 0:

                print("Goal state has been reached\n\n")

                break

            current_node.create_childnodes()

            for node in current_node.generate_child_nodes:

                temp_node = current_node.move_element(node,current_node.zero)
```

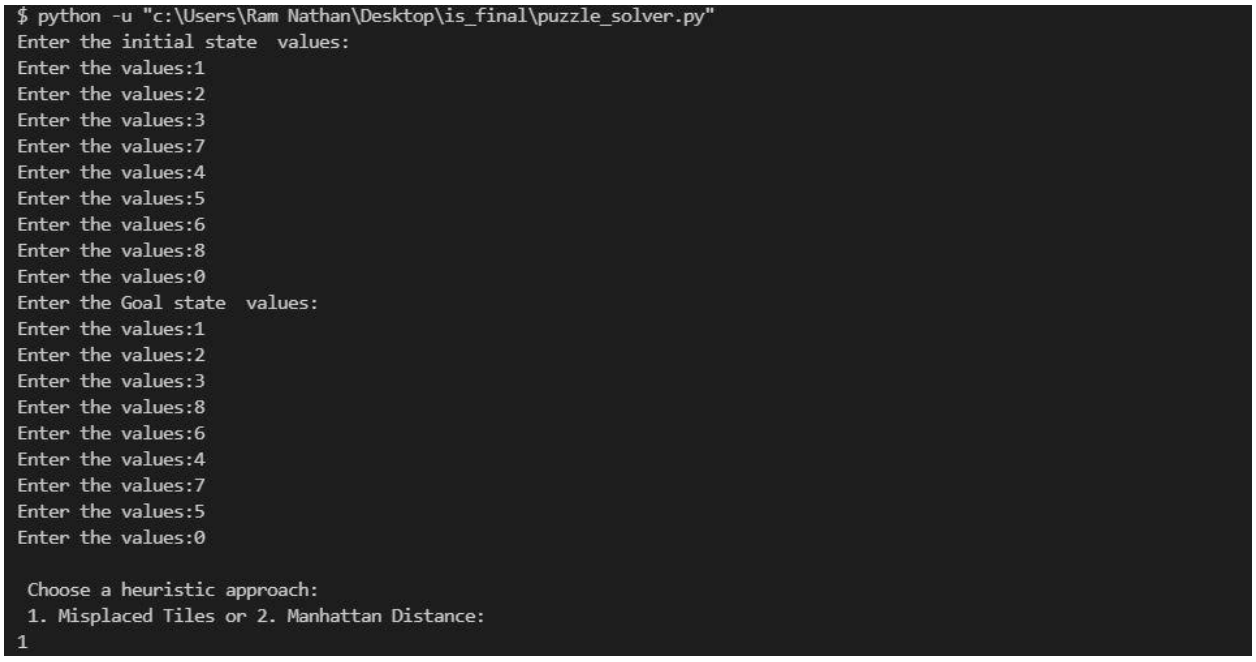
```
        #Calculating the heuristic value using the heuristic function
        temp_hueristic = self.heuristic_functions(temp_node,self.goal_state)

current_node.child_nodes.append(Node(temp_node,current_node.path_cost+1
,temp_hueristic,current_node))
        for node in current_node.child_nodes:
            self.open_list.append(node)
        #if a node is visited it is appended to the close_list
        self.close_list.append(current_node)
        del self.open_list[0]
        self.open_list.sort(key = lambda val:val.Tcost,reverse=False)
        # if the program is unable to find solution after 500 iterations then we
end it saying no solution is found
        if self.node_count > 150:
            print("Unable to get a solution after 150 iterations!!")
            break
#the execution of the program starts from here
if __name__=="__main__":
    puzzle = puzzle_solver()
    puzzle.solver_function()
    print("Number of nodes generated: ", puzzle.generated_count)
    print("Number of  nodes expanded: ", len(puzzle.close_list))
```

Input/Output 1:

Sample	Initial State	Goal State	Number of Nodes Generated (misplaced Titles)	Number of Nodes Expanded (misplaced Titles)	Number of Nodes Generated (manhattan distance)	Number of Nodes Expanded (manhattan distance)
Sample Input 1	[1 2 3] [7 4 5] [6 8 0]	[1 2 3] [8 6 4] [7 5 0]	137	48	54	20

Screenshots:

Getting Input:



Misplaced Titles Output:

