Problem Formulation:

An eight puzzle problem can be best described as a 3 by 3 grid with 3 to the power of 2 , minus one square tiles , i.e. 8 with a blank space. Here the 8 square tiles are labelled from 1 to 8. Here we are given with an initial state and a goal state, we have to slide the tiles one at a time and reach the given goal state. The possible legal moves a player can make are sliding a given tile horizontally or vertically into the blank space. So the possible moves being UP, LEFT, DOWN, RIGHT.

1	2	3
	4	5
7	8	6

Initial State

1	2	3
4		5
7	8	6

1	2	3
4	5	
7	8	6

1	2	3
4	5	6
7	8	

Goal State

Heuristics: There are two heuristics we use in solving the puzzle problem

Misplaced Tiles: In this heuristic the h(n) value is number of tiles that are misplaced when compared to the goal state.

Manhattan Distance: In this heuristic the h(n) value is the sum of the distances of each misplaced tile from its goal state

Program Structure:

First import the required Packages

1.import numpy

2.import copy

3.import itemgetter from operator

Define required Class definitions

We have defined a class named node

Define required Function Definitions

Functions:

1.solution(length,i,j,state):

a.Input Parameters:

Length: It takes the size of the puzzle

i , j: row and column position of the blank space in the puzzle

state: The state of the node that is to be expanded.

Returns the possible solutions.

Local Variables:

1.s1,s2,s3,s4 of type numpy arrays

2.temp1,temp2,temp3,temp4 are temporary variables used for swapping

2.misplaced_tiles(state,goal_state): Calculates the Misplaced tiles heuristic value of a given node element

Input parameters:

State: Current state of the puzzle is given as input

Goal_state: Calculates the Manhattan heuristic value of a given node element

Returns the heuristic value of the current state

Local Variables:

1.length,h of type int

3. manhattan_heuristic(state,goal_state):

Input Parameters:

State: Current state of the puzzle is given as input

Goal_state: Calculates the Manhattan heuristic value of a given node element

Returns the heuristic value of the current state

Local Variables:

1.length,h,g1,g2,f of type int

2.g of type tuple

4. possible_solution(State,Goal_state:):

Input parameters:

State:Current state object

Goal_state:goal state for the puzzle problem

Returns the child nodes of a given particular node

Local Variables:

1.state of type numpy array

2.g_n,g,length of type int

3.child_list of type list

4.s1,s2,s3,s4 of type numpy array

5.h1,h2,h3,h4 of type int

6.f1,f2,f3,f4 of type int

7.child1, child2, child3, child4 of type node objects

5. check_existing(expanded_nodes,state):

Input Parameters:

expanded_nodes: list of expanded nodes

state: Current state object

Returns type boolean

Local variables:

1.element,node_element of type numpy arrays

Input(state,goal_state): it performs the complete A* algorithm.

Input Parameters:

State: object of current state

goal_state: goal state

Returns the final solution which is equal to goal state

Local variables:

1.generated_nodes of type dictionary

2.expanded_nodes of type list

3.all_generated_nodes of type list

4.generated_list of type list

5.state of type node object

6.parent of type numpyarray

7.result of type Boolean

8.top_element of type tupple

9.child_nodes of type list

6.get_g():Returns the G(n) value , where G(n) is path cost from root node to that specific node.

7.get_f(): Returns the F(n)(total cost) value of that specific node

8.get_node_element(): Returns the puzzle related to a particular node object

9.get_parent(): Returns the parent of a given particular node

Table Summarizing Initial State Goal State Number of Nodes Generated Number of Nodes Expanded using heuristic functions Misplaced Tiles and Manhattan Distance.

<table>
<tr><td>Initial State</td><td>Goal State</td><td colspan="2">Number of Nodes Generated</td><td colspan="2">Number of Nodes Expanded</td></tr>
<tr><td rowspan="2"><table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>7</td><td>4</td><td>5</td></tr><tr><td>6</td><td>8</td><td>0</td></tr></table></td><td rowspan="2"><table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>8</td><td>6</td><td>4</td></tr><tr><td>7</td><td>5</td><td>0</td></tr></table></td><td>Misplaced tiles</td><td>Manhattan distance</td><td>Misplaced tiles</td><td>Manhattan distance</td></tr>
<tr><td>61</td><td>26</td><td>21</td><td>9</td></tr>
</table>

