

# CS 205 - Artificial Intelligence

Dr. Eamonn Keogh

*Project -1 : 8 - Puzzle problem*

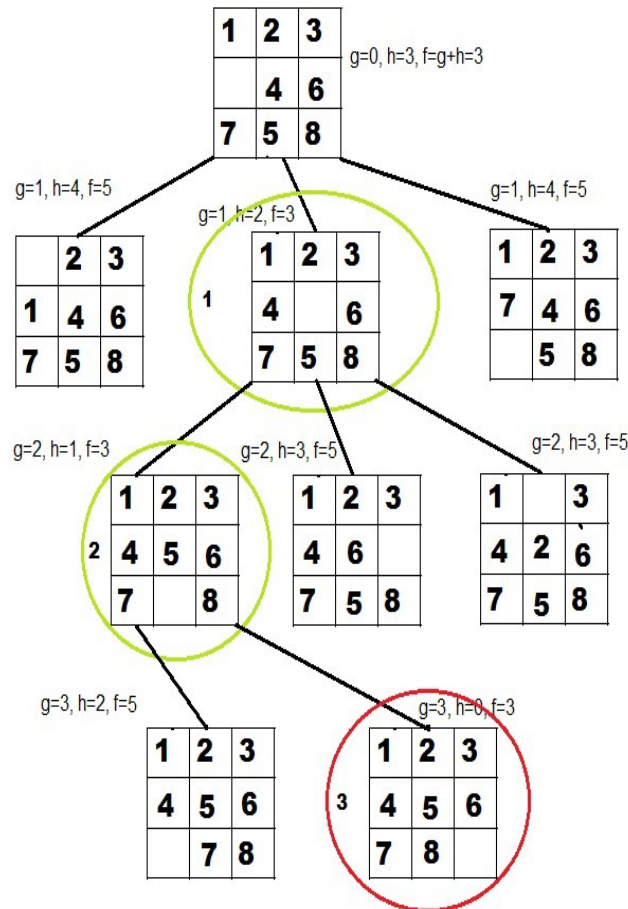


Figure 1: An Example of 8-puzzle Problem

**HANISHA SREE SANGAM**

05/12/2021.

SPRING 2021

## TABLE OF CONTENTS

| S.No | Content                  | Page Number |
|------|--------------------------|-------------|
| 1    | INTRODUCTION             | 2           |
| 2    | COMPARISON OF ALGORITHMS | 3           |
| 3    | CONCLUSION               | 4           |
| 4    | REFERENCES               | 4           |
| 5    | WORKING OF THE ALGORITHM | 5           |
| 6    | EXAMPLES                 | 6           |

## INTRODUCTION

Solving The 8-Puzzle Problem is the Project in CS 205- Artificial Intelligence course, under the Professor - Dr. Eamonn Keogh at University of California, Riverside for the Spring 2021 quarter. This Report is a detailed explanatory of my work and findings.

Problem Statement : Solving 8-Puzzle problem using Uniform Cost Search, A\* with Misplaced Tile and A\* with Manhattan Distance Heuristic.

8 - Puzzle Problem : It is a problem that solves a 3 x 3 grid with 8 numbered tiles and one blank space tile. 15 - Puzzle is a bigger version of 8 - Puzzle Problem. The Aim of this project is to reach the goal state (pre -defined state i.e the numerical order) by moving each tile up, down, left and right.

Uniform Cost Search : It is a type of Dijkstra's Algorithm that doesn't involve any cost calculation. It is equivalent to sliding each tile randomly with fingers. It is a special case of A\* search algorithm.

A\* search : A Star is an informed search algorithm, which is calculated using weighted terms. From a start node to goal node it is used to find a path with lowest cost.

A\* search with Misplaced tiles Heuristic : As per the name Misplaced Tile, Algorithm compares the input puzzle with the goal state, checks with the numbered tile that is wrongly placed. Each misplaced tile has to move at least once. This always results in an optimal solution as it never over estimates the cost of reaching the goal state.

