Implement A* search algorithm using 8-puzzle

Explanation:
The puzzle consists of 8 tiles and one empty space where the tiles can be 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. The tiles in the initial(start) state can be moved in the empty space in a particular order and thus achieve the goal state.

Rules for solving the puzzle:
Instead of moving the tiles in the empty space, we can visualize moving the empty space in place of the tile, basically swapping the tile with the empty space. The empty space can only move in four directions viz.,

- ✓ Up
- ✓ Down
- ✓ Right
- ✓ Left

The empty space cannot move diagonally and can take only one step at a time (i.e. move the empty space one position at a time).

Using A* search algorithm:
A* uses a combination of heuristic value (h-score: how far the goal node is) as well as the g-score (i.e. the number of nodes traversed from the start node to current node).

f-score = h-score + g-score

In our 8-Puzzle problem, we can define the **h-score** as the number of misplaced tiles by comparing the current state and the goal state or summation of the Manhattan distance between misplaced nodes.
g-score will remain as the number of nodes traversed from a start node to get to the current node.

Implementation of algorithm:
Below are the functions used in the implementation of a* search.

1. def heuristic(states, target):
   # Calculates Manhattan distance (Heuristic Function)
2. def astarSearch(source, target):
   # Implements A* search algorithm

3. def possibleMoves(state, visitedStates):
   #To find all possible moves of empty tile
4. def moveTile(state, direction, b):
   #Moves a tile Up, Down, Left, Right
5. def printState(source):
   #Prints the required states