Date:

Assignment No-B1

Title:- A* algorithm.

Problem Statement:-
Solve 8 puzzle problem using A*. algorithm Assume any initial configuration and define goal configuration clearly.

Objectives:-
- Understand A* algorithm.
- Understand searching algorithms for 8 puzzle problem

Outcome:-
Solve 8 puzzle problem using A* algorithm

Software and Hardware Requirement:-
64 bit CPU, 4 GB RAM, UNIX/LINUX based OS, python3, Visual Studio Code/any IDE

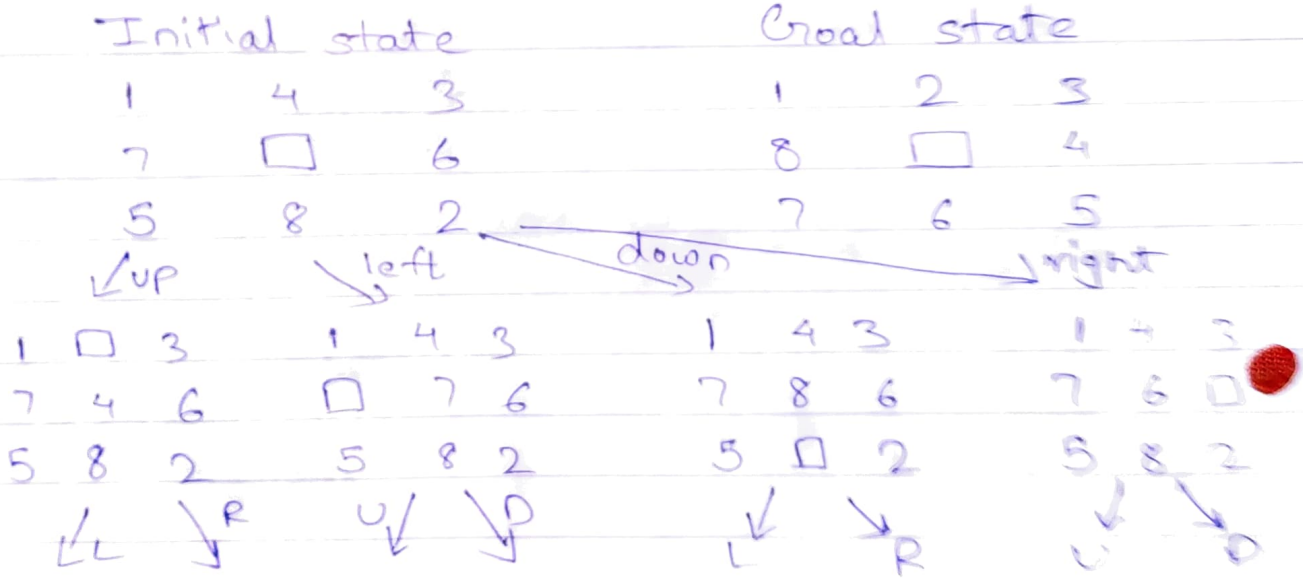
Theory:-

- 8 puzzle problem: N-puzzle or sliding puzzle is a popular puzzle that consists of N tiles where N can be 8, 15, 24 and so on.
- We are considering the case N=8. The puzzle is divided into sqrt(N+1) rows and

sqrt(N+1) columns. So, 8-puzzle will have 3 rows and 3 columns.

The puzzle consists of 8 tiles and one empty space where the tiles and one empty moved start and Goal configurations (also called state) of the puzzle are provided The puzzle can be solved by moving the tiles one by one in the single empty space and thus achieving the goal configuration.

State space of 8 puzzle problem



In this way 'children' states of current state can be deprived, because the empty space can only be moved in 4 directions which is further restricted by the position.

A* algorithm.

The A* algorithm integrates characteristics of uniform cost search and heuristic based search to find optimally efficient.

The key feature of A* is that it keeps track of each visited node which helps in ignoring the already visited nodes, as well as a list contains nodes yet to be explored. From this list it chooses the most optimal node.

So we use the two lists namely open list and closed list open list contains all the nodes that are being generated and are not existing in the closed list.

As each node is explored it is added to the closed list and its neighbours are added to the open list, this is how the nodes expand.

Each node has a pointer to its parent so that at any given point the path to the parent can be retraced.

The metric used to determine the optimal ness of a node is the f-score.

f-score = h-score + g-score

h-score: how far goal node is

g-score: number of nodes traversed from start node to current node

Thus A* uses a combination of heuristic value (h-score) + g score to calculate heuristic cost

The h-score is Manhattan distance (the distance between two points measured along axes at right angles) = abs($x_1 - x_2$) + abs($y_1 - y_2$).

Conclusion:-

Successfully implemented A* algorithm to solve 8 puzzle problem.