```
The path_cost g(n) =  0 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  5

1 2 3
7 4 5
6 8 0
The path_cost g(n) =  1 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7

1 2 3
7 4 0
6 8 5
The path_cost g(n) =  1 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7

1 2 3
7 4 5
6 0 8
The path_cost g(n) =  2 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7

1 2 3
7 0 4
6 8 5
The path_cost g(n) =  2 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7

1 2 3
7 4 5
6 8 0
The path_cost g(n) =  2 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7

1 2 3
7 4 5
6 8 0
The path_cost g(n) =  2 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8

1 2 3
7 0 5
6 4 8
The path_cost g(n) =  2 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8

1 2 3
7 4 5
0 6 8
```

```
The path_cost g(n) =  4 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7

1 2 3
7 8 4
6 5 0
The path_cost g(n) =  3 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8

1 2 3
0 4 5
7 6 8
The path_cost g(n) =  2 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 0
7 4 3
6 8 5
The path_cost g(n) =  3 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
7 4 0
6 8 5
The path_cost g(n) =  3 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 0 3
7 2 4
6 8 5
The path_cost g(n) =  3 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
7 4 0
6 8 5
The path_cost g(n) =  3 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
7 4 5
6 0 8
The path_cost g(n) =  3 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
7 4 0
6 8 5
```

```
The path_cost g(n) =  4 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
7 0 4
6 8 5
The path_cost g(n) =  4 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
7 4 5
6 8 0
The path_cost g(n) =  6 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
8 0 4
7 6 5
The path_cost g(n) =  6 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
4 6 5
7 8 0
The path_cost g(n) =  7 , heuristic value h(n) =  2 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9

1 2 3
8 6 4
7 0 5
The path_cost g(n) =  8 , heuristic value h(n) =  0 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8

1 2 3
8 6 4
7 5 0
Goal state has been reached


Number of nodes generated:  137
Number of  nodes expanded:  48
```

Manhattan Distance Output:

```
 Choose a heuristic approach:
 1. Misplaced Tiles or 2. Manhattan Distance:
2
The path_cost g(n) =  0 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
1 2 3
7 4 5
6 8 0
The path_cost g(n) =  1 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9
1 2 3
7 4 0
6 8 5
The path_cost g(n) =  2 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
7 0 4
6 8 5
The path_cost g(n) =  3 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9
1 2 3
7 8 4
6 0 5
The path_cost g(n) =  4 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
1 2 3
7 8 4
6 5 0
The path_cost g(n) =  2 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
7 4 5
6 8 0
The path_cost g(n) =  4 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
7 8 4
0 6 5
The path_cost g(n) =  1 , heuristic value h(n) =  10 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
1 2 3
7 4 5
6 0 8
The path_cost g(n) =  2 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
7 4 5
6 8 0
```

```
6 5 0
The path_cost g(n) =  5 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
1 2 3
7 8 4
6 0 5
The path_cost g(n) =  6 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
7 8 4
6 5 0
The path_cost g(n) =  3 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
1 2 3
7 4 0
6 8 5
The path_cost g(n) =  5 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
1 2 3
7 8 4
6 0 5
The path_cost g(n) =  6 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
7 8 4
6 5 0
The path_cost g(n) =  5 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
1 2 3
0 8 4
7 6 5
The path_cost g(n) =  6 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
1 2 3
8 0 4
7 6 5
The path_cost g(n) =  7 , heuristic value h(n) =  2 And the total cost to reach the goal state f(n) = g(n)+h(n) =  9
1 2 3
8 6 4
7 0 5
The path_cost g(n) =  8 , heuristic value h(n) =  0 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
1 2 3
8 6 4
7 5 0
Goal state has been reached


Number of nodes generated:  54
Number of  nodes expanded:  20
```

Input/Output 2:

Sample	Initial State	Goal State	Number of Nodes Generated (misplaced titles)	Number of Nodes Expanded (misplaced titles)	Number of Nodes Generated (manhattan distance)	Number of Nodes Expanded (manhattan distance)
Sample Input 2	[2 8 1] [3 4 6] [7 5 0]	[3 2 1] [8 0 4] [7 5 6]	28	9	17	6

Getting Input:

