```
Total time: 0.059s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: expand at line 48

Line #      Hits         Time     Per Hit    % Time     Line Contents
=====================================================================
    48                                                  @cpu
    49                                                  def expand(board):
    50      1676        0.5ms           .      0.8%         for i in range(len(board.data)):
    51      4153        1.2ms           .      2.1%             for j in range(len(board.data[i])):
    52      3315        1.0ms           .      1.7%                 if board.data[i][j] == '*':
    53       419        0.1ms           .      0.2%                     location = [i,j];
    54       419        0.1ms           .      0.1%                     break
    55
    56       419        0.1ms           .      0.2%         actions = []
    57      1510        6.7ms           .     11.3%         for move in possible_actions(constants.board, location):
    58      1091       49.5ms           .     83.4%             actions.append([result(location, move, board.data) , move])
    59
    60       419        0.1ms           .      0.1%         return actions
```

```
Total time: 0.003s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: possible_actions at line 62

Line #      Hits         Time     Per Hit    % Time     Line Contents
=====================================================================
    62                                                  @cpu
    63                                                  def possible_actions(board, location):
    64       419        0.1ms           .      4.6%         actions = ["RIGHT","LEFT","UP","DOWN"]
    65       419        0.2ms           .      5.4%         actionstopeform = []
    66
    67      2095        0.6ms           .     18.1%         for x in actions:
    68                                                          # for moving right
    69      1676        0.4ms           .     13.9%             if x == "RIGHT":
    70       419        0.2ms           .      5.4%                 if location[1]+1 < len(board):
    71       291        0.1ms           .      4.5%                     actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73      1257        0.3ms           .     10.8%             elif x == "LEFT":
    74       419        0.1ms           .      4.5%                 if location[1]-1 >= 0:
    75       254        0.1ms           .      3.8%                     actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77       838        0.2ms           .      6.3%             elif x == "UP":
    78       419        0.1ms           .      4.5%                 if location[0]-1 >= 0:
    79       253        0.1ms           .      3.3%                     actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81       419        0.1ms           .      3.2%             elif x == "DOWN":
    82       419        0.1ms           .      4.6%                 if location[0]+1 < len(board):
    83       293        0.1ms           .      3.9%                     actionstopeform.append([x,location[0]+1,location[1]])
    84
    85       419        0.1ms           .      3.3%         return actionstopeform
```

```
Total time: 0.047s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: result at line 87

Line #      Hits         Time     Per Hit    % Time     Line Contents
=====================================================================
    87                                                  @cpu
    88                                                  def result(location,action,board):
    89                                                      # copy of a board so that we can modify it
    90      1091       38.4ms           .     82.0%         newBoard = copy.deepcopy(board)
    91      1091        2.8ms           .      5.9%         temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92      1091        2.7ms           .      5.8%         newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93      1091        2.7ms           .      5.7%         newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95      1091        0.3ms           .      0.6%         return newBoard
```

```
Total time: 0.026s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: linear_conflict at line 178

Line #      Hits         Time     Per Hit    % Time     Line Contents
=====================================================================
   178                                                  @cpu
   179                                                  def linear_conflict(board, goal):
   180       664        0.3ms           .      1.1%         n = len(board)
   181       664        0.2ms           .      0.9%         linear_conflicts = 0
   182
   183                                                      # Find the linear conflicts in rows
   184      2656        1.2ms           .      4.4%         for i in range(n):
   185      1992        0.8ms           .      3.1%             row = board[i]
   186      1992        0.8ms           .      3.2%             goal_row = goal[i]
   187      7968        3.4ms           .     13.2%             for j in range(n):
   188      5976        3.0ms           .     11.6%                 if row[j] != '*' and row[j] in goal_row:
   189      2461        1.2ms           .      4.7%                     for k in range(j+1, n):
   190      1204        0.7ms           .      2.7%                         if row[k] != '*' and row[k] in goal_row and goal_row.index(row[j])
   191        90            .           .      0.2%                             linear_conflicts += 2
   192
   193                                                      # Find the linear conflicts in columns
   194      2656        1.1ms           .      4.4%         for j in range(n):
   195      1992        2.3ms           .      8.8%             column = [board[i][j] for i in range(n)]
   196      1992        2.2ms           .      8.5%             goal_column = [goal[i][j] for i in range(n)]
   197      7968        3.5ms           .     13.3%             for i in range(n):
   198      5976        3.0ms           .     11.5%                 if column[i] != '*' and column[i] in goal_column:
   199      2516        1.2ms           .      4.8%                     for k in range(i+1, n):
```

```
   200      1200        0.7ms            .      2.6%                        if column[k] != '*' and column[k] in goal_column and goal_column.i
   201        81            .            .      0.1%                            linear_conflicts += 2
   202
   203       664        0.2ms            .      0.9%        return linear_conflicts


Total time: 0.022s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: manhattan at line 205

Line #      Hits         Time      Per Hit    % Time    Line Contents
