Project report
on

Solving 8-puzzle using A* algorithm

Project Guidance By

Dr. Dewan Ahmed

Team details

Sriganesh Lokesh
slokesh1@uncc.edu
801135650

Karthik Rangaraj
krangar2@uncc.edu
801135834

Problem Statement:

Given a n*n matrix having n^2 tiles out of which one tile is empty, we find our goal state by moving the tiles in a way that we reach the goal by swapping the '0'(missing) tile each step. The actions we perform to swap the tiles are up, left, right, down. Here we consider a 3*3 matrix having 9 tiles out of which one tile is empty.

The A* Search problem can be solved using to 2 approaches:

- **Misplaced Tiles**: This calculates the number of misplaced tiles in any given state.
- **Manhattan distance** : This calculates the sum of distances from the blocks in the current state to the respective goal state.

Initial State:

4	1	3
0	2	6
7	5	8

Goal State:

1	2	3
4	5	6
7	8	0

We solve the problem using A* Search algorithm. The A* Search is used to find the path from the initial to the goal state. Each move is considered as a new state. The A* algorithm uses the minimizing function F(n) = g(n) + h(n) where, g(n) = the cost to reach goal state, h(n) = the cost to goal from the nth state and f(n) = the total cost to goal from n.

At each step it picks the node having the lowest f(n) value, and proceeds to that state. If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, it returns a least-cost path from start to goal.

A* uses a fringe to perform the repeated selection of minimum cost nodes to expand. At each step of the algorithm, the node with the lowest f(x) value is removed from the queue, the f(n) and g(n) values of its neighbors are updated accordingly, and these neighbors are

added to the queue. The algorithm continues until a goal node has a lower f(n) value than any other node in the queue or until the queue is empty. The f(n) value of the goal is then the cost of the shortest path, since h(n) at the goal is zero in a heuristic.

