```
Total time: 0.001s
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        28            .            .      0.5%        for i in range(len(board.data)):
    51        66            .            .      1.5%            for j in range(len(board.data[i])):
    52        52            .            .      1.9%                if board.data[i][j] == '*':
    53         7            .            .      0.1%                    location = [i,j];
    54         7            .            .      0.2%                    break
    55
    56         7            .            .      0.3%        actions = []
    57        28       0.1ms            .     10.2%        for move in possible_actions(constants.board, location):
    58        21       0.9ms            .     85.1%            actions.append([result(location, move, board.data) , move])
    59
    60         7            .            .      0.3%        return actions


Total time: 0.000s
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         7            .            .      3.9%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65         7            .            .      7.8%        actionstopeform = []
    66
    67        35            .            .      3.9%        for x in actions:
    68                                                          # for moving right
    69        28            .            .     21.6%            if x == "RIGHT":
    70         7            .            .      2.0%                if location[1]+1 < len(board):
    71         7            .            .      2.0%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73        21            .            .      3.9%            elif x == "LEFT":
    74         7            .            .      3.9%                if location[1]-1 >= 0:
    75         3            .            .      3.9%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77        14            .            .      9.8%            elif x == "UP":
    78         7            .            .      3.9%                if location[0]-1 >= 0:
    79         6            .            .      7.8%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81         7            .            .      3.9%            elif x == "DOWN":
    82         7            .            .      5.9%                if location[0]+1 < len(board):
    83         5            .            .      7.8%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85         7            .            .      7.8%        return actionstopeform


Total time: 0.001s
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        21       0.7ms            .     83.1%        newBoard = copy.deepcopy(board)
    91        21            .            .      5.6%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92        21            .            .      5.6%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93        21            .            .      5.3%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95        21            .            .      0.5%        return newBoard


Total time: 0.000s
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        16            .            .      1.1%        state = state.data
   208        16            .            .      1.1%        goal_state = constants.goalBoard
   209        16            .            .      1.1%        distance = 0
   210
   211                                                      # Create a dictionary that maps each value to its position in the goal state
   212        16            .            .      1.4%        goal_dict = {}
   213        64            .            .      5.8%        for i in range(len(goal_state)):
   214       192       0.1ms            .     14.8%            for j in range(len(goal_state[0])):
   215       144            .            .     12.1%                if goal_state[i][j] != '*':
   216       128            .            .     12.1%                    goal_dict[goal_state[i][j]] = (i, j)
   217
   218                                                      # Calculate Manhattan distance
   219        64            .            .      5.2%        for i in range(len(state)):
   220       192       0.1ms            .     15.9%            for j in range(len(state[0])):
   221       144       0.1ms            .     15.9%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   222        51            .            .      3.6%                    value = state[i][j]
   223        51            .            .      3.3%                    row, col = goal_dict[value]
   224        51            .            .      6.0%                    distance += abs(row - i) + abs(col - j)
   225
```

```
   226        16            .            .      0.8%        return distance
```

```
Total time: 0.002s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: a_star at line 235

Line #      Hits         Time      Per Hit    % Time    Line Contents
==============================================================
   235                                                  @cpu
   236                                                  def a_star(initialProblem, f):
   237         1            .            .      0.1%        initialNode = Node(data = initialProblem)   # node←NODE(STATE=problem.INITIAL)
   238         1            .            .      0.1%        frontier = PriorityQueue()
   239         1            .            .      2.5%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queue
   240
   241         1            .            .      0.2%        reached = {str(initialProblem): initialNode}           # reached←a lookup table, w
   242
   243         8            .            .      0.3%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   244         8            .            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   245
   246         8            .            .      0.2%            if constants.goalBoard == node[1].data:            # if problem.IS-GOAL(node.S
   247         1            .            .      1.1%                print('Max queue size:', frontier.getSize())
   248         1            .            .         .                return node[1]                                 # then return node
   249
   250        28        1.2ms            .     59.3%            for child in expand(node[1]):               # for each child in EXPAND(problem
   251                                                              # s←child.STATE
   252        21            .            .      1.4%                s =  Node( data = child[0], depth = node[1].depth + 1, move = child[1], pr
   253
   254                                                              # if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
   255        21            .            .      2.2%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   256        15            .            .      1.1%                    reached[str(s.data)] = s                    # reached[s]←child
   257        15        0.6ms            .     31.4%                    frontier.append((f(s) ,s))                  # add child to frontier
   258
   259                                                  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 261

Line #      Hits         Time      Per Hit    % Time    Line Contents
==============================================================
   261                                                  @cpu
   262                                                  def printStatistics(solution):
   263         1            .            .         .        pathCost = 0
   264         1            .            .         .        stateSequence = []
   265         1            .            .      2.2%        actionSequence = []
   266
   267         7            .            .      6.7%        while solution.prev != None:
   268         6            .            .      2.2%            stateSequence.insert(0, solution.data)
   269         6            .            .      6.7%            actionSequence.insert(0, solution.move)
   270         6            .            .      6.7%            solution = solution.prev
   271         6            .            .      4.4%            pathCost += 1
   272
   273         1            .            .      4.4%        print('Action sequence:')
   274         1            .            .     26.7%        print(*actionSequence, sep='\n')
   275
   276         1            .            .      6.7%        print('\nState sequence:')
   277         1            .            .     26.7%        print(*stateSequence, sep='\n')
   278
   279         1            .            .      6.7%        print('\nPath cost:', pathCost)
```