Assignment : B1

- TITLE : Solve 8-Puzzle problem using A* Algorithm Assume any initial configuration and define goal configuration clearly.

- Objective :
  - To learn and understand use and need of A* algorithm.
  - To apply A* algorithm to real time problem
  - To implement A* algorithm using suitable programming language.

- Outcome : we will able to :
  - learn about A* algorithm
  - Apply A* algorithm to solve 8-puzzle problem.

- Software and Hardware Requirements : OS : Fedora 20/Ubuntu (64-bit)
  RAM : 4GB
  HDD : 500 GB
  Python libraries, Python framework, Java framework.

- Theory :

- A* is the most popular heuristic search Algorithm for finding path in a graph.
- A* (A star) is a search algorithm that is used for finding path from one node to another. So it can be computed with Breadth first search or Dijkstra's algorithm or Depth first search. A* algorithm is

widely used in graph search for being better in efficiency and accuracy where graph Pre-Processing is not an option.

- A* is an specialization of Best first search, in which the function of evaluation f is define in a particular way.
- $f(n) = g(n) + h(n)$ is the minimum cost since the initial node to the objective conditioned to go thought node n.
- $g(n)$ is the minimum cost from the initial node to n.
- $h(n)$ is the minimum cost from n to the cloest objective to n.
- A* is an informed search Algorithm and it always guarantees to find the smallest path in the least possible time if (uses admissible heuristic). so it is both complete and optimal.

For example:

An 8 puzzle game consist of a 3x3 grid (containing 9 squares). one of the square is empty. The object is to move to squares around into different positions and having the numbers displayed in the goal state.

Initial state:			goal state:		
_	1	3	1	2	3
4	2	5	4	5	6
7	8	6	7	8	_

- Heuristic to be assumed:

Let us consider the Manhattan distance between the current and final state as the heuristic for this Problem statement.

$$h(n) = |x-P| + |y-q|$$

where x and y are cell co-ordinates in the current state.

P and q are cell co-ordinates in the final state.

- Total cost function:

So the total cost function $f(n)$ is given by $f(n) = g(n) + h(n)$ where $g(n)$ is the cost required to reach the current state from given initial state.

Algorithm:

```
1. Initialize the open list
2. Initialize the close list
        Put the starting node in the open list
3. while openlist is not empty
      1. find the node with least f on the open list
         call it 'q'
      2. Pop 'q' off open list
      3. generate 'q''s successor
      4. for each successor
            1. if successor is the goal, stop search
            g = q.g + distance(successor, q)
            successor.h = distance from goal to successor
            successor.f = ~~distance from~~ successor.g +
            2. if a node with the same Position as successor
               is in the open list which has a lower 'f'
               score than successor skip this successor.
            3. 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 to the open list.
        5. end for
        6. Push q on the closed list
    4. end while.
```

- Test cases :

	Discription	Expected output	Actual output
i]	Start State : 1 2 3 4 - 5 7 8 6 goal State : 1 2 3 4 5 6 7 - 8	Goal State : 1 2 3 4 5 6 7 - 8	Goal State : 1 2 3 4 5 6 7 - 8 Result : Pass.

- Conclusion : Thus we successfully implemented 8-Puzzle Problem and solved using A* Algorithm.