```
Total time: 0.049s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: expand at line 51

Line #      Hits         Time      Per Hit    % Time   Line Contents
==============================================================
    51                                                 @cpu
    52                                                 def expand(board):
    53      1504         0.4ms          .      0.8%         for i in range(len(board.data)):                    # to find the loca
    54      3771         1.0ms          .      2.1%             for j in range(len(board.data[i])):
    55      3019         0.8ms          .      1.7%                 if board.data[i][j] == '*':
    56       376         0.1ms          .      0.2%                     location = [i,j];
    57       376         0.1ms          .      0.2%                     break
    58
    59       376         0.1ms          .      0.1%         actions = []
    60      1413         5.1ms          .     10.4%         for move in possible_actions(constants.board, location):    # to find all poss
    61      1037        41.4ms          .     84.3%             actions.append([result(location, move, board.data) , move]) # prepare all poss
    62
    63       376         0.1ms          .      0.1%         return actions                                      # After expanding


Total time: 0.003s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: possible_actions at line 67

Line #      Hits         Time      Per Hit    % Time   Line Contents
==============================================================
    67                                                 @cpu
    68                                                 def possible_actions(board, location):                  # to find all poss
    69       376         0.1ms          .      4.6%         actions = ["RIGHT","LEFT","UP","DOWN"]
    70       376         0.1ms          .      3.3%         actionstopeform = []
    71
    72      1880         0.5ms          .     18.0%         for x in actions:
    73                                                         # for moving right
    74      1504         0.3ms          .     13.4%             if x == "RIGHT":
    75       376         0.1ms          .      5.5%                 if location[1]+1 < len(board):
    76       254         0.1ms          .      4.3%                     actionstopeform.append([x,location[0],location[1]+1])
    77                                                         # for moving left
    78      1128         0.3ms          .     11.1%             elif x == "LEFT":
    79       376         0.1ms          .      4.7%                 if location[1]-1 >= 0:
    80       265         0.1ms          .      4.1%                     actionstopeform.append([x,location[0],location[1]-1])
    81                                                         # for moving up
    82       752         0.2ms          .      6.3%             elif x == "UP":
    83       376         0.1ms          .      4.8%                 if location[0]-1 >= 0:
    84       265         0.1ms          .      4.0%                     actionstopeform.append([x,location[0]-1,location[1]])
    85                                                         # for moving down
    86       376         0.1ms          .      4.0%             elif x == "DOWN":
    87       376         0.1ms          .      4.4%                 if location[0]+1 < len(board):
    88       253         0.1ms          .      3.9%                     actionstopeform.append([x,location[0]+1,location[1]])
    89
    90       376         0.1ms          .      3.5%         return actionstopeform


Total time: 0.039s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: result at line 94

Line #      Hits         Time      Per Hit    % Time   Line Contents
==============================================================
    94                                                 @cpu
    95                                                 def result(location,action,board):
    96      1037        32.0ms          .     81.8%         newBoard = copy.deepcopy(board)                     # copy of a board so t
    97      1037         2.4ms          .      6.1%         temp = copy.deepcopy(newBoard[action[1]][action[2]])
    98      1037         2.3ms          .      5.9%         newBoard[action[1]][action[2]] = copy.deepcopy('*')
    99      1037         2.2ms          .      5.7%         newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
   100      1037         0.2ms          .      0.5%         return newBoard                                     # return new board aft


Total time: 0.000s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: misplaced at line 104

Line #      Hits         Time      Per Hit    % Time   Line Contents
==============================================================
   104                                                 @cpu
   105                                                 def misplaced(puzzle):
   106                                                     num_misplaced = 0
   107                                                     for i in range(len(puzzle.data)):
   108                                                         for j in range(len(puzzle.data)):
   109                                                             if puzzle.data[i][j] != constants.goalBoard[i][j] and puzzle.data[i][j] !=
   110                                                                 num_misplaced += 1
   111                                                     return num_misplaced


Total time: 0.015s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: manhattan at line 114

Line #      Hits         Time      Per Hit    % Time   Line Contents
==============================================================
   114                                                 @cpu
   115                                                 def manhattan(state):
   116       634         0.2ms          .      1.2%         state = state.data
   117       634         0.2ms          .      1.3%         goal_state = constants.goalBoard
   118       634         0.1ms          .      0.9%         distance = 0
   119
   120                                                         # Create a dictionary that maps each value to its position in the goal state
   121       634         0.2ms          .      1.1%         goal_dict = {}
```

```
122    2536     0.7ms          .      4.8%          for i in range(len(goal_state)):
123    7608     2.2ms          .     14.2%              for j in range(len(goal_state[0])):
124    5706     1.7ms          .     11.1%                  if goal_state[i][j] != '*':
125    5072     1.7ms          .     11.0%                      goal_dict[goal_state[i][j]] = (i, j)
126
127                                                     # Calculate Manhattan distance
128    2536     0.7ms          .      4.6%          for i in range(len(state)):
129    7608     2.2ms          .     14.1%              for j in range(len(state[0])):
130    5706     2.3ms          .     14.9%                  if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
131    3294     0.9ms          .      5.6%                      value = state[i][j]
132    3294     0.9ms          .      6.1%                      row, col = goal_dict[value]
133    3294     1.2ms          .      8.1%                      distance += abs(row - i) + abs(col - j)
134
135     634     0.2ms          .      1.0%          return distance
```