```
$ python -u "c:\Users\Ram Nathan\Desktop\is_final\puzzle_solver.py"
Enter the initial state  values:
Enter the values:2
Enter the values:8
Enter the values:1
Enter the values:3
Enter the values:4
Enter the values:6
Enter the values:7
Enter the values:5
Enter the values:0
Enter the Goal state  values:
Enter the values:3
Enter the values:2
Enter the values:1
Enter the values:8
Enter the values:0
Enter the values:4
Enter the values:7
Enter the values:5
Enter the values:6
```

Misplaced Titles Output:

```
 Choose a heuristic approach:
 1. Misplaced Tiles or 2. Manhattan Distance:
1
The path_cost g(n) =  0 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  6
2 8 1
3 4 6
7 5 0
The path_cost g(n) =  1 , heuristic value h(n) =  5 And the total cost to reach the goal state f(n) = g(n)+h(n) =  6
2 8 1
3 4 0
7 5 6
The path_cost g(n) =  2 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  5
2 8 1
3 0 4
7 5 6
The path_cost g(n) =  3 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
2 0 1
3 8 4
7 5 6
The path_cost g(n) =  3 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
2 8 1
0 3 4
7 5 6
The path_cost g(n) =  4 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
0 2 1
3 8 4
7 5 6
The path_cost g(n) =  4 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
2 8 1
3 0 4
7 5 6
```

```
The path_cost g(n) =  4 , heuristic value h(n) =  3 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
2 8 1
3 0 4
7 5 6
The path_cost g(n) =  5 , heuristic value h(n) =  2 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
3 2 1
0 8 4
7 5 6
The path_cost g(n) =  6 , heuristic value h(n) =  0 And the total cost to reach the goal state f(n) = g(n)+h(n) =  6
3 2 1
8 0 4
7 5 6
Goal state has been reached


Number of nodes generated:  28
Number of  nodes expanded:  9
```

Manhattan Distance Output:

```
 Choose a heuristic approach:
 1. Misplaced Tiles or 2. Manhattan Distance:
2
The path_cost g(n) =  0 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
2 8 1
3 4 6
7 5 0
The path_cost g(n) =  1 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
2 8 1
3 4 0
7 5 6
The path_cost g(n) =  2 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  6
2 8 1
3 0 4
7 5 6
The path_cost g(n) =  3 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
2 0 1
3 8 4
7 5 6
The path_cost g(n) =  4 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
0 2 1
3 8 4
7 5 6
The path_cost g(n) =  5 , heuristic value h(n) =  2 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
3 2 1
0 8 4
7 5 6
The path_cost g(n) =  6 , heuristic value h(n) =  0 And the total cost to reach the goal state f(n) = g(n)+h(n) =  6
3 2 1
8 0 4
7 5 6
Goal state has been reached


Number of nodes generated:  17
Number of  nodes expanded:  6
```

Input/Output 3:

Sample	Initial State	Goal State	Number of Nodes Generated (misplaced titles)	Number of Nodes Expanded (misplaced titles)	Number of Nodes Generated (Manhattan Distance)	Number of Nodes Expanded (Manhattan Distance)
Sample 3	[7 2 4] [5 0 6] [8 3 1]	[1 2 3] [4 5 6] [7 8 0]	578 (stopped at) Unable to Find solution	201 (stopped at) Unable to Find solution	538 (stopped at) Unable to Find solution	201 (stopped at) Unable to Find solution

Getting Input:

Misplaced Titles Output:

```
 Choose a heuristic approach:
 1. Misplaced Tiles or 2. Manhattan Distance:
1
The path_cost g(n) =  0 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
7 2 4
5 0 6
8 3 1
The path_cost g(n) =  1 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  7
7 2 4
0 5 6
8 3 1
The path_cost g(n) =  1 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  2 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
0 2 4
7 5 6
8 3 1
The path_cost g(n) =  2 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
7 2 4
8 5 6
0 3 1
The path_cost g(n) =  2 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
7 2 4
5 3 6
8 1 0
The path_cost g(n) =  2 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  8
7 2 4
5 3 6
0 8 1
```