<table>
<tr>
<td><table><tr><td>2</td><td>8</td><td>1</td></tr><tr><td>3</td><td>4</td><td>6</td></tr><tr><td>7</td><td>5</td><td>0</td></tr></table></td>
<td><table><tr><td>3</td><td>2</td><td>1</td></tr><tr><td>8</td><td>0</td><td>4</td></tr><tr><td>7</td><td>5</td><td>6</td></tr></table></td>
<td>20</td><td>17</td><td>7</td><td>6</td>
</tr>
<tr>
<td><table><tr><td>0</td><td>1</td><td>3</td></tr><tr><td>4</td><td>2</td><td>5</td></tr><tr><td>7</td><td>8</td><td>6</td></tr></table></td>
<td><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>7</td><td>8</td><td>0</td></tr></table></td>
<td>12</td><td>12</td><td>4</td><td>4</td>
</tr>
<tr>
<td><table><tr><td>8</td><td>1</td><td>3</td></tr><tr><td>4</td><td>0</td><td>2</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table></td>
<td><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>7</td><td>8</td><td>0</td></tr></table></td>
<td>826</td><td>206</td><td>301</td><td>76</td>
</tr>
</table>

1	2	3
7	4	5
6	8	0

Initial State

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

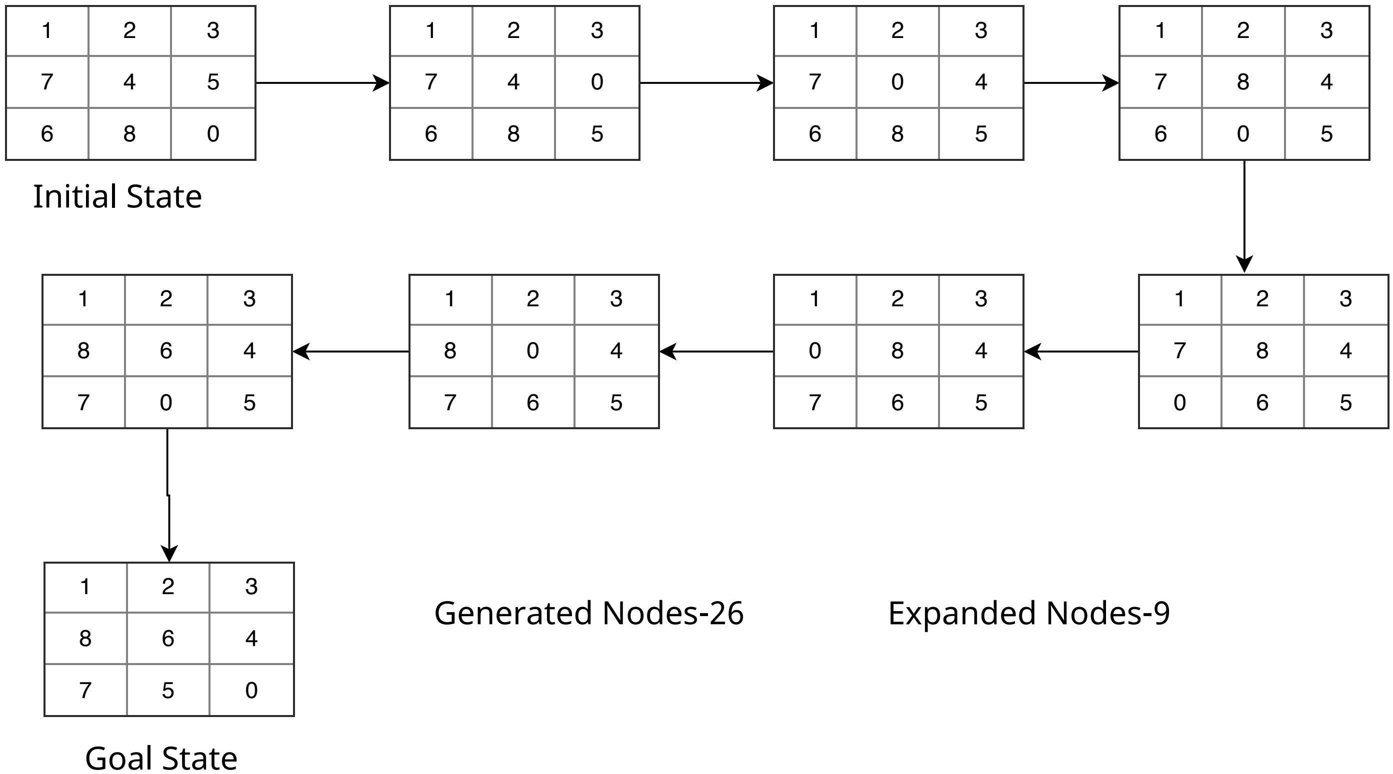
1	2	3
8	6	4
7	0	5

1	2	3
8	6	4
7	5	0

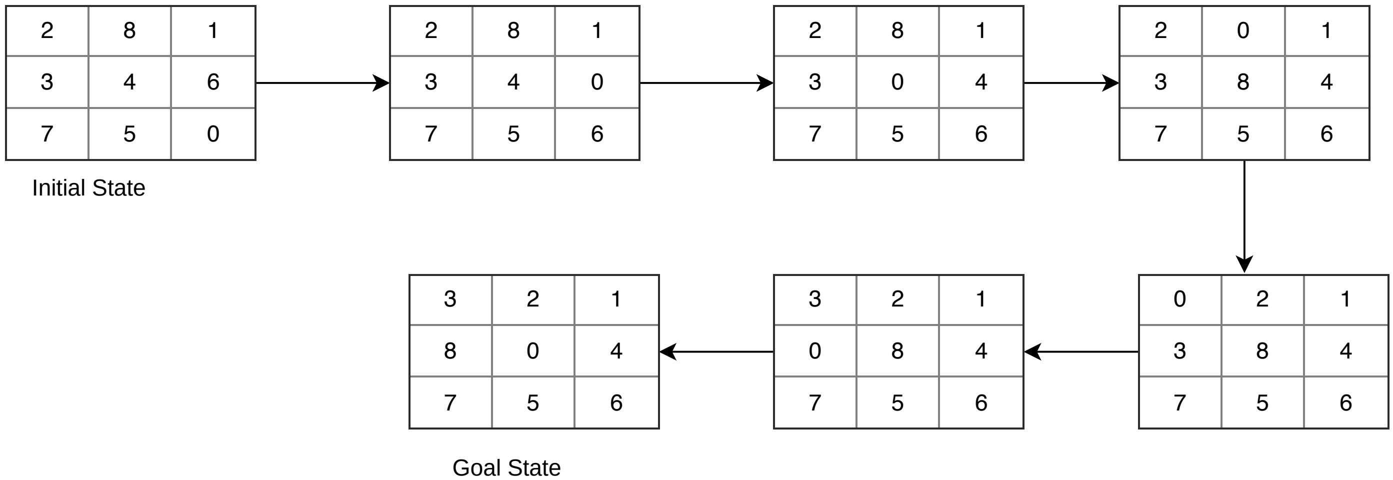
Goal State

Generated Nodes-61 Expanded Nodes-21

Implementation of A* algorithm using Misplaced Tiles

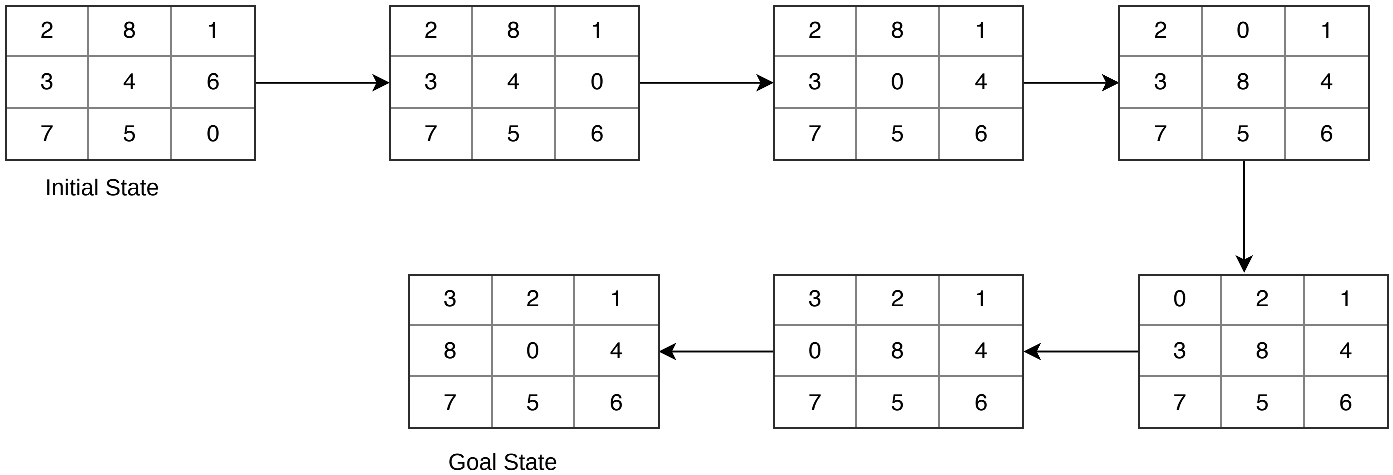


Implementation of A* algorithm using Manhattan distance



Generated Nodes-20 Expanded Nodes-7

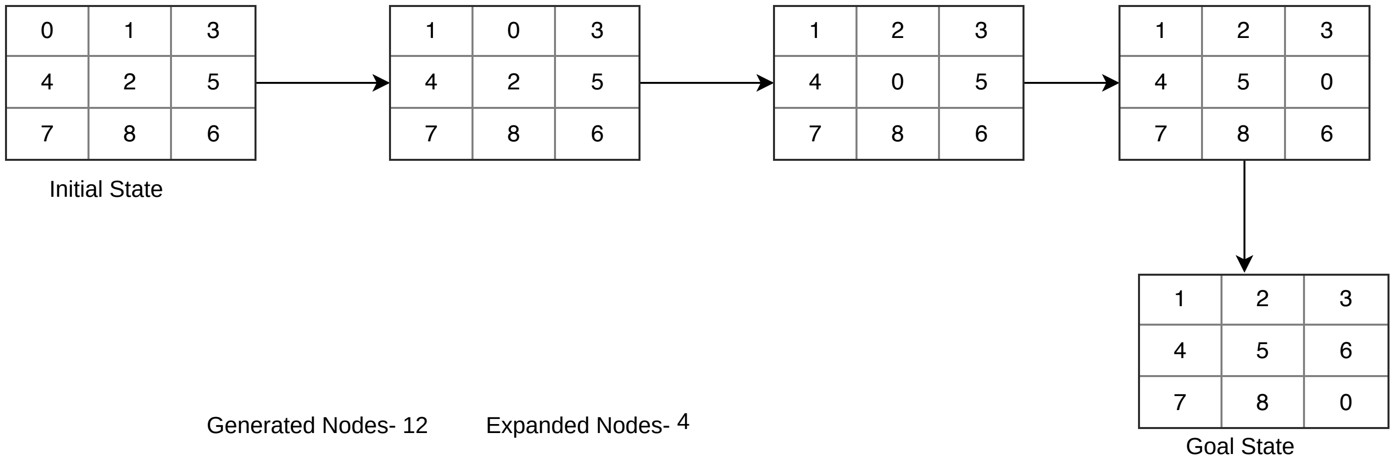
Implementation of A* algorithm using Misplaced Tiles



