```
Total time: 0.004s
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       136          .            .      1.2%        for i in range(len(board.data)):                                # to find the loca
    54       338       0.1ms           .      2.2%            for j in range(len(board.data[i])):
    55       270       0.1ms           .      1.9%                if board.data[i][j] == '*':
    56        34          .            .      0.2%                    location = [i,j];
    57        34          .            .      0.1%                    break
    58
    59        34          .            .      0.1%        actions = []
    60       119       0.5ms           .     11.2%        for move in possible_actions(constants.board, location):        # to find all poss
    61        85       3.5ms           .     83.0%            actions.append([result(location, move, board.data) , move]) # prepare all poss
    62
    63        34          .            .      0.1%        return actions                                                  # After expanding


Total time: 0.000s
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        34          .            .      5.1%        actions = ["RIGHT","LEFT","UP","DOWN"]
    70        34          .            .      1.4%        actionstopeform = []
    71
    72       170          .            .     15.7%        for x in actions:
    73                                                        # for moving right
    74       136          .            .     17.6%            if x == "RIGHT":
    75        34          .            .      5.1%                if location[1]+1 < len(board):
    76        22          .            .      4.6%                    actionstopeform.append([x,location[0],location[1]+1])
    77                                                        # for moving left
    78       102          .            .      8.8%            elif x == "LEFT":
    79        34          .            .      4.6%                if location[1]-1 >= 0:
    80        20          .            .      3.2%                    actionstopeform.append([x,location[0],location[1]-1])
    81                                                        # for moving up
    82        68          .            .      8.8%            elif x == "UP":
    83        34          .            .      3.2%                if location[0]-1 >= 0:
    84        22          .            .      5.1%                    actionstopeform.append([x,location[0]-1,location[1]])
    85                                                        # for moving down
    86        34          .            .      1.9%            elif x == "DOWN":
    87        34          .            .      7.9%                if location[0]+1 < len(board):
    88        21          .            .      3.2%                    actionstopeform.append([x,location[0]+1,location[1]])
    89
    90        34          .            .      3.7%        return actionstopeform


Total time: 0.003s
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        85       2.7ms           .     81.5%        newBoard = copy.deepcopy(board)                             # copy of a board so t
    97        85       0.2ms           .      6.3%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    98        85       0.2ms           .      5.8%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    99        85       0.2ms           .      5.8%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
   100        85          .            .      0.6%        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: 0.001s
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        53          .            .      1.1%        state = state.data
   117        53          .            .      0.9%        goal_state = constants.goalBoard
   118        53          .            .      1.2%        distance = 0
   119
   120                                                   # Create a dictionary that maps each value to its position in the goal state
   121        53          .            .      0.7%        goal_dict = {}
```

```
   122       212        0.1ms            .      5.4%        for i in range(len(goal_state)):
   123       636        0.2ms            .     13.2%            for j in range(len(goal_state[0])):
   124       477        0.1ms            .     10.1%                if goal_state[i][j] != '*':
   125       424        0.1ms            .     10.4%                    goal_dict[goal_state[i][j]] = (i, j)
   126
   127                                                            # Calculate Manhattan distance
   128       212        0.1ms            .      4.3%        for i in range(len(state)):
   129       636        0.2ms            .     11.2%            for j in range(len(state[0])):
   130       477        0.2ms            .     15.4%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   131       348        0.1ms            .      7.1%                    value = state[i][j]
   132       348        0.1ms            .      7.3%                    row, col = goal_dict[value]
   133       348        0.2ms            .     10.9%                    distance += abs(row - i) + abs(col - j)
   134
   135        53            .            .      0.7%        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: 0.008s
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      0.8%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queu
   185
   186         1            .            .      0.1%        reached = {str(initialProblem): initialNode}           # reached←a lookup table,
   187
   188        35            .            .      0.3%        while not frontier.empty():                            # while not IS-EMPTY(front
   189        35            .            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   190
   191        35            .            .      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       119        4.4ms            .     58.8%            for child in expand(node[1]):                      # for each child in EXPANI
```

```
   196                                                   # s←child.STATE
   197        85        0.1ms            .      1.3%      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        85        0.1ms            .      2.0%      if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   201        52        0.1ms            .      0.9%          reached[str(s.data)] = s                    # reached[s]←child
   202        52        2.7ms        0.1ms     35.3%          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            .            .      0.3%       pathCost = 0
   210         1            .            .      0.3%       stateSequence = []
   211         1            .            .         .       actionSequence = []
   212
   213        29            .            .      2.8%       while solution.prev != None:
   214        28            .            .      2.8%           stateSequence.insert(0, solution.data)
   215        28            .            .      3.4%           actionSequence.insert(0, solution.move)
   216        28            .            .      1.9%           solution = solution.prev
   217        28            .            .      3.1%           pathCost += 1
   218
   219         1            .            .      1.2%       print('Action sequence:')
   220         1            .            .     12.0%       print(*actionSequence, sep='\n')
   221
   222         1            .            .      0.6%       print('\nState sequence:')
   223         1        0.2ms        0.2ms     70.1%       print(*stateSequence, sep='\n')
   224
   225         1            .            .      1.5%       print('\nPath cost:', pathCost)
```