```
Total time: 0.057s
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      1660        0.5ms            .      0.9%        for i in range(len(board.data)):
    51      4136        1.2ms            .      2.1%            for j in range(len(board.data[i])):
    52      3306        1.0ms            .      1.8%                if board.data[i][j] == '*':
    53       415        0.1ms            .      0.2%                    location = [i,j];
    54       415        0.1ms            .      0.1%                    break
    55
    56       415        0.1ms            .      0.2%        actions = []
    57      1548        6.0ms            .     10.5%        for move in possible_actions(constants.board, location):
    58      1133       47.5ms            .     84.0%            actions.append([result(location, move, board.data) , move])
    59
    60       415        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       415        0.1ms            .      4.2%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65       415        0.1ms            .      3.5%        actionstopeform = []
    66
    67      2075        0.5ms            .     18.4%        for x in actions:
    68                                                          # for moving right
    69      1660        0.4ms            .     14.5%            if x == "RIGHT":
    70       415        0.2ms            .      5.1%                if location[1]+1 < len(board):
    71       292        0.1ms            .      3.9%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73      1245        0.3ms            .     11.6%            elif x == "LEFT":
    74       415        0.1ms            .      4.3%                if location[1]-1 >= 0:
    75       278        0.1ms            .      4.0%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77       830        0.2ms            .      7.1%            elif x == "UP":
    78       415        0.1ms            .      4.5%                if location[0]-1 >= 0:
    79       274        0.1ms            .      3.7%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81       415        0.1ms            .      3.6%            elif x == "DOWN":
    82       415        0.1ms            .      4.5%                if location[0]+1 < len(board):
    83       289        0.1ms            .      3.9%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85       415        0.1ms            .      3.3%        return actionstopeform


Total time: 0.045s
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      1133       36.7ms            .     81.7%        newBoard = copy.deepcopy(board)
    91      1133        2.7ms            .      6.1%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92      1133        2.7ms            .      5.9%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93      1133        2.6ms            .      5.7%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95      1133        0.2ms            .      0.5%        return newBoard


Total time: 0.029s
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       683        0.3ms            .      1.1%        n = len(board)
   181       683        0.3ms            .      0.9%        linear_conflicts = 0
   182
   183                                                      # Find the linear conflicts in rows
   184      2732        1.2ms            .      4.0%        for i in range(n):
   185      2049        0.8ms            .      2.8%            row = board[i]
   186      2049        0.8ms            .      2.8%            goal_row = goal[i]
   187      8196        3.4ms            .     11.7%            for j in range(n):
   188      6147        2.9ms            .     10.0%                if row[j] != '*' and row[j] in goal_row:
   189      4087        1.9ms            .      6.6%                    for k in range(j+1, n):
   190      1783        1.0ms            .      3.5%                        if row[k] != '*' and row[k] in goal_row and goal_row.index(row[j])
   191        71            .            .      0.1%                            linear_conflicts += 2
   192
   193                                                      # Find the linear conflicts in columns
   194      2732        1.2ms            .      4.0%        for j in range(n):
   195      2049        2.4ms            .      8.1%            column = [board[i][j] for i in range(n)]
   196      2049        2.1ms            .      7.3%            goal_column = [goal[i][j] for i in range(n)]
   197      8196        3.4ms            .     11.7%            for i in range(n):
   198      6147        2.9ms            .     10.1%                if column[i] != '*' and column[i] in goal_column:
   199      5636        2.6ms            .      8.8%                    for k in range(i+1, n):
```

```
   200      2773        1.6ms            .      5.4%                        if column[k] != '*' and column[k] in goal_column and goal_column.i
   201       177        0.1ms            .      0.3%                            linear_conflicts += 2
   202
   203       683        0.3ms            .      0.9%        return linear_conflicts


Total time: 0.017s
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       683        0.2ms            .      1.2%        state = state.data
   208       683        0.2ms            .      1.3%        goal_state = constants.goalBoard
   209       683        0.2ms            .      1.0%        distance = 0
   210
   211                                               # Create a dictionary that maps each value to its position in the goal state
   212       683        0.2ms            .      1.0%        goal_dict = {}
   213      2732        0.8ms            .      4.9%        for i in range(len(goal_state)):
   214      8196        2.3ms            .     13.6%            for j in range(len(goal_state[0])):
   215      6147        1.8ms            .     10.7%                if goal_state[i][j] != '*':
   216      5464        1.8ms            .     10.5%                    goal_dict[goal_state[i][j]] = (i, j)
   217
   218                                               # Calculate Manhattan distance
   219      2732        0.7ms            .      4.4%        for i in range(len(state)):
   220      8196        2.3ms            .     13.9%            for j in range(len(state[0])):
   221      6147        2.4ms            .     14.5%                if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   222      3985        1.1ms            .      6.4%                    value = state[i][j]
   223      3985        1.0ms            .      6.2%                    row, col = goal_dict[value]
   224      3985        1.6ms            .      9.4%                    distance += abs(row - i) + abs(col - j)
   225
   226       683        0.2ms            .      1.0%        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       683       29.9ms            .     35.3%            manhattan_distance = manhattan(board)
   232                                                   # Add linear conflict distance
   233       683       54.8ms        0.1ms     64.5%            manhattan_distance += linear_conflict(board.data, constants.goalBoard)
   234       683        0.2ms            .      0.2%            return manhattan_distance + board.depth


Total time: 0.165s
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       416        0.3ms            .      0.2%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   245       416        0.3ms            .      0.2%            node = frontier.get()                              # node←POP(frontier)
   246
   247       416        0.3ms            .      0.2%            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      1548       60.1ms            .     36.3%            for child in expand(node[1]):                # for each child in EXPAND(problem
   252                                                               # s←child.STATE
   253      1133        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      1133        2.3ms            .      1.4%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   257       682        0.9ms            .      0.6%                    reached[str(s.data)] = s                    # reached[s]←child
   258       682       99.6ms        0.1ms     60.2%                    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            .            .      0.8%        pathCost = 0
   265         1            .            .      0.8%        stateSequence = []
   266         1            .            .         .        actionSequence = []
   267
   268        22            .            .      5.5%        while solution.prev != None:
   269        21            .            .      5.5%            stateSequence.insert(0, solution.data)
```

```
270        21            .            .      7.1%            actionSequence.insert(0, solution.move)
271        21            .            .      4.7%            solution = solution.prev
272        21            .            .      5.5%            pathCost += 1
273
274         1            .            .      2.4%        print('Action sequence:')
275         1            .            .     28.3%        print(*actionSequence, sep='\n')
276
277         1            .            .      2.4%        print('\nState sequence:')
278         1            .            .     34.6%        print(*stateSequence, sep='\n')
279
280         1            .            .      2.4%        print('\nPath cost:', pathCost)
```