```
 The path_cost g(n) =  4 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
 7 2 4
 5 3 6
 8 1 0
 The path_cost g(n) =  4 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
 7 2 4
 5 3 6
 0 8 1
 The path_cost g(n) =  3 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
 7 2 4
 5 3 0
 8 1 6
 The path_cost g(n) =  4 , heuristic value h(n) =  6 And the total cost to reach the goal state f(n) = g(n)+h(n) =  10
 7 2 4
 5 3 6
 8 1 0
 The path_cost g(n) =  3 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
 7 2 4
 5 6 0
 8 3 1
 The path_cost g(n) =  3 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
 7 0 4
 5 2 6
 8 3 1
 The path_cost g(n) =  3 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
 7 2 4
 5 6 0
 8 3 1
 The path_cost g(n) =  3 , heuristic value h(n) =  8 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
 7 0 4
 5 2 6
 8 3 1
 The path_cost g(n) =  4 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  11
 7 2 4
 5 0 6
 8 3 1
```

```
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
2 0 4
7 5 6
8 3 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  5 , heuristic value h(n) =  7 And the total cost to reach the goal state f(n) = g(n)+h(n) =  12
7 2 4
5 3 6
8 0 1
Unable to get a solution after 200 iterations!!
Number of nodes generated:  578
Number of  nodes expanded:  201
```

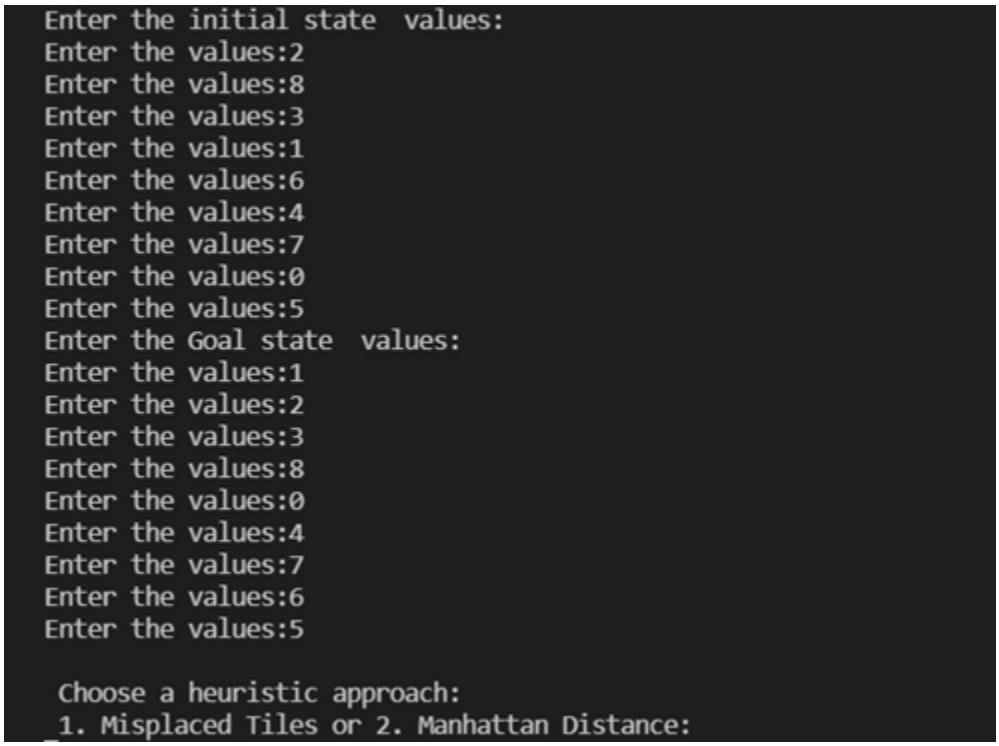
Manhattan Distance Output:

