```
Total time: 0.004s
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       136            .            .      1.0%        for i in range(len(board.data)):
    51       338        0.1ms            .      2.2%            for j in range(len(board.data[i])):
    52       270        0.1ms            .      1.8%                if board.data[i][j] == '*':
    53        34            .            .      0.2%                    location = [i,j];
    54        34            .            .      0.3%                    break
    55
    56        34            .            .      0.1%        actions = []
    57       119        0.5ms            .     11.0%        for move in possible_actions(constants.board, location):
    58        85        3.6ms            .     83.3%            actions.append([result(location, move, board.data) , move])
    59
    60        34            .            .      0.1%        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        34            .            .      4.6%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65        34            .            .      2.5%        actionstopeform = []
    66
    67       170        0.1ms            .     22.6%        for x in actions:
    68                                                          # for moving right
    69       136            .            .     11.3%            if x == "RIGHT":
    70        34            .            .      5.4%                if location[1]+1 < len(board):
    71        22            .            .      4.2%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73       102            .            .     11.7%            elif x == "LEFT":
    74        34            .            .      4.6%                if location[1]-1 >= 0:
    75        20            .            .      2.1%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77        68            .            .      7.5%            elif x == "UP":
    78        34            .            .      5.0%                if location[0]-1 >= 0:
    79        22            .            .      2.5%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81        34            .            .      3.8%            elif x == "DOWN":
    82        34            .            .      5.4%                if location[0]+1 < len(board):
    83        21            .            .      3.3%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85        34            .            .      3.3%        return actionstopeform


Total time: 0.003s
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        85        2.8ms            .     81.8%        newBoard = copy.deepcopy(board)
    91        85        0.2ms            .      6.2%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92        85        0.2ms            .      5.8%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93        85        0.2ms            .      5.6%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95        85            .            .      0.7%        return newBoard


Total time: 0.001s
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        53            .            .      1.3%        state = state.data
   207        53            .            .      1.4%        goal_state = constants.goalBoard
   208        53            .            .      0.9%        distance = 0
   209
   210                                                      # Create a dictionary that maps each value to its position in the goal state
   211        53            .            .      0.9%        goal_dict = {}
   212       212        0.1ms            .      5.5%        for i in range(len(goal_state)):
   213       636        0.2ms            .     15.0%            for j in range(len(goal_state[0])):
   214       477        0.1ms            .     10.6%                if goal_state[i][j] != '*':
   215       424        0.1ms            .     11.2%                    goal_dict[goal_state[i][j]] = (i, j)
   216
   217                                                      # Calculate Manhattan distance
   218       212        0.1ms            .      4.0%        for i in range(len(state)):
   219       636        0.2ms            .     13.2%            for j in range(len(state[0])):
   220       477        0.2ms            .     13.1%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   221       348        0.1ms            .      6.3%                    value = state[i][j]
   222       348        0.1ms            .      6.2%                    row, col = goal_dict[value]
   223       348        0.1ms            .      9.5%                    distance += abs(row - i) + abs(col - j)
   224
```

```
   225        53            .            .      0.9%        return distance
```

```
Total time: 0.007s
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      0.8%        frontier.append((f(initialNode), initialNode))         # frontier←a priority queue
   239
   240         1            .            .      0.1%        reached = {str(initialProblem): initialNode}           # reached←a lookup table, w
   241
   242        35            .            .      0.3%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   243        35            .            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   244
   245        35            .            .      0.1%            if constants.goalBoard == node[1].data:            # if problem.IS-GOAL(node.S
   246         1            .            .      0.3%                print('Max queue size:', frontier.getSize())
   247         1            .            .         .                return node[1]                                 # then return node
   248
   249       119        4.6ms            .     61.5%            for child in expand(node[1]):               # for each child in EXPAND(problem
   250                                                              # s←child.STATE
   251        85        0.1ms            .      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        85        0.2ms            .      2.1%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   255        52        0.1ms            .      1.0%                    reached[str(s.data)] = s                    # reached[s]←child
   256        52        2.4ms            .     32.1%                    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            .            .         .        pathCost = 0
   263         1            .            .         .        stateSequence = []
   264         1            .            .         .        actionSequence = []
   265
   266        29            .            .      2.7%        while solution.prev != None:
   267        28            .            .      2.0%            stateSequence.insert(0, solution.data)
   268        28            .            .      2.3%            actionSequence.insert(0, solution.move)
   269        28            .            .      1.7%            solution = solution.prev
   270        28            .            .      2.0%            pathCost += 1
   271
   272         1            .            .      0.7%        print('Action sequence:')
   273         1            .            .     13.7%        print(*actionSequence, sep='\n')
   274
   275         1            .            .      0.7%        print('\nState sequence:')
   276         1        0.2ms        0.2ms     71.7%        print(*stateSequence, sep='\n')
   277
   278         1            .            .      2.7%        print('\nPath cost:', pathCost)
```