Total time: 0.000s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: linear_conflict at line 137

```
Line #      Hits         Time   Per Hit   % Time  Line Contents
==============================================================
137                                               @cpu
138                                               def linear_conflict(board, goal):
139                                                   n = len(board)
140                                                   linear_conflicts = 0
141
142                                                   # Find the linear conflicts in rows
143                                                   for i in range(n):
144                                                       row = board[i]
145                                                       goal_row = goal[i]
146                                                       for j in range(n):
147                                                           if row[j] != '*' and row[j] in goal_row:
148                                                               for k in range(j+1, n):
149                                                                   if row[k] != '*' and row[k] in goal_row and goal_row.index(row[j])
150                                                                       linear_conflicts += 2
151
152                                                   # Find the linear conflicts in columns
153                                                   for j in range(n):
154                                                       column = [board[i][j] for i in range(n)]
155                                                       goal_column = [goal[i][j] for i in range(n)]
156                                                       for i in range(n):
157                                                           if column[i] != '*' and column[i] in goal_column:
158                                                               for k in range(i+1, n):
159                                                                   if column[k] != '*' and column[k] in goal_column and goal_column.i
160                                                                       linear_conflicts += 2
161
162                                                   return linear_conflicts
```

Total time: 0.000s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: f at line 165

```
Line #      Hits         Time   Per Hit   % Time  Line Contents
==============================================================
165                                               @cpu
166                                               # Heuristic Function to calculate hueristic value f(x) = h(x) + g(x)
167                                               def f(board):
168                                                   manhattan_distance = manhattan(board)
169                                                   manhattan_distance += linear_conflict(board.data, constants.goalBoard) # Add l
170                                                   return manhattan_distance + board.depth
```

Total time: 0.000s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: zero_function at line 174

```
Line #      Hits         Time   Per Hit   % Time  Line Contents
==============================================================
174                                               @cpu
175                                               def zero_function(board):
176                                                   return 0
```

Total time: 0.091s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: a_star at line 179

```
Line #      Hits         Time   Per Hit   % Time  Line Contents
==============================================================
179                                               @memory_profiler.profile
180                                               @cpu
181                                               def a_star(initialProblem, f):
182       1        .          .          .            initialNode = Node(data = initialProblem)              # node←NODE(STATE=problem.
183       1        .          .          .            frontier = PriorityQueue()
184       1     0.1ms      0.1ms       0.1%           frontier.append((f(initialNode), initialNode))         # frontier←a priority queu
185
186       1        .          .          .            reached = {str(initialProblem): initialNode}           # reached←a lookup table,
187
188     377     0.2ms          .       0.2%           while not frontier.empty():                            # while not IS-EMPTY(front
189     377     0.2ms          .       0.3%               node = frontier.get()                              # node←POP(frontier)
190
191     377     0.2ms          .       0.2%               if constants.goalBoard == node[1].data:            # if problem.IS-GOAL(node.
192                                                           #print('Max queue size:', frontier.getSize())  # only for debug
193       1        .          .          .                    return node[1]                                 # then return node
194
195    1413    52.2ms          .      57.1%               for child in expand(node[1]):                      # for each child in EXPAND
```

```
196                                                        # s←child.STATE
197      1037        1.2ms           .         1.4%        s =  Node( data = child[0], depth = node[1].depth + 1, move = child[1], pr
198
199                                                        # if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
200      1037        1.9ms           .         2.1%        if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
201       633        0.8ms           .         0.9%            reached[str(s.data)] = s                    # reached[s]←child
202       633       34.5ms        0.1ms       37.8%            frontier.append((f(s) ,s))                  # add child to frontier
203
204                                                        return constants.failure                        # return failure
```

```
Total time: 0.000s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/assignment_1/a_star.py
Function: printStatistics at line 207

Line #      Hits         Time      Per Hit   % Time  Line Contents
==============================================================
207                                                  @cpu
208                                                  def printStatistics(solution):
209         1           .            .         0.4%      pathCost = 0
210         1           .            .         0.4%      stateSequence = []
211         1           .            .          .        actionSequence = []
212
213        50           .            .         4.8%      while solution.prev != None:
214        49           .            .         6.1%          stateSequence.insert(0, solution.data)
215        49           .            .         6.9%          actionSequence.insert(0, solution.move)
216        49           .            .         4.3%          solution = solution.prev
217        49           .            .         5.6%          pathCost += 1
218
219         1           .            .         1.7%      print('Action sequence:')
220         1         0.1ms        0.1ms      29.0%      print(*actionSequence, sep='\n')
221
222         1           .            .         1.3%      print('\nState sequence:')
223         1         0.1ms        0.1ms      38.1%      print(*stateSequence, sep='\n')
224
225         1           .            .         1.3%      print('\nPath cost:', pathCost)
```