```
The path_cost g(n) =  6 , heuristic value h(n) =  12 And the total cost to reach the goal state f(n) = g(n)+h(n) =  18
7 2 4
5 3 6
8 1 0
The path_cost g(n) =  4 , heuristic value h(n) =  16 And the total cost to reach the goal state f(n) = g(n)+h(n) =  20
7 2 4
5 0 6
8 3 1
The path_cost g(n) =  5 , heuristic value h(n) =  14 And the total cost to reach the goal state f(n) = g(n)+h(n) =  19
7 2 4
5 3 6
8 0 1
The path_cost g(n) =  6 , heuristic value h(n) =  12 And the total cost to reach the goal state f(n) = g(n)+h(n) =  18
7 2 4
5 3 6
8 1 0
The path_cost g(n) =  4 , heuristic value h(n) =  16 And the total cost to reach the goal state f(n) = g(n)+h(n) =  20
0 2 4
7 5 6
8 3 1
The path_cost g(n) =  4 , heuristic value h(n) =  16 And the total cost to reach the goal state f(n) = g(n)+h(n) =  20
7 2 4
8 5 6
0 3 1
The path_cost g(n) =  6 , heuristic value h(n) =  14 And the total cost to reach the goal state f(n) = g(n)+h(n) =  20
7 2 0
5 3 4
8 1 6
Unable to get a solution after 200 iterations!!
Number of nodes generated:  538
Number of  nodes expanded:  201
```

Input/Output 4:

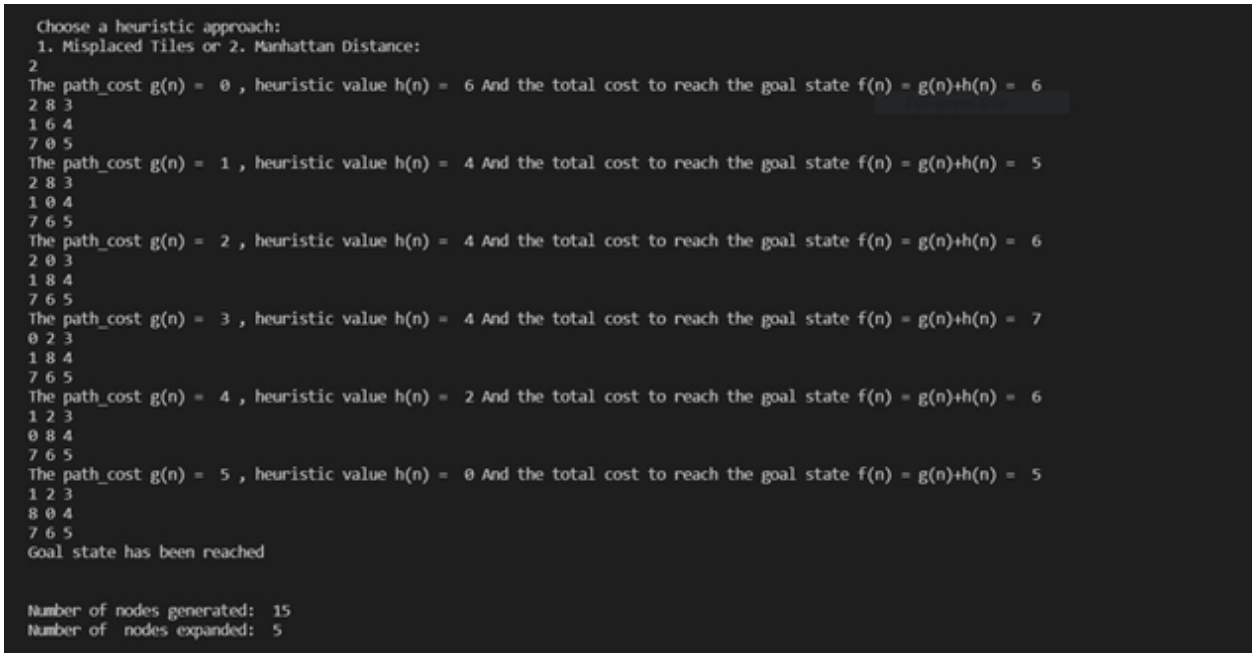
