REPORT

Project 1: A* Search Algorithm (8-puzzle problem)

Submitted By

Jawad Chowdhury, (ID# 801135477)

8-puzzle Problem Formulation

An 8-puzzle problem is a specific version of n-puzzle solving problem. Here, in this implementation, we are going to use A* search algorithm to solve the problem.

In a general sense, the problem gives us an initial state (a random formation of the tiles) and exploring the possible search paths with a strategy we are required to reach the goal state.

We can associate a path cost f(n) to reach the goal state where f(n) = g(n) + h(n).

Here, g(n) is the cost to reach the state n, and h(n) is the heuristic cost (an estimation) to reach the goal state from n.

In A* search, we follow the strategy in such a way so that f(n) becomes optimal.

Program Structure (Variables, Function & Procedures)

In my implementation, I have basically used 2 classes. One is to maintain the basic flow of the problem solving which is being denoted as '**Puzzle**' and another one is the '**Node**' class which is kind of the unit block representing each state and its associating information.

Node: The class Node is comparatively the simple one. It consists of some instance variables such as state of the node, depth of the node and the f score f(n).

Puzzle: This class is the compound one which keeps the info regarding the number of tiles, the frontier list and the list of explored nodes.

This class also has some helper functions to calculate the g score, h score, f score of the nodes, function to get the location (index) of the blank tile in a state, function for node expansion to generate the children, a goal testing function.

The Puzzle class contains a **run()** function which is the basic procedure to run the A* searching algorithm on the problem.

Analysis of 6 input/output cases (with 2 heuristics)

Case 1

(Initial State)

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

(Goal State)

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

(Path, No of Nodes Generated & No of Nodes Expanded)

```
Using heuristic 1: (Miss-placed Tiles)
Goal State Reached!!!
===   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

No of Nodes Generated : 118
No of Nodes Expanded : 39
=====================

Using heuristic 2: (Manhattan Distance)
Goal State Reached!!!
===   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

No of Nodes Generated : 27
No of Nodes Expanded : 10
```

Case 2

(Initial State)

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

(Goal State)

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

(Path, No of Nodes Generated & No of Nodes Expanded)

```
Using heuristic 1: (Miss-placed Tiles)
Goal State Reached!!!
===   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

No of Nodes Generated : 21
No of Nodes Expanded : 8
=====================

Using heuristic 2: (Manhattan Distance)
Goal State Reached!!!
===   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

No of Nodes Generated : 18
No of Nodes Expanded : 7
```

Case 3

(Initial State)

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

(Goal State)

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

(Path, No of Nodes Generated & No of Nodes Expanded)

```
Using heuristic 1: (Miss-placed Tiles)
Goal State Reached!!!
===   PATH   ===
1 2 3
4 5 6
7 8 0

No of Nodes Generated : 1
No of Nodes Expanded : 1
====================

Using heuristic 2: (Manhattan Distance)
Goal State Reached!!!
===   PATH   ===
1 2 3
4 5 6
7 8 0

No of Nodes Generated : 1
No of Nodes Expanded : 1
```

Case 4

(Initial State)

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

(Goal State)

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

(Path, No of Nodes Generated & No of Nodes Expanded)

```
Using heuristic 1: (Miss-placed Tiles)
Goal State Reached!!!
===   PATH   ===
2 8 1
```

```
3 4 6
7 5 0

2 8 1
3 4 6
7 0 5

2 8 1
3 0 6
7 4 5

2 0 1
3 8 6
7 4 5

2 1 0
3 8 6
7 4 5

2 1 6
3 8 0
7 4 5

No of Nodes Generated : 15
No of Nodes Expanded : 6
=====================

Using heuristic 2: (Manhattan Distance)
Goal State Reached!!!
===   PATH   ===
2 8 1
3 4 6
7 5 0

2 8 1
3 4 6
7 0 5

2 8 1
3 0 6
```

```
7 4 5

2 0 1
3 8 6
7 4 5

2 1 0
3 8 6
7 4 5

2 1 6
3 8 0
7 4 5

No of Nodes Generated : 15
No of Nodes Expanded : 6
```

Case 5

(Initial State)

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

(Goal State)

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

(Path, No of Nodes Generated & No of Nodes Expanded)

```
Using heuristic 1: (Miss-placed Tiles)
Goal State Reached!!!
===   PATH   ===
4 1 3
2 5 6
7 8 0

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

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

4 1 3
0 2 5
7 8 6

0 1 3
4 2 5
7 8 6

1 0 3
4 2 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

No of Nodes Generated : 112
No of Nodes Expanded : 42
=====================

Using heuristic 2: (Manhattan Distance)
Goal State Reached!!!
===   PATH   ===
4 1 3
2 5 6
7 8 0
```

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

4 1 3
2 0 5
7 8 6

4 1 3
0 2 5
7 8 6

0 1 3
4 2 5
7 8 6

1 0 3
4 2 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

No of Nodes Generated : 95
No of Nodes Expanded : 36
```

Case 6

(Initial State)

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

(Goal State)

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

(Path, No of Nodes Generated & No of Nodes Expanded)

```
Using heuristic 1: (Miss-placed Tiles)
Goal State Reached!!!
===   PATH   ===
1 2 3
4 5 6
7 8 0

1 2 3
4 5 0
7 8 6

1 2 3
4 0 5
7 8 6

1 2 3
4 8 5
7 0 6

1 2 3
4 8 5
7 6 0

No of Nodes Generated : 16
No of Nodes Expanded : 6
=====================

Using heuristic 2: (Manhattan Distance)
```

```
Goal State Reached!!!
===   PATH   ===
1 2 3
4 5 6
7 8 0

1 2 3
4 5 0
7 8 6

1 2 3
4 0 5
7 8 6

1 2 3
4 8 5
7 0 6

1 2 3
4 8 5
7 6 0

No of Nodes Generated : 13
No of Nodes Expanded : 5
=====================
```