Python Code:

```
import math
import copy

class A_Star_Search:
    def __init__(self, init_state=None):
        self.state = init_state
        self.h = 0
        self.g = 0
        self.f = 0
        self.parent = None
        self.action = None
    def next(self, state):
        steps = []
        for i in range(0, 3):
            for j in range(0, 3):
                if state[i][j] == 0:
                    row, col = i, j

        if row > 0:
            node = copy.deepcopy(state)
            row_new = row - 1
            node[row][col] = node[row_new][col]
            node[row_new][col] = 0
            steps.append((node, 'up'))
        if col > 0:
            node = copy.deepcopy(state)
            col_new = col - 1
            node[row][col] = node[row][col_new]
            node[row][col_new] = 0
            steps.append((node, 'left'))
        if row < 2:
            node = copy.deepcopy(state)
            row_new = row + 1
            node[row][col] = node[row_new][col]
            node[row_new][col] = 0
            steps.append((node, 'down'))
        if col < 2:
            node = copy.deepcopy(state)
```

```
            col_new = col + 1
            node[row][col] = node[row][col_new]
            node[row][col_new] = 0
            steps.append((node, 'right'))
        return steps

    def Path(self, node):
        actionsPerformed = []
        path = []
        path_cost = node.g
        while node:
            path.append(node.state)
            actionsPerformed.append(node.action)
            node = node.parent
        actionsPerformed.remove(None)
        print('Given Path: ')
        for node in reversed(path):
            printState(node)
        print('Operations Performed for the given Input: ')
        actionsPerformed = reversed(actionsPerformed)
        actionsSeq = ", ".join(actionsPerformed)
        print(actionsSeq)
        print('Path cost is: ', path_cost)

    def puzzle_solve(self, initial_state, goal_state, function='Manhattan'):
        generatedNodes_count = 0
        expandedNodes_count = 0
        fringe = []
        expandedNodes = []
        if initial_state.state == goal_state.state:
            print("Solution Found!")
            self.Path(initial_state)
            print("Generated Nodes Count: ", generatedNodes_count)
            print("Expanded Nodes Count: ", expandedNodes_count)
            return
        if function == 'misPlacedTiles':
            initial_state.h = misPlacedTiles(initial_state.state, goal_state.state)
        else:
            initial_state.h = manHattan(initial_state.state, goal_state.state)
        initial_state.f = initial_state.g + initial_state.h
        initial_state.parent = None
        initial_state.action = None
        fringe.append(initial_state)
        while fringe:
```

```
current = fringe.pop(0)
neighborNodes = self.next(current.state)
expandedNodes.append(current)
expandedNodes_count += 1
for neighbor in neighborNodes:
    childNode = A_Star_Search()
    childNode.state = neighbor[0]
    childNode.action = neighbor[1]
    childNode.g = current.g + 1
    if function == 'misPlacedTiles':
        childNode.h = misPlacedTiles(childNode.state, goal_state.state)
    else:
        childNode.h = manHattan(childNode.state, goal_state.state)
    childNode.f = childNode.g + childNode.h
    childNode.parent = current
    generatedNodes_count += 1
    if (childNode.state == goal_state.state):
        print("Solution Found.")
        self.Path(childNode)
        print("Generated Nodes Count: ", generatedNodes_count)
        print("Expanded Nodes Count: ", expandedNodes_count)
        return

    Expanded = False
    try:
        expandedNodes.index(childNode.state, )
    except ValueError:
        Expanded = False

    if not Expanded:
        found = False
        k = 0
        for item in fringe:
            if item.state == childNode.state:
                found = True
                if childNode.f < item.f:
                    item.f = childNode.f
                    fringe[k] = item
                    break
            k += 1

        if not found:
            fringe.append(childNode)
    fringe = sorted(fringe, key=lambda x: x.f)
```

```
        print('No Solution')
        return

def manHattan(state1, state2):
    array = []
    manhattanDist = 0
    for i in range(0, 3):
        for j in range(0, 3):
            array.append(state2[i][j])

    for i in range(0, 3, 1):
        for j in range(0, 3, 1):
            current_indexVals = state1[i][j]
            i_index = i
            j_index = j
            index = array.index(current_indexVals)
            goalI, goalJ = index // 3, index % 3
            if current_indexVals != 0:
                manhattanDist += (math.fabs(goalI - i_index) + math.fabs(goalJ - j_index))
    return manhattanDist

def misPlacedTiles(state1, state2):
    h = 0
    for i in range(0, 3, 1):
        for j in range(0, 3, 1):
            if state1[i][j] != state2[i][j] and state1[i][j] != 0:
                h += 1
    return h

def printState(state):
    for i in range(3):
        result = ""
        for j in range(3):
            result += str(state[i][j]) + " "
        print(result)
    print("")

def userInput():
    print("Enter Initial State: ")
    inputs = []
    goal = []
    items = input().split(" ")
    k = 0
    try:
```

```
        for i in range(0, 3):
            inputs += [0]
        for i in range(0, 3):
            inputs[i] = [0] * 3
        for i in range(0, 3):
            for j in range(0, 3):
                inputs[i][j] = int(items[k])
                k += 1
    except (ValueError, IndexError):
        print("Enter Input with Space Seperation")
        return [], []
    print("Enter Goal State: ")
    items = input().split(" ")
    k = 0
    try:
        for i in range(0, 3):
            goal += [0]
        for i in range(0, 3):
            goal[i] = [0] * 3
        for i in range(0, 3):
            for j in range(0, 3):
                goal[i][j] = int(items[k])
                k += 1
    except (ValueError, IndexError):
        print("Enter Input with Space Seperation")
        return inputs, []
    return inputs, goal

def main():
    inputArray, goalArray = userInput()
    if inputArray and goalArray:
        print('Initial State: ')
        printState(inputArray)
        print('Goal state: ')
        printState(goalArray)
        initial = A_Star_Search(inputArray)
        goal = A_Star_Search(goalArray)
        print("A star Search for the 8 puzzle problem using Manhattan Distance is :")
        initial.puzzle_solve(initial, goal)
        print("\nA star search for the 8 puzzle problem using MisplacedTiles is :")
        initial.puzzle_solve(initial, goal, 'misPlacedTiles')
```

Program structure:

The algorithm has been designed in Python.

Global Variable:

Two variables expanded and fringe (priority queue) are used in solve, which is our function to solve A* algorithm.

