AIR - Assignment-1

Title:- A* algorithm

Problem statement:- Solve 8-puzzle problem using A* algorithm. Assume any initial config & define goal config clearly.

Objectives:- - To learn & understand use & need of A* algo.
- To apply A* algo to real time problem
- To implement A* algo using suitable programming language.

Outcomes:-
- Learn about A* algo.
- Apply A* algo to gaming problem
- Implement A* algo using Python

H/w & S/w req.:-

- OS : Ubuntu
- Eclipse IDE
- Python libraries

Theory:-

- A* is one of the most popular heuristic search algo. for finding paths in a graph.
- It is really a smart algo. which separates it from other conventional algos.

- Consider a square grid having many obstacles & we are given a starting cell & a target cell.
- We want to reach target cell from the starting cell as quickly as possible
- What A* algo. does is at each step, it picks the node according to a value '-f' which is a parameter equal to sum of other two parameters - g & h.
- At each step, it picks the node cell having least '-f' & process that node/cell.

- We define 'g' & 'h' simply as possible

g = the movement cost to move from the starting point to a given square on the grid following the path generated to get there.

h = the estimated movement cost to move from that given square on the grid to the final destination. This is often referred to as the heuristic which is nothing but a kind of smart guess.

- We really dont know the actual division until we find the path because all sorts of things can be in the way.

Algorithm:-

```
1) Initialize the open list
2) Initialize the closed list
   put the starting node on the open list.
3) While the open list is not empty
   a) Find the node with the least f
      on the open list. Call it 'q'.
   b) pop 'q' off open list.
   c) Generate q's successors.
   d) For each successor:
      i) if successor is the goal, stop
         search successor. $g = q.g + distance(successor.q)$
         $successor.h$ = distance from goal to successor
         $successor.f = successor.g + successor.h$

      ii) If a node with the same position
          as successor is in the open list which
          has a lower 'f' than successor,
          skip this successor.
      iii) If a node with the same
          position as successor is in the closed
          list which has a lower 'f' than
          successor, skip this successor otherwise
          & the node to the open list.
   e) end for
   f) push q on the closed list
4) End while
```

Test cases:-

Initial config.

1	2	x
4	5	3
7	8	6

Final config

1	2	3
4	5	6
7	8	x

Output:-

The puzzle is solved in 18 moves

Conclusion:-

We successfully implemented A* algo. for 8-puzzle problem.