=====================================================================
   205                                                  @cpu
   206                                                  def manhattan(state):
   207       664        0.2ms            .      0.9%        state = state.data
   208       664        0.2ms            .      1.0%        goal_state = constants.goalBoard
   209       664        0.2ms            .      0.8%        distance = 0
   210
   211                                                      # Create a dictionary that maps each value to its position in the goal state
   212       664        0.2ms            .      0.8%        goal_dict = {}
   213      2656        0.9ms            .      4.3%        for i in range(len(goal_state)):
   214      7968        2.4ms            .     11.2%            for j in range(len(goal_state[0])):
   215      5976        1.8ms            .      8.4%                if goal_state[i][j] != '*':
   216      5312        1.9ms            .      8.6%                    goal_dict[goal_state[i][j]] = (i, j)
   217
   218                                                      # Calculate Manhattan distance
   219      2656        0.8ms            .      3.6%        for i in range(len(state)):
   220      7968        2.3ms            .     10.7%            for j in range(len(state[0])):
   221      5976        2.5ms            .     11.6%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   222      4791        1.9ms            .      8.7%                    value = state[i][j]
   223      4791        1.3ms            .      5.9%                    row, col = goal_dict[value]
   224      4791        5.0ms            .     22.8%                    distance += abs(row - i) + abs(col - j)
   225
   226       664        0.2ms            .      0.8%        return distance


Total time: 0.085s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: f at line 228

Line #      Hits         Time      Per Hit    % Time    Line Contents
=====================================================================
   228                                                  @cpu
   229                                                  def f(board):
   230                                                          """ Heuristic Function to calculate hueristic value f(x) = h(x) + g(x) """
   231       664       35.7ms        0.1ms     42.0%            manhattan_distance = manhattan(board)
   232                                                          # Add linear conflict distance
   233       664       49.1ms        0.1ms     57.8%            manhattan_distance += linear_conflict(board.data, constants.goalBoard)
   234       664        0.2ms            .      0.2%            return manhattan_distance + board.depth


Total time: 0.164s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: a_star at line 236

Line #      Hits         Time      Per Hit    % Time    Line Contents
=====================================================================
   236                                                  @cpu
   237                                                  def a_star(initialProblem, f):
   238         1            .            .         .        initialNode = Node(data = initialProblem)   # node←NODE(STATE=problem.INITIAL)
   239         1            .            .         .        frontier = PriorityQueue()
   240         1        0.1ms        0.1ms      0.1%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queue
   241
   242         1            .            .         .        reached = {str(initialProblem): initialNode}           # reached←a lookup table, w
   243
   244       420        0.3ms            .      0.2%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   245       420        0.3ms            .      0.2%            node = frontier.get()                              # node←POP(frontier)
   246
   247       420        0.2ms            .      0.1%            if constants.goalBoard == node[1].data:            # if problem.IS-GOAL(node.S
   248         1            .            .         .                print('Max queue size:', frontier.getSize())
   249         1            .            .         .                return node[1]                                 # then return node
   250
   251      1510       63.3ms            .     38.5%            for child in expand(node[1]):               # for each child in EXPAND(problem
   252                                                                  # s←child.STATE
   253      1091        1.5ms            .      0.9%                s =  Node( data = child[0], depth = node[1].depth + 1, move = child[1], pr
   254
   255                                                                  # if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
   256      1091        2.5ms            .      1.5%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   257       663        0.9ms            .      0.6%                    reached[str(s.data)] = s                    # reached[s]←child
   258       663       95.1ms        0.1ms     57.8%                    frontier.append((f(s) ,s))                  # add child to frontier
   259
   260                                                          return constants.failure                              # return failure


Total time: 0.000s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: printStatistics at line 262

Line #      Hits         Time      Per Hit    % Time    Line Contents
=====================================================================
   262                                                  @cpu
   263                                                  def printStatistics(solution):
   264         1            .            .         .        pathCost = 0
   265         1            .            .         .        stateSequence = []
   266         1            .            .         .        actionSequence = []
   267
   268        29            .            .      5.4%        while solution.prev != None:
   269        28            .            .      9.0%            stateSequence.insert(0, solution.data)
```

```
    270        28            .            .      6.0%           actionSequence.insert(0, solution.move)
    271        28            .            .      3.6%           solution = solution.prev
    272        28            .            .      6.0%           pathCost += 1
    273
    274         1            .            .      2.4%       print('Action sequence:')
    275         1        0.1ms        0.1ms     31.1%       print(*actionSequence, sep='\n')
    276
    277         1            .            .      1.8%       print('\nState sequence:')
    278         1        0.1ms        0.1ms     32.9%       print(*stateSequence, sep='\n')
    279
    280         1            .            .      1.8%       print('\nPath cost:', pathCost)
```