```
Total time: 0.051s
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      1540        0.4ms            .      0.8%        for i in range(len(board.data)):
    51      3867        1.1ms            .      2.2%            for j in range(len(board.data[i])):
    52      3097        0.8ms            .      1.7%                if board.data[i][j] == '*':
    53       385        0.1ms            .      0.2%                    location = [i,j];
    54       385        0.1ms            .      0.2%                    break
    55
    56       385        0.1ms            .      0.2%        actions = []
    57      1427        5.4ms            .     10.6%        for move in possible_actions(constants.board, location):
    58      1042       42.9ms            .     84.1%            actions.append([result(location, move, board.data) , move])
    59
    60       385        0.1ms            .      0.2%        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       385        0.1ms            .      4.5%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65       385        0.1ms            .      2.9%        actionstopeform = []
    66
    67      1925        0.5ms            .     17.6%        for x in actions:
    68                                                          # for moving right
    69      1540        0.4ms            .     14.7%            if x == "RIGHT":
    70       385        0.1ms            .      5.5%                if location[1]+1 < len(board):
    71       252        0.1ms            .      3.5%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73      1155        0.3ms            .     10.5%            elif x == "LEFT":
    74       385        0.1ms            .      4.7%                if location[1]-1 >= 0:
    75       269        0.1ms            .      4.7%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77       770        0.2ms            .      6.9%            elif x == "UP":
    78       385        0.1ms            .      4.4%                if location[0]-1 >= 0:
    79       268        0.1ms            .      4.2%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81       385        0.1ms            .      3.5%            elif x == "DOWN":
    82       385        0.1ms            .      4.9%                if location[0]+1 < len(board):
    83       253        0.1ms            .      4.0%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85       385        0.1ms            .      3.4%        return actionstopeform


Total time: 0.041s
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      1042       33.1ms            .     81.6%        newBoard = copy.deepcopy(board)
    91      1042        2.5ms            .      6.3%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92      1042        2.4ms            .      5.9%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93      1042        2.3ms            .      5.7%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95      1042        0.2ms            .      0.5%        return newBoard


Total time: 0.015s
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       626        0.2ms            .      1.3%        state = state.data
   207       626        0.2ms            .      1.4%        goal_state = constants.goalBoard
   208       626        0.1ms            .      1.0%        distance = 0
   209
   210                                                      # Create a dictionary that maps each value to its position in the goal state
   211       626        0.1ms            .      1.0%        goal_dict = {}
   212      2504        0.7ms            .      4.9%        for i in range(len(goal_state)):
   213      7512        2.1ms            .     14.2%            for j in range(len(goal_state[0])):
   214      5634        1.6ms            .     10.7%                if goal_state[i][j] != '*':
   215      5008        1.5ms            .     10.6%                    goal_dict[goal_state[i][j]] = (i, j)
   216
   217                                                      # Calculate Manhattan distance
   218      2504        0.7ms            .      4.6%        for i in range(len(state)):
   219      7512        2.0ms            .     13.9%            for j in range(len(state[0])):
   220      5634        2.2ms            .     14.8%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   221      3278        0.9ms            .      5.9%                    value = state[i][j]
   222      3278        0.8ms            .      5.7%                    row, col = goal_dict[value]
   223      3278        1.3ms            .      8.8%                    distance += abs(row - i) + abs(col - j)
   224
```

```
   225       626        0.2ms            .      1.1%        return distance
```

```
Total time: 0.094s
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       386        0.2ms            .      0.3%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   243       386        0.3ms            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   244
   245       386        0.2ms            .      0.3%            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      1427       54.2ms            .     57.7%            for child in expand(node[1]):               # for each child in EXPAND(problem
   250                                                              # s←child.STATE
   251      1042        1.3ms            .      1.4%                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      1042        2.1ms            .      2.2%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   255       625        0.8ms            .      0.9%                    reached[str(s.data)] = s                   # reached[s]←child
   256       625       34.7ms        0.1ms     37.0%                    frontier.append((f(s) ,s))                 # add child to frontier
   257
   258                                                  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 260

Line #      Hits         Time      Per Hit    % Time    Line Contents
==============================================================
   260                                                  @cpu
   261                                                  def printStatistics(solution):
   262         1            .            .      0.6%        pathCost = 0
   263         1            .            .      0.6%        stateSequence = []
   264         1            .            .         .        actionSequence = []
   265
   266        29            .            .      5.7%        while solution.prev != None:
   267        28            .            .      7.6%            stateSequence.insert(0, solution.data)
   268        28            .            .      7.0%            actionSequence.insert(0, solution.move)
   269        28            .            .      5.1%            solution = solution.prev
   270        28            .            .      5.7%            pathCost += 1
   271
   272         1            .            .      1.9%        print('Action sequence:')
   273         1            .            .     27.4%        print(*actionSequence, sep='\n')
   274
   275         1            .            .      1.9%        print('\nState sequence:')
   276         1        0.1ms        0.1ms     34.4%        print(*stateSequence, sep='\n')
   277
   278         1            .            .      1.9%        print('\nPath cost:', pathCost)
```