Sample	Initial State	Goal State	Number of nodes generated (Misplaced tiles)	Number of nodes expanded (Misplaced tiles)	Number of nodes generated(Manhattan Distance)	Number of nodes expanded (Manhattan Distance)
Sample 4	[2 8 3] [1 6 4] [7 0 5]	[1 2 3] [8 0 4] [7 6 5]	26	8	15	5

Getting Input:



Misplaced Titles Output:

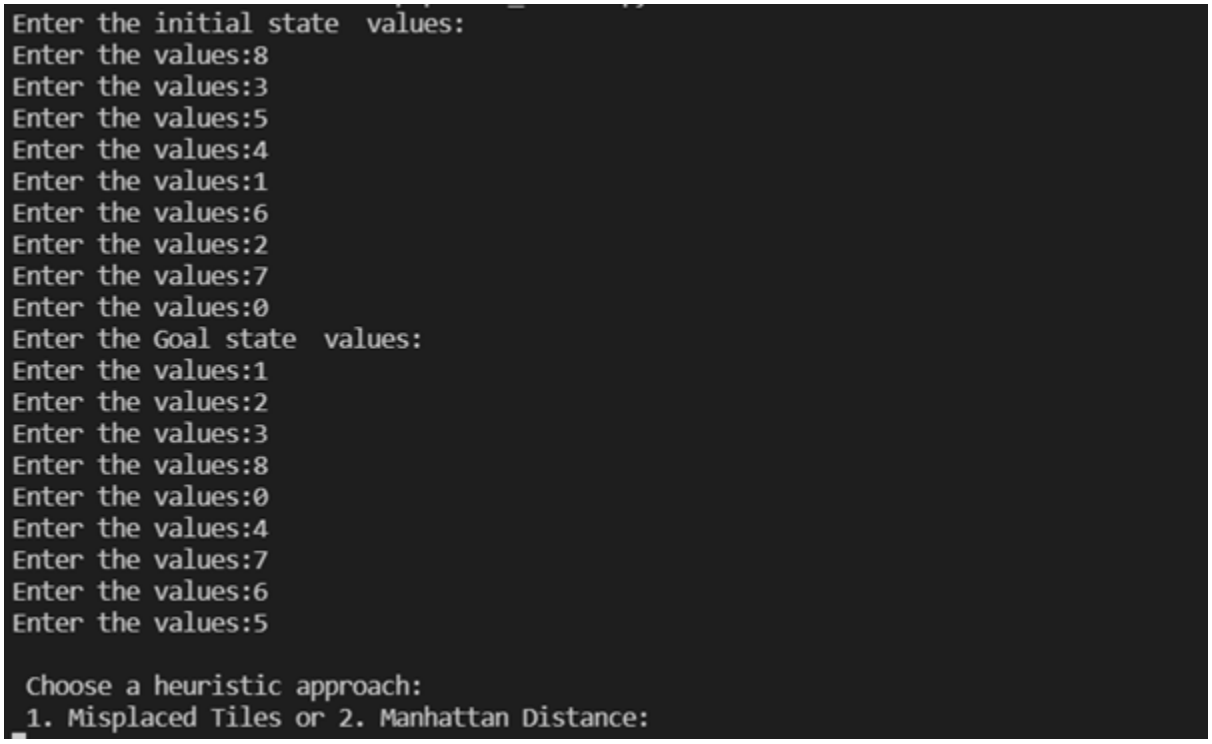
Manhattan Distance Output:



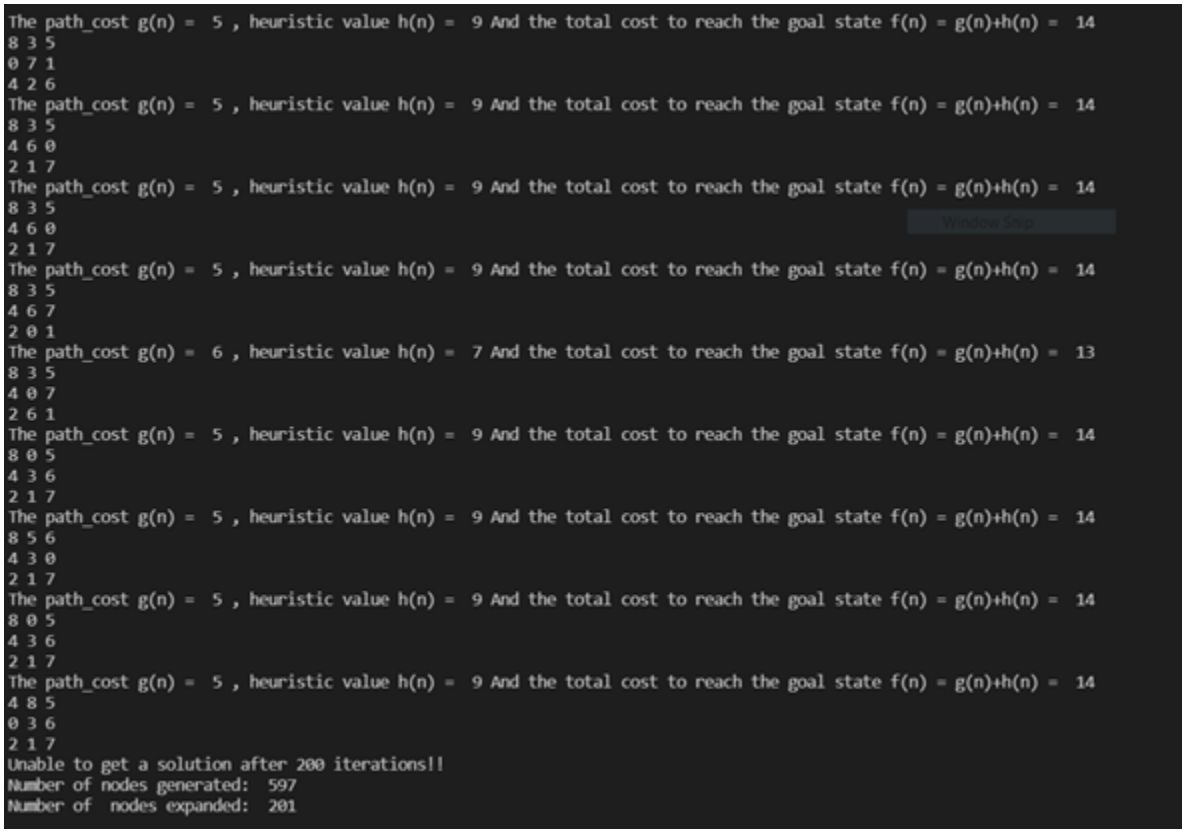
Input/Output 5:

Sample	Initial state	Goal State	Number of Nodes Generated (misplaced Tiles)	Number of nodes expanded (Misplaced Tiles)	Number of Nodes Generated (Manhattan Distance)	Number of Nodes expanded (Manhattan Distance)
Sample 5	[8 3 5] [4 1 6] [2 7 0]	[1 2 3] [8 0 4] [7 6 5]	597 (Unable to Find the solution)	201 (Unable to Find the Solution)	64 (Goal state Found)	21 (Goal state Found)

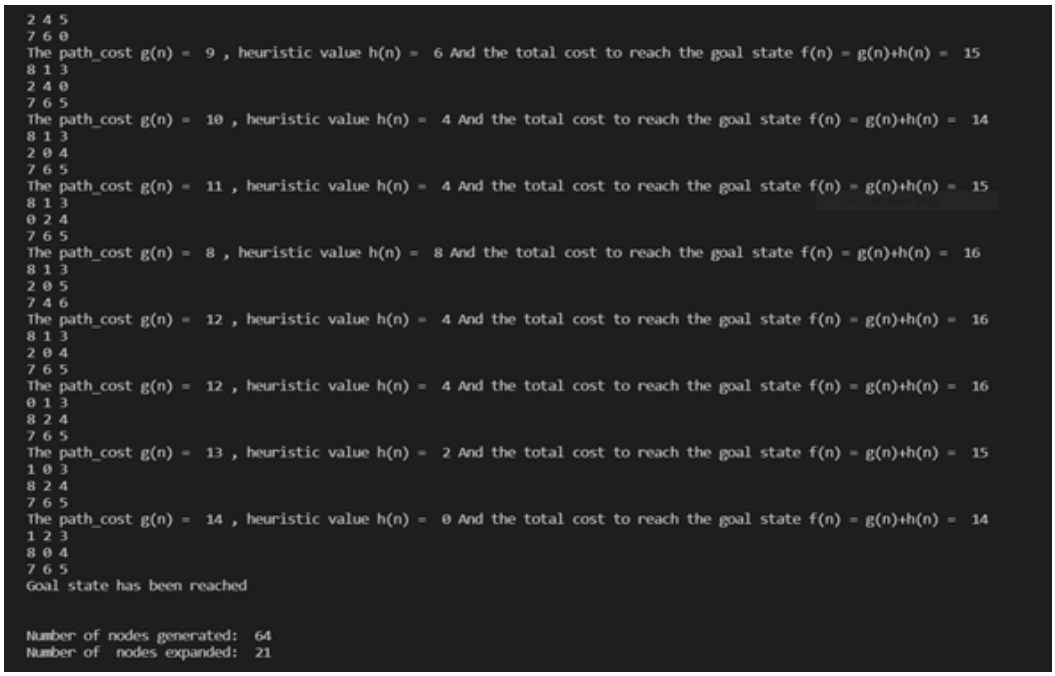
Getting Input:



Misplaced Titles Output:



Manhattan Distance Output:



Input/Output 6:

Sample	Initial State	Goal State	Number of Nodes Generated (misplaced Tiles)	Number of nodes expanded (Misplaced Tiles)	Number of Nodes Generated (Manhattan Distance)	Number of Nodes expanded (Manhattan Distance)
Sample 6	[1 2 0] [4 5 3] [7 8 6]	[1 2 3] [4 5 6] [7 8 0]	5	2	5	2

Getting Input:

