```
Total time: 0.048s
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      1444        0.4ms            .      0.8%        for i in range(len(board.data)):
    51      3625        1.0ms            .      2.1%            for j in range(len(board.data[i])):
    52      2903        0.8ms            .      1.7%                if board.data[i][j] == '*':
    53       361        0.1ms            .      0.2%                    location = [i,j];
    54       361        0.1ms            .      0.1%                    break
    55
    56       361        0.1ms            .      0.2%        actions = []
    57      1349        5.0ms            .     10.5%        for move in possible_actions(constants.board, location):
    58       988       40.1ms            .     84.2%            actions.append([result(location, move, board.data) , move])
    59
    60       361        0.1ms            .      0.2%        return actions


Total time: 0.002s
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       361        0.1ms            .      3.7%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65       361        0.1ms            .      3.5%        actionstopeform = []
    66
    67      1805        0.4ms            .     16.3%        for x in actions:
    68                                                          # for moving right
    69      1444        0.3ms            .     13.9%            if x == "RIGHT":
    70       361        0.1ms            .      5.1%                if location[1]+1 < len(board):
    71       239        0.1ms            .      4.4%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73      1083        0.3ms            .     11.4%            elif x == "LEFT":
    74       361        0.1ms            .      5.0%                if location[1]-1 >= 0:
    75       254        0.1ms            .      4.1%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77       722        0.2ms            .      7.3%            elif x == "UP":
    78       361        0.1ms            .      4.6%                if location[0]-1 >= 0:
    79       256        0.1ms            .      3.4%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81       361        0.1ms            .      3.9%            elif x == "DOWN":
    82       361        0.1ms            .      4.6%                if location[0]+1 < len(board):
    83       239        0.1ms            .      4.6%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85       361        0.1ms            .      4.0%        return actionstopeform


Total time: 0.038s
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       988       31.0ms            .     81.7%        newBoard = copy.deepcopy(board)
    91       988        2.3ms            .      6.1%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92       988        2.2ms            .      5.9%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93       988        2.2ms            .      5.7%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95       988        0.2ms            .      0.6%        return newBoard


Total time: 0.014s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: manhattan at line 204

Line #      Hits         Time      Per Hit    % Time    Line Contents
==============================================================
   204                                                  @cpu
   205                                                  def manhattan(state):
   206       602        0.2ms            .      1.4%        state = state.data
   207       602        0.2ms            .      1.2%        goal_state = constants.goalBoard
   208       602        0.1ms            .      1.0%        distance = 0
   209
   210                                                      # Create a dictionary that maps each value to its position in the goal state
   211       602        0.1ms            .      0.9%        goal_dict = {}
   212      2408        0.7ms            .      5.1%        for i in range(len(goal_state)):
   213      7224        2.0ms            .     13.9%            for j in range(len(goal_state[0])):
   214      5418        1.5ms            .     10.5%                if goal_state[i][j] != '*':
   215      4816        1.5ms            .     10.8%                    goal_dict[goal_state[i][j]] = (i, j)
   216
   217                                                      # Calculate Manhattan distance
   218      2408        0.7ms            .      5.0%        for i in range(len(state)):
   219      7224        1.9ms            .     13.6%            for j in range(len(state[0])):
   220      5418        2.2ms            .     15.3%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   221      3214        0.9ms            .      6.1%                    value = state[i][j]
   222      3214        0.8ms            .      5.8%                    row, col = goal_dict[value]
   223      3214        1.2ms            .      8.5%                    distance += abs(row - i) + abs(col - j)
   224
```

```
   225       602        0.1ms            .      1.1%        return distance
```

```
Total time: 0.086s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: a_star at line 234

Line #      Hits         Time      Per Hit    % Time    Line Contents
==============================================================
   234                                                  @cpu
   235                                                  def a_star(initialProblem, f):
   236         1            .            .         .        initialNode = Node(data = initialProblem)   # node←NODE(STATE=problem.INITIAL)
   237         1            .            .         .        frontier = PriorityQueue()
   238         1        0.1ms        0.1ms      0.1%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queue
   239
   240         1            .            .         .        reached = {str(initialProblem): initialNode}           # reached←a lookup table, w
   241
   242       362        0.2ms            .      0.3%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   243       362        0.2ms            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   244
   245       362        0.2ms            .      0.2%            if constants.goalBoard == node[1].data:            # if problem.IS-GOAL(node.S
   246         1            .            .         .                print('Max queue size:', frontier.getSize())
   247         1            .            .         .                return node[1]                                 # then return node
   248
   249      1349       50.7ms            .     59.0%            for child in expand(node[1]):               # for each child in EXPAND(problem
   250                                                              # s←child.STATE
   251       988        1.3ms            .      1.5%                s =  Node( data = child[0], depth = node[1].depth + 1, move = child[1], pr
   252
   253                                                              # if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
   254       988        1.9ms            .      2.2%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   255       601        0.8ms            .      0.9%                    reached[str(s.data)] = s                    # reached[s]←child
   256       601       30.5ms        0.1ms     35.5%                    frontier.append((f(s) ,s))                  # add child to frontier
   257
   258                                                  return constants.failure                              # return failure
```

```
Total time: 0.001s
File: /Users/rishabhjain/Documents/Masters/SEM 2/Aritificial Intelligence/Program/Program 1/program_1.py
Function: printStatistics at line 260

Line #      Hits         Time      Per Hit    % Time    Line Contents
==============================================================
   260                                                  @cpu
   261                                                  def printStatistics(solution):
   262         1            .            .      0.1%        pathCost = 0
   263         1            .            .         .        stateSequence = []
   264         1            .            .         .        actionSequence = []
   265
   266        47            .            .      1.3%        while solution.prev != None:
   267        46            .            .      1.8%            stateSequence.insert(0, solution.data)
   268        46            .            .      1.9%            actionSequence.insert(0, solution.move)
   269        46            .            .      1.5%            solution = solution.prev
   270        46            .            .      1.1%            pathCost += 1
   271
   272         1            .            .      0.5%        print('Action sequence:')
   273         1        0.1ms        0.1ms      6.5%        print(*actionSequence, sep='\n')
   274
   275         1            .            .      0.2%        print('\nState sequence:')
   276         1        0.8ms        0.8ms     83.6%        print(*stateSequence, sep='\n')
   277
   278         1            .            .      1.6%        print('\nPath cost:', pathCost)
```