A\* search with Manhattan Distance Heuristic : This algorithm uses distance to measure the cost to reach the goal state. The heuristic considers all of the misplaced tiles and the number of tiles away from its goal state.

Total cost is the sum of all the cost of all misplaced tile distances.

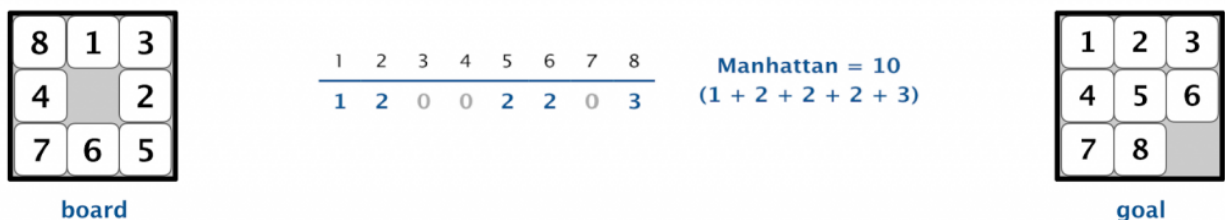


Figure 2 : Example of Manhattan Distance calculation

### Comparison of Algorithms on Sample Cases :

|  | DEPTH | UNIFORM COST SEARCH | A STAR WITH MISPLACED TILE | A STAR WITH MANHATTAN DISTANCE |   |   |   |   |   |   |                                     |                                      |                                      |
|--|-------|---------------------|----------------------------|--------------------------------|---|---|---|---|---|---|-------------------------------------|--------------------------------------|--------------------------------------|
| <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>0</td><td>7</td><td>8</td></tr></table> | 1     | 2                   | 3                          | 4                              | 5 | 6 | 0 | 7 | 8 | 2 | Cost - 9<br>Time taken - 33 seconds | Cost - 11<br>Time taken - 24 seconds | Cost - 10<br>Time taken - 12 seconds |
| 1  | 2     | 3                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| 4  | 5     | 6                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| 0  | 7     | 8                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>5</td><td>0</td><td>6</td></tr><tr><td>4</td><td>7</td><td>8</td></tr></table> | 1     | 2                   | 3                          | 5                              | 0 | 6 | 4 | 7 | 8 | 4 | Cost - 9<br>Time taken - 31 seconds | Cost - 11<br>Time taken - 17 seconds | Cost - 10<br>Time taken - 11 seconds |
| 1  | 2     | 3                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| 5  | 0     | 6                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| 4  | 7     | 8                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| <table><tr><td>1</td><td>3</td><td>6</td></tr><tr><td>5</td><td>0</td><td>2</td></tr><tr><td>4</td><td>7</td><td>8</td></tr></table> | 1     | 3                   | 6                          | 5                              | 0 | 2 | 4 | 7 | 8 | 8 | Cost - 9<br>Time taken - 19 seconds | Cost - 11<br>Time taken - 19 seconds | Cost - 10<br>Time taken - 15 seconds |
| 1  | 3     | 6                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| 5  | 0     | 2                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |
| 4  | 7     | 8                   |                            |                                |   |   |   |   |   |   |                                     |                                      |                                      |