Main class: A_Star_Search

Functions & Procedures:

next(): Function to generate next states from the current node.
Path(): Function used to print the path from the initial state to the goal state with state cost.
puzzle_solve() : Function to solve using A* algorithm.
manhattan() : Function to calculate Manhattan Distance.
misPlacedTiles() : Function to calculate Misplaced tiles.
printState() : Function to print the state in a 3x3 matrix.
userInput() : Function to accept initial and goal states from the user.

Detailed Output:

1.

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

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

A star Search for the 8 puzzle problem using Manhattan Distance is :
Solution Found.
Given Path:
1 2 3
7 4 5
6 8 0

1 2 3
7 4 0
6 8 5

1 2 3
7 0 4
6 8 5

1 2 3
7 8 4
6 0 5

1 2 3
7 8 4
0 6 5

1 2 3
0 8 4
7 6 5

1 2 3
8 0 4
7 6 5

1 2 3
8 6 4
7 0 5

1 2 3
8 6 4
7 5 0

Operations Performed for the given Input:
up, left, down, left, up, right, down, right
Path cost is: 8
Generated Nodes Count: 26
Expanded Nodes Count: 9

A star search for the 8 puzzle problem using MisplacedTiles is :
Solution Found.
Given Path:
1 2 3
7 4 5
6 8 0

1 2 3
7 4 0
6 8 5

1 2 3

7 0 4
6 8 5

1 2 3
7 8 4
6 0 5

1 2 3
7 8 4
0 6 5

1 2 3
0 8 4
7 6 5

1 2 3
8 0 4
7 6 5

1 2 3
8 6 4
7 0 5

1 2 3
8 6 4
7 5 0

Operations Performed for the given Input:
up, left, down, left, up, right, down, right
Path cost is: 8
Generated Nodes Count: 73
Expanded Nodes Count: 25

2.

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

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

A star Search for the 8 puzzle problem using Manhattan Distance is :

Solution Found.
Given Path:
2 8 1
3 4 6
7 5 0

2 8 1
3 4 0
7 5 6

2 8 1
3 0 4
7 5 6

2 0 1
3 8 4
7 5 6

0 2 1
3 8 4
7 5 6

3 2 1
0 8 4
7 5 6

3 2 1
8 0 4
7 5 6

Operations Performed for the given Input:
up, left, up, left, down, right
Path cost is: 6
Generated Nodes Count: 17
Expanded Nodes Count: 6

A star search for the 8 puzzle problem using MisplacedTiles is :
Solution Found.
Given Path:
2 8 1
3 4 6
7 5 0

2 8 1
3 4 0
7 5 6

2 8 1
3 0 4
7 5 6

2 0 1
3 8 4
7 5 6

0 2 1
3 8 4
7 5 6

3 2 1
0 8 4
7 5 6

3 2 1
8 0 4
7 5 6

Operations Performed for the given Input:
up, left, up, left, down, right
Path cost is: 6
Generated Nodes Count: 20
Expanded Nodes Count: 7

3.

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

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

A star Search for the 8 puzzle problem using Manhattan Distance is :
Solution Found.
Given Path:
7 2 4
5 0 6
8 3 1

7 2 4
5 3 6
8 0 1

7 2 4

5 3 6
8 1 0

7 2 4
5 3 0
8 1 6

7 2 4
5 0 3
8 1 6

7 2 4
0 5 3
8 1 6

0 2 4
7 5 3
8 1 6

2 0 4
7 5 3
8 1 6

2 4 0
7 5 3
8 1 6

2 4 3
7 5 0
8 1 6

2 4 3
7 0 5
8 1 6

2 4 3
7 1 5
8 0 6

2 4 3
7 1 5
0 8 6

2 4 3
0 1 5
7 8 6

2 4 3
1 0 5
7 8 6

2 0 3
1 4 5
7 8 6

0 2 3
1 4 5
7 8 6

1 2 3
0 4 5
7 8 6

1 2 3
4 0 5
7 8 6

1 2 3
4 5 0
7 8 6

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
down, right, up, left, left, up, right, right, down, left, down, left, up, right, up, left, down, right, right, down
Path cost is: 20
Generated Nodes Count: 1039
Expanded Nodes Count: 393

