```
Total time: 0.097s
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      2968        0.8ms            .      0.9%        for i in range(len(board.data)):
    51      7411        2.0ms            .      2.1%            for j in range(len(board.data[i])):
    52      5927        1.6ms            .      1.7%                if board.data[i][j] == '*':
    53       742        0.2ms            .      0.2%                    location = [i,j];
    54       742        0.1ms            .      0.1%                    break
    55
    56       742        0.2ms            .      0.2%        actions = []
    57      2753       10.2ms            .     10.6%        for move in possible_actions(constants.board, location):
    58      2011       81.5ms            .     84.1%            actions.append([result(location, move, board.data) , move])
    59
    60       742        0.2ms            .      0.2%        return actions


Total time: 0.005s
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       742        0.2ms            .      4.6%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65       742        0.2ms            .      3.2%        actionstopeform = []
    66
    67      3710        0.9ms            .     18.0%        for x in actions:
    68                                                          # for moving right
    69      2968        0.7ms            .     14.3%            if x == "RIGHT":
    70       742        0.3ms            .      5.7%                if location[1]+1 < len(board):
    71       508        0.2ms            .      4.0%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                          # for moving left
    73      2226        0.5ms            .     10.6%            elif x == "LEFT":
    74       742        0.2ms            .      4.9%                if location[1]-1 >= 0:
    75       499        0.2ms            .      3.9%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                          # for moving up
    77      1484        0.3ms            .      6.9%            elif x == "UP":
    78       742        0.2ms            .      4.7%                if location[0]-1 >= 0:
    79       467        0.2ms            .      3.3%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                          # for moving down
    81       742        0.2ms            .      3.4%            elif x == "DOWN":
    82       742        0.3ms            .      5.2%                if location[0]+1 < len(board):
    83       537        0.2ms            .      3.9%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85       742        0.2ms            .      3.4%        return actionstopeform


Total time: 0.077s
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      2011       63.0ms            .     81.9%        newBoard = copy.deepcopy(board)
    91      2011        4.7ms            .      6.1%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92      2011        4.5ms            .      5.9%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93      2011        4.4ms            .      5.7%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                      # return new board after moving * - NIL to the new location
    95      2011        0.4ms            .      0.5%        return newBoard


Total time: 0.051s
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      1194        0.6ms            .      1.1%        n = len(board)
   181      1194        0.5ms            .      0.9%        linear_conflicts = 0
   182
   183                                                      # Find the linear conflicts in rows
   184      4776        2.0ms            .      4.0%        for i in range(n):
   185      3582        1.5ms            .      2.8%            row = board[i]
   186      3582        1.4ms            .      2.7%            goal_row = goal[i]
   187     14328        5.9ms            .     11.5%            for j in range(n):
   188     10746        5.1ms            .      9.9%                if row[j] != '*' and row[j] in goal_row:
   189      8941        4.2ms            .      8.1%                    for k in range(j+1, n):
   190      4387        2.6ms            .      5.1%                        if row[k] != '*' and row[k] in goal_row and goal_row.index(row[j])
   191       192        0.1ms            .      0.2%                            linear_conflicts += 2
   192
   193                                                      # Find the linear conflicts in columns
   194      4776        1.9ms            .      3.8%        for j in range(n):
   195      3582        3.8ms            .      7.4%            column = [board[i][j] for i in range(n)]
   196      3582        3.8ms            .      7.4%            goal_column = [goal[i][j] for i in range(n)]
   197     14328        5.9ms            .     11.5%            for i in range(n):
   198     10746        5.2ms            .     10.1%                if column[i] != '*' and column[i] in goal_column:
   199      8804        4.1ms            .      8.0%                    for k in range(i+1, n):
```

```
   200      4147        2.3ms            .      4.5%                      if column[k] != '*' and column[k] in goal_column and goal_column.i
   201       164        0.1ms            .      0.1%                          linear_conflicts += 2
   202
   203      1194        0.5ms            .      0.9%          return linear_conflicts


Total time: 0.030s
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      1194        0.4ms            .      1.3%          state = state.data
   208      1194        0.4ms            .      1.2%          goal_state = constants.goalBoard
   209      1194        0.3ms            .      1.1%          distance = 0
   210
   211                                                       # Create a dictionary that maps each value to its position in the goal state
   212      1194        0.3ms            .      1.1%          goal_dict = {}
   213      4776        1.4ms            .      4.8%          for i in range(len(goal_state)):
   214     14328        4.2ms            .     14.0%              for j in range(len(goal_state[0])):
   215     10746        3.3ms            .     10.8%                  if goal_state[i][j] != '*':
   216      9552        3.2ms            .     10.8%                      goal_dict[goal_state[i][j]] = (i, j)
   217
   218                                                       # Calculate Manhattan distance
   219      4776        1.4ms            .      4.6%          for i in range(len(state)):
   220     14328        4.1ms            .     13.6%              for j in range(len(state[0])):
   221     10746        4.3ms            .     14.4%                  if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
   222      6877        1.9ms            .      6.3%                      value = state[i][j]
   223      6877        1.9ms            .      6.2%                      row, col = goal_dict[value]
   224      6877        2.7ms            .      9.1%                      distance += abs(row - i) + abs(col - j)
   225
   226      1194        0.3ms            .      1.0%          return distance


Total time: 0.153s
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      1194       56.2ms            .     36.6%          manhattan_distance = manhattan(board)
   232                                                       # Add linear conflict distance
   233      1194       96.9ms        0.1ms     63.2%          manhattan_distance += linear_conflict(board.data, constants.goalBoard)
   234      1194        0.3ms            .      0.2%          return manhattan_distance + board.depth


Total time: 0.307s
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         .          frontier.append((f(initialNode), initialNode))         # frontier←a priority queue
   241
   242         1            .            .         .          reached = {str(initialProblem): initialNode}           # reached←a lookup table, w
   243
   244       743        0.5ms            .      0.2%          while not frontier.empty():                            # while not IS-EMPTY(fronti
   245       743        0.5ms            .      0.2%              node = frontier.get()                              # node←POP(frontier)
   246
   247       743        0.5ms            .      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      2753      103.0ms            .     33.5%              for child in expand(node[1]):                # for each child in EXPAND(problem
   252                                                                 # s←child.STATE
   253      2011        2.5ms            .      0.8%                  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      2011        4.1ms            .      1.3%                  if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   257      1193        1.6ms            .      0.5%                      reached[str(s.data)] = s                 # reached[s]←child
   258      1193      194.1ms        0.2ms     63.2%                      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            .            .         .          stateSequence = []
   266         1            .            .         .          actionSequence = []
   267
   268        23            .            .      5.9%          while solution.prev != None:
   269        22            .            .      5.9%              stateSequence.insert(0, solution.data)
```

```
270        22            .            .      5.9%            actionSequence.insert(0, solution.move)
271        22            .            .      3.4%            solution = solution.prev
272        22            .            .      5.9%            pathCost += 1
273
274         1            .            .      5.0%        print('Action sequence:')
275         1            .            .     28.6%        print(*actionSequence, sep='\n')
276
277         1            .            .      2.5%        print('\nState sequence:')
278         1            .            .     34.5%        print(*stateSequence, sep='\n')
279
280         1            .            .      1.7%        print('\nPath cost:', pathCost)
```