```
Total time: 0.013s
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       412        0.1ms            .      0.9%        for i in range(len(board.data)):                                # to find the loca
    54      1013        0.3ms            .      1.9%            for j in range(len(board.data[i])):
    55       807        0.2ms            .      1.8%                if board.data[i][j] == '*':
    56       103            .            .      0.2%                    location = [i,j];
    57       103            .            .      0.2%                    break
    58
    59       103            .            .      0.1%        actions = []
    60       381        1.4ms            .     10.6%        for move in possible_actions(constants.board, location):        # to find all poss
    61       278       11.1ms            .     84.1%            actions.append([result(location, move, board.data) , move]) # prepare all poss
    62
    63       103            .            .      0.2%        return actions                                                  # After expanding


Total time: 0.001s
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       103            .            .      3.7%        actions = ["RIGHT","LEFT","UP","DOWN"]
    70       103            .            .      3.4%        actionstopeform = []
    71
    72       515        0.1ms            .     17.2%        for x in actions:
    73                                                       # for moving right
    74       412        0.1ms            .     13.9%            if x == "RIGHT":
    75       103            .            .      5.5%                if location[1]+1 < len(board):
    76        80            .            .      4.9%                    actionstopeform.append([x,location[0],location[1]+1])
    77                                                       # for moving left
    78       309        0.1ms            .     10.3%            elif x == "LEFT":
    79       103            .            .      4.9%                if location[1]-1 >= 0:
    80        63            .            .      4.5%                    actionstopeform.append([x,location[0],location[1]-1])
    81                                                       # for moving up
    82       206            .            .      6.9%            elif x == "UP":
    83       103            .            .      4.9%                if location[0]-1 >= 0:
    84        65            .            .      3.6%                    actionstopeform.append([x,location[0]-1,location[1]])
    85                                                       # for moving down
    86       103            .            .      3.4%            elif x == "DOWN":
    87       103            .            .      4.8%                if location[0]+1 < len(board):
    88        70            .            .      4.2%                    actionstopeform.append([x,location[0]+1,location[1]])
    89
    90       103            .            .      3.4%        return actionstopeform


Total time: 0.010s
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       278        8.5ms            .     81.8%        newBoard = copy.deepcopy(board)                             # copy of a board so t
    97       278        0.6ms            .      6.2%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    98       278        0.6ms            .      5.8%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    99       278        0.6ms            .      5.8%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
   100       278            .            .      0.4%        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: 0.004s
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       172        0.1ms            .      1.4%        state = state.data
   117       172            .            .      1.1%        goal_state = constants.goalBoard
   118       172            .            .      1.1%        distance = 0
   119
   120                                                       # Create a dictionary that maps each value to its position in the goal state
   121       172            .            .      0.9%        goal_dict = {}
```

```
   122       688         0.2ms            .       5.2%        for i in range(len(goal_state)):
   123      2064         0.5ms            .      14.2%            for j in range(len(goal_state[0])):
   124      1548         0.4ms            .      11.1%                if goal_state[i][j] != '*':
   125      1376         0.4ms            .      11.0%                    goal_dict[goal_state[i][j]] = (i, j)
   126
   127                                                             # Calculate Manhattan distance
   128       688         0.2ms            .       4.7%        for i in range(len(state)):
   129      2064         0.6ms            .      14.7%            for j in range(len(state[0])):
   130      1548         0.6ms            .      15.2%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   131       764         0.2ms            .       5.5%                    value = state[i][j]
   132       764         0.2ms            .       5.2%                    row, col = goal_dict[value]
   133       764         0.3ms            .       7.6%                    distance += abs(row - i) + abs(col - j)
   134
   135       172             .            .       1.2%        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
   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: 0.023s
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      0.2%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queu
   185
   186         1             .            .         .        reached = {str(initialProblem): initialNode}           # reached←a lookup table,
   187
   188       104         0.1ms            .      0.2%        while not frontier.empty():                            # while not IS-EMPTY(front
   189       104         0.1ms            .      0.2%            node = frontier.get()                              # node←POP(frontier)
   190
   191       104             .            .      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       381        14.0ms            .     61.8%            for child in expand(node[1]):                      # for each child in EXPAND
```

```
   196                                                   # s←child.STATE
   197       278        0.3ms            .      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       278        0.5ms            .      2.2%      if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   201       171        0.2ms            .      1.0%          reached[str(s.data)] = s                    # reached[s]←child
   202       171        7.4ms            .     32.6%          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            .            .      0.5%       pathCost = 0
   210         1            .            .         .       stateSequence = []
   211         1            .            .         .       actionSequence = []
   212
   213        38            .            .      5.8%       while solution.prev != None:
   214        37            .            .      6.9%           stateSequence.insert(0, solution.data)
   215        37            .            .      6.9%           actionSequence.insert(0, solution.move)
   216        37            .            .      3.7%           solution = solution.prev
   217        37            .            .      4.8%           pathCost += 1
   218
   219         1            .            .      2.1%       print('Action sequence:')
   220         1        0.1ms        0.1ms     26.5%       print(*actionSequence, sep='\n')
   221
   222         1            .            .      1.6%       print('\nState sequence:')
   223         1        0.1ms        0.1ms     39.7%       print(*stateSequence, sep='\n')
   224
   225         1            .            .      1.6%       print('\nPath cost:', pathCost)
```