A star search for the 8 puzzle problem using MisplacedTiles is :
Solution Found.
Given Path:
7 2 4
5 0 6
8 3 1

7 2 4
5 3 6
8 0 1

7 2 4
5 3 6
8 1 0

7 2 4
5 3 0
8 1 6

7 2 4

5 0 3
8 1 6

7 2 4
0 5 3
8 1 6

0 2 4
7 5 3
8 1 6

2 0 4
7 5 3
8 1 6

2 4 0
7 5 3
8 1 6

2 4 3
7 5 0
8 1 6

2 4 3
7 0 5
8 1 6

2 4 3
7 1 5
8 0 6

2 4 3
7 1 5
0 8 6

2 4 3
0 1 5
7 8 6

2 4 3
1 0 5
7 8 6

2 0 3
1 4 5
7 8 6

0 2 3
1 4 5
7 8 6

1 2 3
0 4 5
7 8 6

1 2 3
4 0 5
7 8 6

1 2 3
4 5 0
7 8 6

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
down, right, up, left, left, up, right, right, down, left, down, left, up, right, up, left, down, right, right, down
Path cost is: 20
Generated Nodes Count: 15995
Expanded Nodes Count: 5918

4.

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

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

A star Search for the 8 puzzle problem using Manhattan Distance is :
Solution Found.
Given Path:
1 2 3
4 6 5
8 7 0

1 2 3
4 6 5
8 0 7

1 2 3
4 0 5
8 6 7

1 2 3
4 5 0
8 6 7

1 2 3
4 5 7
8 6 0

1 2 3
4 5 7
8 0 6

1 2 3
4 5 7
0 8 6

1 2 3
0 5 7
4 8 6

1 2 3
5 0 7
4 8 6

1 2 3
5 7 0
4 8 6

1 2 3
5 7 6
4 8 0

1 2 3
5 7 6
4 0 8

1 2 3
5 0 6
4 7 8

1 2 3
0 5 6
4 7 8

1 2 3
4 5 6
0 7 8

1 2 3
4 5 6
7 0 8

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
left, up, right, down, left, left, up, right, right, down, left, up, left, down, right, right
Path cost is: 16
Generated Nodes Count: 1337
Expanded Nodes Count: 496

A star search for the 8 puzzle problem using MisplacedTiles is :
Solution Found.
Given Path:
1 2 3
4 6 5
8 7 0

1 2 3
4 6 5
8 0 7

1 2 3
4 6 5
0 8 7

1 2 3
0 6 5
4 8 7

1 2 3
6 0 5
4 8 7

1 2 3
6 8 5
4 0 7

1 2 3
6 8 5
4 7 0

1 2 3
6 8 0
4 7 5

1 2 3

6 0 8
4 7 5

1 2 3
0 6 8
4 7 5

1 2 3
4 6 8
0 7 5

1 2 3
4 6 8
7 0 5

1 2 3
4 6 8
7 5 0

1 2 3
4 6 0
7 5 8

1 2 3
4 0 6
7 5 8

1 2 3
4 5 6
7 0 8

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
left, left, up, right, down, right, up, left, left, down, right, right, up, left, down, right
Path cost is: 16
Generated Nodes Count: 3088
Expanded Nodes Count: 1129

5.

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

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

A star Search for the 8 puzzle problem using Manhattan Distance is :
Solution Found.
Given Path:
0 1 2
4 5 3
7 8 6

1 0 2
4 5 3
7 8 6

1 2 0
4 5 3
7 8 6

1 2 3
4 5 0
7 8 6

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
right, right, down, down
Path cost is: 4
Generated Nodes Count: 10
Expanded Nodes Count: 4

A star search for the 8 puzzle problem using MisplacedTiles is :
Solution Found.
Given Path:
0 1 2
4 5 3
7 8 6

1 0 2
4 5 3
7 8 6

1 2 0
4 5 3
7 8 6

1 2 3

4 5 0
7 8 6

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
right, right, down, down
Path cost is: 4
Generated Nodes Count: 10
Expanded Nodes Count: 4

6.