Generated Nodes-17

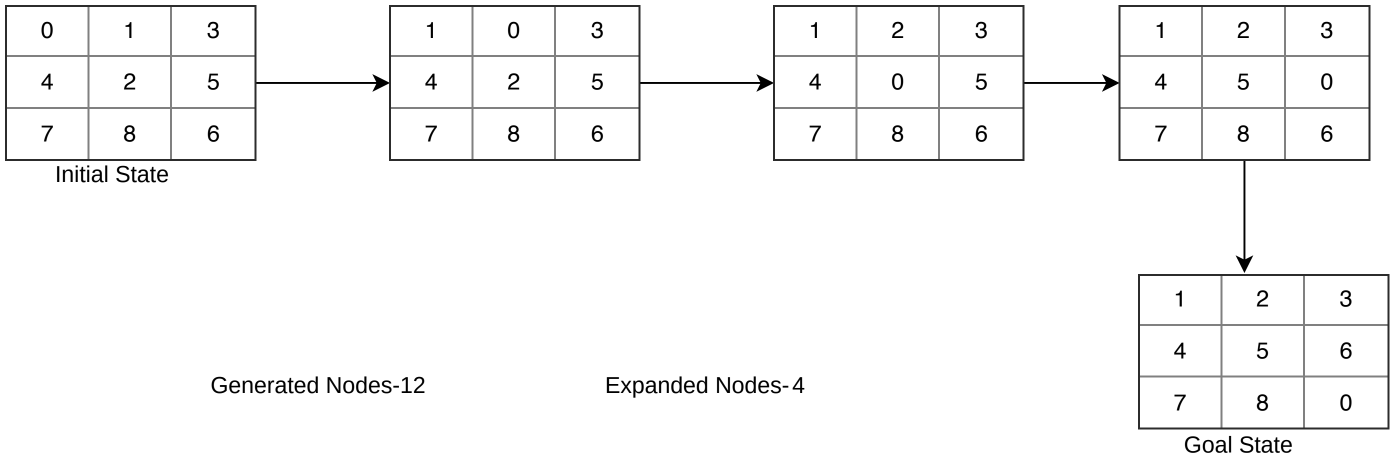
Expanded Nodes-6

Implementation of A* algorithm using Manhattan distance



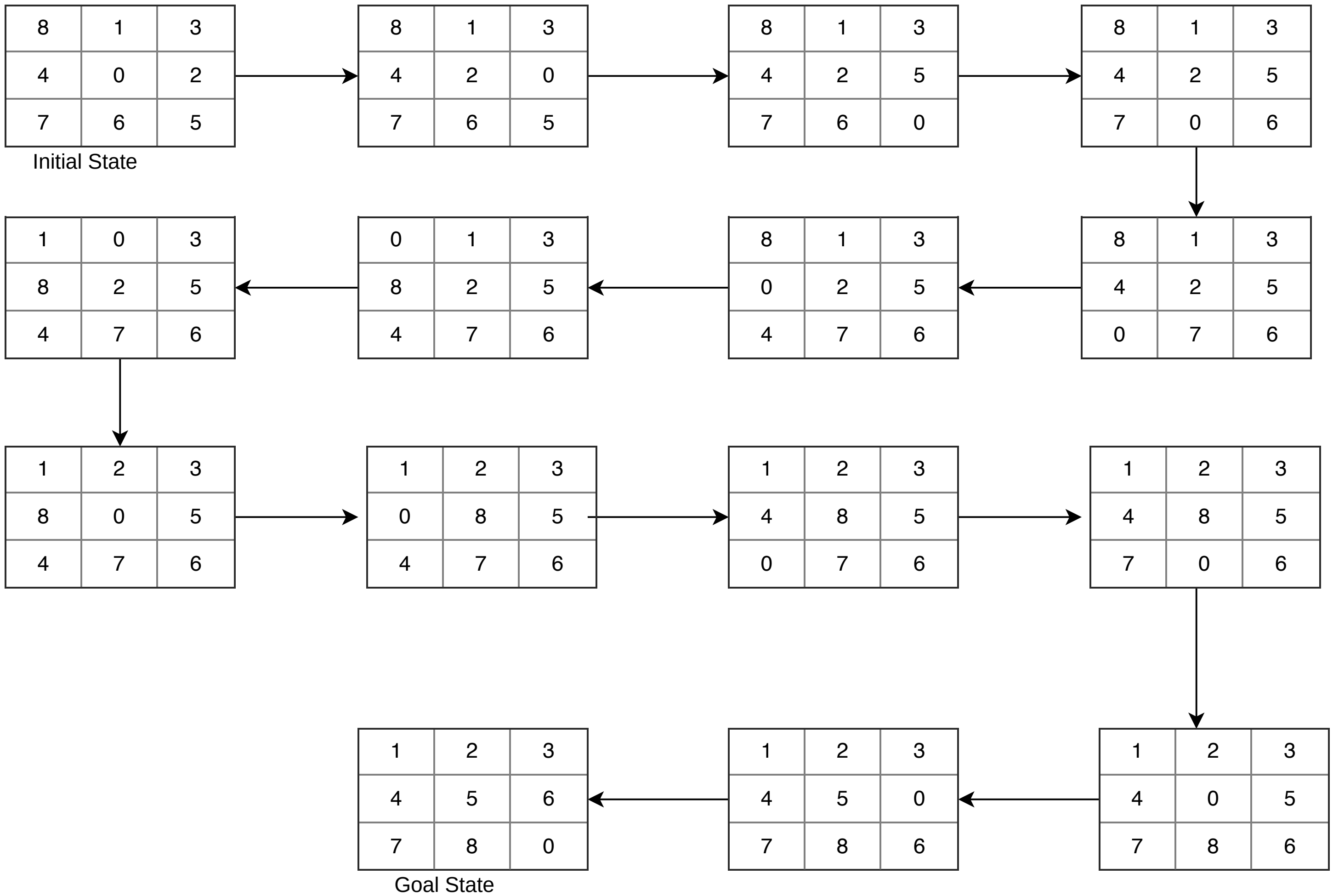
Generated Nodes- 12 Expanded Nodes- 4

Implementation of A* algorithm using Misplaced Tiles



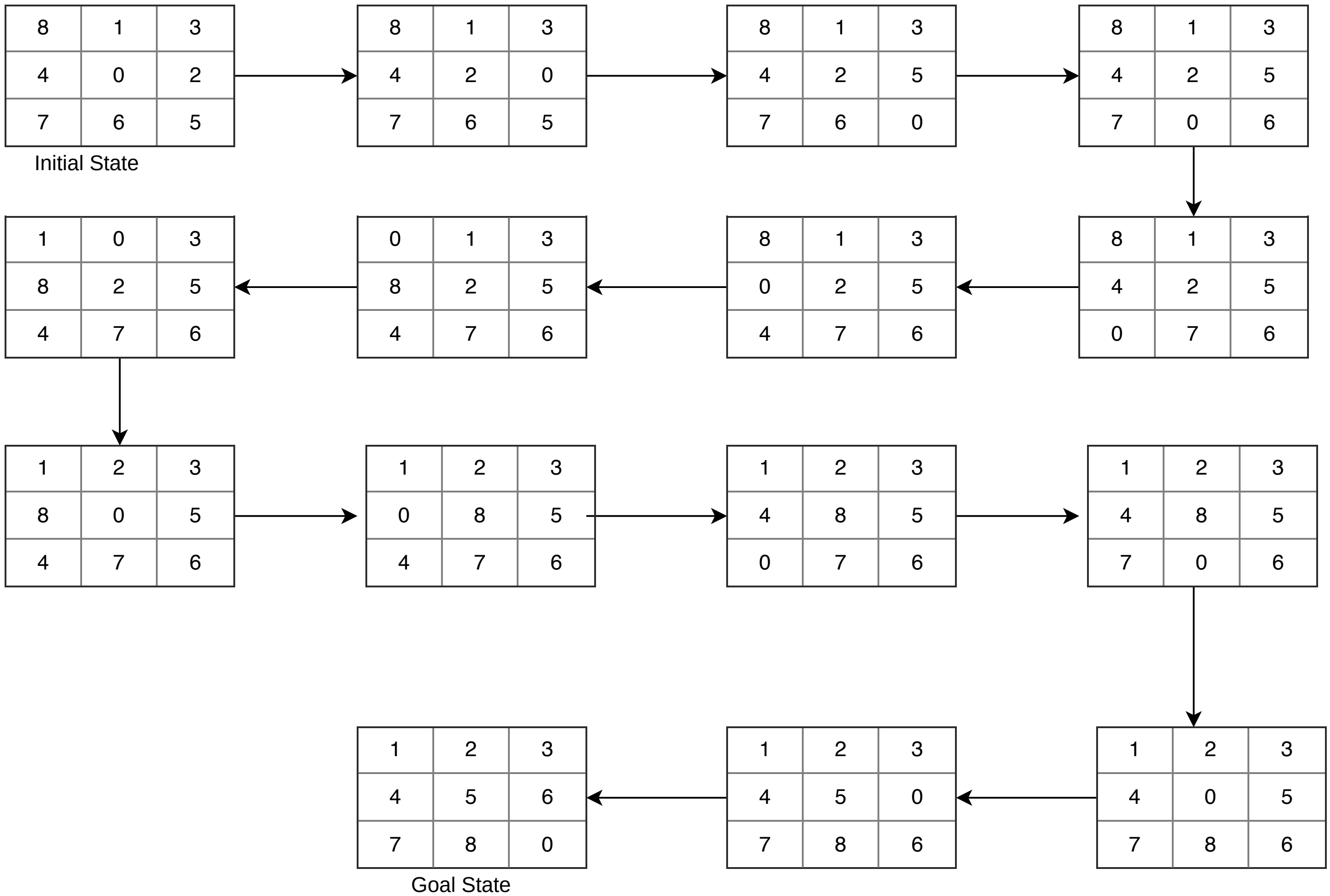
Generated Nodes-12 Expanded Nodes- 4

Implementation of A* algorithm using Manhattan Distance



Generated Nodes- 826 Expanded Nodes- 301

Implementation of A* algorithm using Misplaced Tiles



Generated Nodes- 206

Expanded Nodes- 76

Implementation of A* algorithm using manhattan distance