|  |   |   |   |   |   |   |   |   |   |    |  |   |   |
|--|---|---|---|---|---|---|---|---|---|----|--|---|---|
| <table><tr><td>7</td><td>1</td><td>2</td></tr><tr><td>4</td><td>8</td><td>5</td></tr><tr><td>6</td><td>3</td><td>0</td></tr></table> | 7 | 1 | 2 | 4 | 8 | 5 | 6 | 3 | 0 | 20 | Cost - 9<br>Time taken - 18<br>seconds | Cost - 11<br>Time taken - 13<br>seconds | Cost - 10<br>Time taken - 12<br>seconds |
| 7  | 1 | 2 |   |   |   |   |   |   |   |    |  |   |   |
| 4  | 8 | 5 |   |   |   |   |   |   |   |    |  |   |   |
| 6  | 3 | 0 |   |   |   |   |   |   |   |    |  |   |   |
| <table><tr><td>0</td><td>7</td><td>2</td></tr><tr><td>4</td><td>6</td><td>1</td></tr><tr><td>3</td><td>5</td><td>8</td></tr></table> | 0 | 7 | 2 | 4 | 6 | 1 | 3 | 5 | 8 | 24 | Cost - 9<br>Time taken - 27<br>seconds | Cost - 11<br>Time taken - 15<br>seconds | Cost - 10<br>Time taken - 10<br>seconds |
| 0  | 7 | 2 |   |   |   |   |   |   |   |    |  |   |   |
| 4  | 6 | 1 |   |   |   |   |   |   |   |    |  |   |   |
| 3  | 5 | 8 |   |   |   |   |   |   |   |    |  |   |   |

## CONCLUSION

- 1) Always Uniform cost search takes more time to reach the goal state.
- 2) A star with Manhattan distance takes the least possible time to reach the goal state.
- 3) Uniform cost search takes the least possible cost needed to find the goal state.
- 4) When we compare A star with misplaced tile and A star with Manhattan distance, later is the more efficient to use.

## REFERENCES

1. Google (Images)
2. CS 205 Lecture slides
3. Wikipedia
4. Geeks for geeks
5. Artificial Intelligence, Modern Approach Textbook.

## WORKING OF THE ALGORITHM :

Below is the precise step-wise explanation of my code :

- 1) Class input.java is a gui class that takes input from user through textfields, compares it with the final state (if the given input is already the solution it prints on the screen and exits the program otherwise, it creates an object of class 'heuristic.java' which asks user to choose a heuristic.
- 2) Once a heuristic is given, based on the selection, an object is created.
- 3) For Misplaced Tiles , an object of A-star.java is created and then function aStarSearchWithMisplacedTiles is called to compute the solution.
- 4) For Manhattan Distance, an object of A-Star.java is created and then function aStarSearchWithManhattanDistance is called to compute the solution.
- 5) For Unit cost search, an object of class unitSearch.java is created and it's method search is called with that object.
- 6) state.java is an abstract class that defines the structure of the puzzle state.
- 7) puzzle inherits state class and implements the definition and operations of the puzzle states.
- 8) Node.java is a class that can be considered as something that creates the graph for the solution. it's a node (in a graph, list, linkedlist, array, etc.).
- 9) EightPuzzle.java is a main driver class which creates an object on class input.java, sets the object visible(which brings the input screen to life).

## CODE :

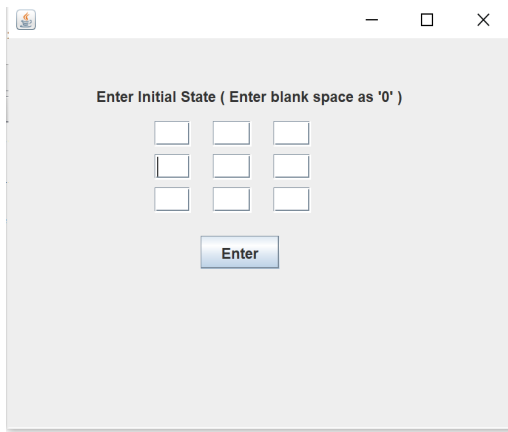
<https://github.com/Hanisha3/CS205Projects/tree/master/eightpuzzle>

The above is the Github link to the code of my project.

## Examples :

Given below are the example of an Easy and Hard Problem :

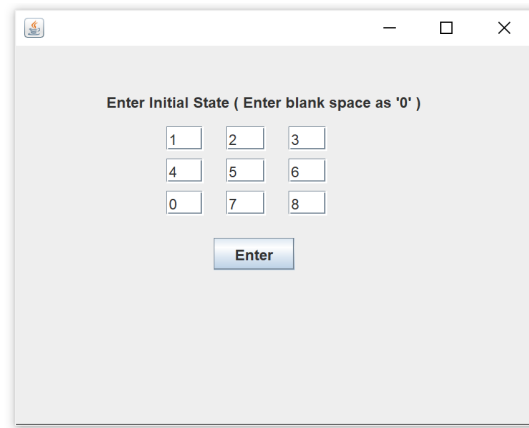
- 1) **Easy Problem** : Below are the screenshots of working on 8 puzzle problems of depth 2.