```
$ python -u "c:\Users\Ram Nathan\Desktop\is_final\puzzle_solver.py"
Enter the initial state  values:
Enter the values:1
Enter the values:2
Enter the values:0
Enter the values:4
Enter the values:5
Enter the values:3
Enter the values:7
Enter the values:8
Enter the values:6
Enter the Goal state  values:
Enter the values:1
Enter the values:2
Enter the values:3
Enter the values:4
Enter the values:5
Enter the values:6
Enter the values:7
Enter the values:8
Enter the values:0
```

Misplaced Titles Output:

```
 Choose a heuristic approach:
 1. Misplaced Tiles or 2. Manhattan Distance:
1
The path_cost g(n) =  0 , heuristic value h(n) =  3 And the total cost to reach the goal
state f(n) = g(n)+h(n) =  3
1 2 0
4 5 3                                                                    state f(n) = g(n)+h(n) =  3
7 8 6
The path_cost g(n) =  1 , heuristic value h(n) =  2 And the total cost to reach the goal
state f(n) = g(n)+h(n) =  3
1 2 3                                                                    state f(n) = g(n)+h(n) =  3
4 5 0
7 8 6
The path_cost g(n) =  2 , heuristic value h(n) =  0 And the total cost to reach the goal
state f(n) = g(n)+h(n) =  2                                              state f(n) = g(n)+h(n) =  2
1 2 3
4 5 6
7 8 0
Goal state has been reached


