```
Total time: 0.001s
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        28            .            .      0.8%        for i in range(len(board.data)):                                # to find the loca
    54        66            .            .      1.6%            for j in range(len(board.data[i])):
    55        52            .            .      1.6%                if board.data[i][j] == '*':
    56         7            .            .      0.2%                    location = [i,j];
    57         7            .            .      0.2%                    break
    58
    59         7            .            .      0.2%        actions = []
    60        28       0.1ms            .     10.3%        for move in possible_actions(constants.board, location):        # to find all poss
    61        21       0.9ms            .     85.1%            actions.append([result(location, move, board.data) , move]) # prepare all poss
    62
    63         7            .            .      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         7            .            .      4.3%        actions = ["RIGHT","LEFT","UP","DOWN"]
    70         7            .            .      2.1%        actionstopeform = []
    71
    72        35            .            .     23.4%        for x in actions:
    73                                                          # for moving right
    74        28            .            .     10.6%            if x == "RIGHT":
    75         7            .            .      6.4%                if location[1]+1 < len(board):
    76         7            .            .      4.3%                    actionstopeform.append([x,location[0],location[1]+1])
    77                                                          # for moving left
    78        21            .            .     14.9%            elif x == "LEFT":
    79         7            .            .      6.4%                if location[1]-1 >= 0:
    80         3            .            .      2.1%                    actionstopeform.append([x,location[0],location[1]-1])
    81                                                          # for moving up
    82        14            .            .      4.3%            elif x == "UP":
    83         7            .            .      2.1%                if location[0]-1 >= 0:
    84         6            .            .      2.1%                    actionstopeform.append([x,location[0]-1,location[1]])
    85                                                          # for moving down
    86         7            .            .      8.5%            elif x == "DOWN":
    87         7            .            .      2.1%                if location[0]+1 < len(board):
    88         5            .            .      6.4%                    actionstopeform.append([x,location[0]+1,location[1]])
    89
    90         7            .            .         .        return actionstopeform


Total time: 0.001s
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        21       0.7ms            .     81.7%        newBoard = copy.deepcopy(board)                                 # copy of a board so t
    97        21            .            .      5.7%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    98        21       0.1ms            .      6.3%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    99        21            .            .      5.4%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
   100        21            .            .      0.9%        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: 0.000s
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        16            .            .      2.1%        state = state.data
   117        16            .            .      1.3%        goal_state = constants.goalBoard
   118        16            .            .      1.3%        distance = 0
   119
   120                                                      # Create a dictionary that maps each value to its position in the goal state
   121        16            .            .      0.3%        goal_dict = {}
```

```
   122        64            .            .      6.5%        for i in range(len(goal_state)):
   123       192        0.1ms            .     16.7%            for j in range(len(goal_state[0])):
   124       144            .            .     12.8%                if goal_state[i][j] != '*':
   125       128            .            .     11.5%                    goal_dict[goal_state[i][j]] = (i, j)
   126
   127                                              # Calculate Manhattan distance
   128        64            .            .      5.0%        for i in range(len(state)):
   129       192        0.1ms            .     14.9%            for j in range(len(state[0])):
   130       144        0.1ms            .     14.9%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   131        51            .            .      3.4%                    value = state[i][j]
   132        51            .            .      3.1%                    row, col = goal_dict[value]
   133        51            .            .      5.5%                    distance += abs(row - i) + abs(col - j)
   134
   135        16            .            .      0.8%        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: 0.001s
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        16        0.6ms            .     99.7%            manhattan_distance = manhattan(board)
   169                                                           # manhattan_distance += linear_conflict(board.data, constants.goalBoard) # Add
   170        16            .            .      0.3%            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: 0.002s
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            .            .      0.2%        initialNode = Node(data = initialProblem)              # node←NODE(STATE=problem.
   183         1            .            .      0.1%        frontier = PriorityQueue()
   184         1            .            .      2.7%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queu
   185
   186         1            .            .      0.2%        reached = {str(initialProblem): initialNode}           # reached←a lookup table,
   187
   188         8            .            .      0.4%        while not frontier.empty():                            # while not IS-EMPTY(front
   189         8            .            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   190
   191         8            .            .      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        28        1.1ms            .     57.7%            for child in expand(node[1]):                      # for each child in EXPAND
```

```
   196                                                  # s←child.STATE
   197        21            .            .      1.5%    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        21            .            .      1.9%    if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   201        15            .            .      1.1%        reached[str(s.data)] = s                    # reached[s]←child
   202        15        0.6ms            .     33.8%        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            .            .         .        pathCost = 0
   210         1            .            .         .        stateSequence = []
   211         1            .            .      2.4%        actionSequence = []
   212
   213         7            .            .      4.9%        while solution.prev != None:
   214         6            .            .      4.9%            stateSequence.insert(0, solution.data)
   215         6            .            .      2.4%            actionSequence.insert(0, solution.move)
   216         6            .            .      2.4%            solution = solution.prev
   217         6            .            .      7.3%            pathCost += 1
   218
   219         1            .            .      7.3%        print('Action sequence:')
   220         1            .            .     26.8%        print(*actionSequence, sep='\n')
   221
   222         1            .            .      4.9%        print('\nState sequence:')
   223         1            .            .     29.3%        print(*stateSequence, sep='\n')
   224
   225         1            .            .      7.3%        print('\nPath cost:', pathCost)
```