Enter Initial State ( Enter blank space as '0' )

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |

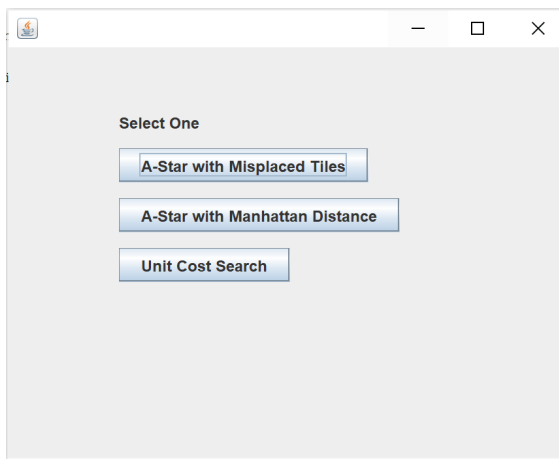
Enter



Enter Initial State ( Enter blank space as '0' )

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 0 | 7 | 8 |

Enter



Select One

A-Star with Misplaced Tiles

A-Star with Manhattan Distance

Unit Cost Search

## Output:

### 1) When we choose option - Uniform Cost Search

```
Updating property file: C:\Users\hanis\Documents\NetBeansProjects\EightPuzzle\build\built-jar.properties
compile:
run:
Moved down
Moved right
Moved left
Moved down
Moved right
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

The total cost was: 11
BUILD SUCCESSFUL (total time: 2 minutes 40 seconds)
```

### 2) When we choose the heuristic - A\* with Manhattan Distance

```
Updating property file: C:\Users\hanis\Documents\NetBeansProjects\EightPuzzle\build\built-jar.properties
compile:
run:
Moved down
Moved right
Moved left
Moved down
Moved right
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

The total cost was: 10
BUILD SUCCESSFUL (total time: 13 seconds)
```



### 3) When we choose the heuristic - A\* with Manhattan Distance

---

```
ant -f C:\Users\hanis\Documents\NetBeansProjects\EightPuzzle -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\hanis\Documents\NetBeansProjects\EightPuzzle\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\hanis\Documents\NetBeansProjects\EightPuzzle\build\build-jar.properties
compile:
run:
Moved down
Moved right
Moved up
Moved down
Moved right
Moved left
Moved down
Moved right
Moved up
Moved up
Moved right
Moved left
Moved up
Moved down
Moved right
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 | 0 | 8

1 | 2 | 3
-----
4 | 5 | 6
-----
7 | 8 | 0

The total cost was: 9
BUILD SUCCESSFUL (total time: 10 seconds)
```

## 2) Hard Problem :

Enter Initial State ( Enter blank space as '0' )

|   |   |   |
|---|---|---|
| 0 | 7 | 2 |
| 4 | 6 | 1 |
| 3 | 5 | 8 |

Enter

Select One

A-Star with Misplaced Tiles

A-Star with Manhattan Distance

Unit Cost Search

Output :

Updating property file:

C:\Users\hanis\Documents\NetBeansProjects\EightPuzzle\build\built-jar.properties

compile:

run:

Moved up

Moved right

Moved left

Moved up

Moved right

.....

.....

// Omitted upto 11 pages of Output.

1 | 2 | 3

-----

4 | 5 | 6

-----

7 | 8 | 0

The total cost was: 9

BUILD SUCCESSFUL (total time: 1 minute 47 seconds)

When we choose the heuristic - A\* with Manhattan Distance

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

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

The total cost was: 10

BUILD SUCCESSFUL (total time: 29 seconds)