Number of nodes generated:  5
Number of  nodes expanded:  2
```

Manhattan Distance Output:

```
 Choose a heuristic approach:
 1. Misplaced Tiles or 2. Manhattan Distance:
2
The path_cost g(n) =  0 , heuristic value h(n) =  4 And the total cost to reach the goal state f(n) = g(n)+h(n) =  4
1 2 0
4 5 3
7 8 6
The path_cost g(n) =  1 , heuristic value h(n) =  2 And the total cost to reach the goal state f(n) = g(n)+h(n) =  3
1 2 3
4 5 0
7 8 6
The path_cost g(n) =  2 , heuristic value h(n) =  0 And the total cost to reach the goal state f(n) = g(n)+h(n) =  2
1 2 3
4 5 6
7 8 0
Goal state has been reached


Number of nodes generated:  5
Number of  nodes expanded:  2
```

SUMMARY TABLE

Sample	Initial State	Goal State	Number of Nodes Generated (Misplaced titles)	Number of Nodes Expanded (Misplaced titles)	Number of Nodes Generated (Manhattan Distance)	Number of Nodes Expanded (Manhattan Distance)
Sample 1	[1 2 3] [7 4 5] [6 8 0]	[1 2 3] [8 6 4] [7 5 0]	137	48	54	20
Sample 2	[2 8 1] [3 4 6] [7 5 0]	[3 2 1] [8 0 4] [7 5 6]	28	9	17	6
Sample 3	[7 2 4] [5 0 6] [8 3 1]	[1 2 3] [4 5 6] [7 8 0]	578 (stopped at) Unable to Find solution	201 (stopped at) Unable to Find solution	538 (stopped at) Unable to Find solution	201 (stopped at) Unable to Solution
Sample 4	[2 8 3] [1 6 4] [7 0 5]	[1 2 3] [8 0 4] [7 6 5]	26	8	15	5
Sample 5	[8 3 5] [4 1 6] [2 7 0]	[1 2 3] [8 0 4] [7 6 5]	597(stopped at) Unable to Find the solution	201(stopped at) Unable to Find the solution	64 (Goal state is Found)	21 (Goal state is Found)
Sample 6	[1 2 0] [4 5 3] [7 8 6]	[1 2 3] [4 5 6] [7 8 0]	5	2	5	2

Conclusion:

The 8 puzzle algorithm is solved using A* algorithm which uses heuristics such as manhattan distance and misplaced tile.

References:

1.https://github.com

2.http://www.sfu.ca/~tjd/310summer2019/a1.html

3.8 puzzle problem using the A star search in C++ | Medium

4.https://www.linkedin.com/pulse/solving-8-puzzle-using-algorithm-python-ajinkya-sonawane