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

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

A star Search for the 8 puzzle problem using Manhattan Distance is :
Solution Found.
Given Path:
0 5 2
4 8 3
7 1 6

5 0 2
4 8 3
7 1 6

5 8 2
4 0 3
7 1 6

5 8 2
4 1 3
7 0 6

5 8 2
4 1 3
0 7 6

5 8 2
0 1 3
4 7 6

5 8 2
1 0 3
4 7 6

5 0 2
1 8 3
4 7 6

0 5 2
1 8 3
4 7 6

1 5 2
0 8 3
4 7 6

1 5 2
4 8 3
0 7 6

1 5 2
4 8 3
7 0 6

1 5 2
4 0 3
7 8 6

1 0 2
4 5 3
7 8 6

1 2 0
4 5 3
7 8 6

1 2 3
4 5 0
7 8 6

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
right, down, down, left, up, right, up, left, down, down, right, up, up, right, down, down

Path cost is: 16
Generated Nodes Count: 716
Expanded Nodes Count: 268

A star search for the 8 puzzle problem using MisplacedTiles is :
Solution Found.
Given Path:
0 5 2
4 8 3
7 1 6

5 0 2
4 8 3
7 1 6

5 8 2
4 0 3
7 1 6

5 8 2
4 1 3
7 0 6

5 8 2
4 1 3
0 7 6

5 8 2
0 1 3
4 7 6

5 8 2
1 0 3
4 7 6

5 0 2
1 8 3
4 7 6

0 5 2
1 8 3
4 7 6

1 5 2
0 8 3
4 7 6

1 5 2
4 8 3
0 7 6

1 5 2
4 8 3
7 0 6

1 5 2
4 0 3
7 8 6

1 0 2
4 5 3
7 8 6

1 2 0
4 5 3
7 8 6

1 2 3
4 5 0
7 8 6

1 2 3
4 5 6
7 8 0

Operations Performed for the given Input:
right, down, down, left, up, right, up, left, down, down, right, up, up, right, down, down
Path cost is: 16
Generated Nodes Count: 2703
Expanded Nodes Count: 994

Summarized Results:

Initial State	Goal State	Results
1 2 3 7 4 5 6 8 0	1 2 3 8 6 4 7 5 0	1) Using Manhattan: Nodes Generated: 26 Nodes Expanded: 9 Path Cost: 8 Using No of Misplaced Tiles: Nodes Generated: 73 Nodes Expanded: 25 Path Cost: 8

2 8 1 3 4 6 7 5 0	3 2 1 8 0 4 7 5 6	2) Using Manhattan: Nodes Generated: 17 Nodes Expanded: 6 Path Cost: 6 Using No of Misplaced Tiles: Nodes Generated: 20 Nodes Expanded: 7 Path Cost: 6
7 2 4 5 0 6 8 3 1	1 2 3 4 5 6 7 8 0	3) Using Manhattan: Nodes Generated: 1039 Nodes Expanded: 393 Path Cost: 20 Using No of Misplaced Tiles: Nodes Generated: 15995 Nodes Expanded: 5918 Path Cost: 20
1 2 3 4 6 5 8 7 0	1 2 3 4 5 6 7 8 0	4) Using Manhattan: Nodes Generated: 1337 Nodes Expanded: 496 Path Cost: 16 Using No of Misplaced Tiles: Nodes Generated: 3088 Nodes Expanded: 1129 Path Cost: 16

0 1 2 4 5 3 7 8 6	1 2 3 4 5 6 7 8 0	5) Using Manhattan: Nodes Generated: 10 Nodes Expanded: 4 Path Cost: 4 Using No of Misplaced Tiles: Nodes Generated: 10 Nodes Expanded: 4 Path Cost: 4
0 5 2 4 8 3 7 1 6	1 2 3 4 5 6 7 8 0	6) Using Manhattan: Nodes Generated: 716 Nodes Expanded: 268 Path Cost: 16 Using No of Misplaced Tiles: Nodes Generated: 2703 Nodes Expanded: 994 Path Cost: 16

References:

- Artificial Intelligence A Modern Approach, Stuart Russell and Peter Norvig
- Wikipedia
- www.stackoverflow.com
- www.geeksforgeeks.com