```
Total time: 0.072s
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      2208         0.6ms          .        0.8%        for i in range(len(board.data)):
    51      5520         1.5ms          .        2.1%            for j in range(len(board.data[i])):
    52      4416         1.3ms          .        1.8%                if board.data[i][j] == '*':
    53       552         0.1ms          .        0.2%                    location = [i,j];
    54       552         0.1ms          .        0.1%                    break
    55
    56       552         0.1ms          .        0.2%        actions = []
    57      2003         7.8ms          .       10.7%        for move in possible_actions(constants.board, location):
    58      1451        60.7ms          .       83.9%            actions.append([result(location, move, board.data) , move])
    59
    60       552         0.1ms          .        0.2%        return actions


Total time: 0.004s
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       552         0.2ms          .        5.0%        actions = ["RIGHT","LEFT","UP","DOWN"]
    65       552         0.1ms          .        3.0%        actionstopeform = []
    66
    67      2760         0.7ms          .       18.1%        for x in actions:
    68                                                           # for moving right
    69      2208         0.5ms          .       13.0%            if x == "RIGHT":
    70       552         0.2ms          .        5.8%                if location[1]+1 < len(board):
    71       359         0.1ms          .        3.8%                    actionstopeform.append([x,location[0],location[1]+1])
    72                                                           # for moving left
    73      1656         0.4ms          .       11.5%            elif x == "LEFT":
    74       552         0.2ms          .        4.4%                if location[1]-1 >= 0:
    75       359         0.1ms          .        3.6%                    actionstopeform.append([x,location[0],location[1]-1])
    76                                                           # for moving up
    77      1104         0.3ms          .        7.0%            elif x == "UP":
    78       552         0.2ms          .        4.7%                if location[0]-1 >= 0:
    79       356         0.1ms          .        3.7%                    actionstopeform.append([x,location[0]-1,location[1]])
    80                                                           # for moving down
    81       552         0.1ms          .        3.9%            elif x == "DOWN":
    82       552         0.2ms          .        5.2%                if location[0]+1 < len(board):
    83       377         0.1ms          .        4.0%                    actionstopeform.append([x,location[0]+1,location[1]])
    84
    85       552         0.1ms          .        3.4%        return actionstopeform


Total time: 0.057s
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      1451        46.9ms          .       81.8%        newBoard = copy.deepcopy(board)
    91      1451         3.5ms          .        6.1%        temp = copy.deepcopy(newBoard[action[1]][action[2]])
    92      1451         3.4ms          .        5.9%        newBoard[action[1]][action[2]] = copy.deepcopy('*')
    93      1451         3.3ms          .        5.7%        newBoard[location[0]][location[1]]  = copy.deepcopy(temp)
    94                                                       # return new board after moving * - NIL to the new location
    95      1451         0.3ms          .        0.5%        return newBoard


Total time: 0.034s
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       842         0.3ms          .        1.0%        n = len(board)
   181       842         0.3ms          .        0.9%        linear_conflicts = 0
   182
   183                                                       # Find the linear conflicts in rows
   184      3368         1.3ms          .        3.9%        for i in range(n):
   185      2526         1.0ms          .        2.9%            row = board[i]
   186      2526         1.0ms          .        2.9%            goal_row = goal[i]
   187     10104         4.0ms          .       11.9%            for j in range(n):
   188      7578         3.5ms          .       10.5%                if row[j] != '*' and row[j] in goal_row:
   189      5313         2.5ms          .        7.5%                    for k in range(j+1, n):
   190      2775         1.5ms          .        4.5%                        if row[k] != '*' and row[k] in goal_row and goal_row.index(row[j])
   191        69            .           .        0.1%                            linear_conflicts += 2
   192
   193                                                       # Find the linear conflicts in columns
   194      3368         1.4ms          .        4.1%        for j in range(n):
   195      2526         2.7ms          .        8.0%            column = [board[i][j] for i in range(n)]
   196      2526         2.5ms          .        7.5%            goal_column = [goal[i][j] for i in range(n)]
   197     10104         4.1ms          .       12.2%            for i in range(n):
   198      7578         3.6ms          .       10.6%                if column[i] != '*' and column[i] in goal_column:
   199      4766         2.3ms          .        6.7%                    for k in range(i+1, n):
```

```
200      2268      1.3ms          .       3.7%              if column[k] != '*' and column[k] in goal_column and goal_column.i
201       222      0.1ms          .       0.2%                  linear_conflicts += 2
202
203       842      0.3ms          .       0.9%          return linear_conflicts
```

Total time: 0.022s
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	842	0.3ms	.	1.2%	state = state.data
208	842	0.2ms	.	1.1%	goal_state = constants.goalBoard
209	842	0.2ms	.	1.0%	distance = 0
210					
211					# Create a dictionary that maps each value to its position in the goal state
212	842	0.2ms	.	0.9%	goal_dict = {}
213	3368	1.1ms	.	4.9%	for i in range(len(goal_state)):
214	10104	2.9ms	.	13.4%	for j in range(len(goal_state[0])):
215	7578	2.2ms	.	10.1%	if goal_state[i][j] != '*':
216	6736	2.3ms	.	10.6%	goal_dict[goal_state[i][j]] = (i, j)
217					
218					# Calculate Manhattan distance
219	3368	0.9ms	.	4.3%	for i in range(len(state)):
220	10104	2.8ms	.	12.8%	for j in range(len(state[0])):
221	7578	3.1ms	.	14.1%	if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
222	5646	1.6ms	.	7.2%	value = state[i][j]
223	5646	1.5ms	.	6.9%	row, col = goal_dict[value]
224	5646	2.3ms	.	10.4%	distance += abs(row - i) + abs(col - j)
225					
226	842	0.2ms	.	1.0%	return distance

Total time: 0.105s
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	842	41.4ms	.	39.3%	manhattan_distance = manhattan(board)
232					# Add linear conflict distance
233	842	63.8ms	0.1ms	60.5%	manhattan_distance += linear_conflict(board.data, constants.goalBoard)
234	842	0.2ms	.	0.2%	return manhattan_distance + board.depth

Total time: 0.207s
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	553	0.4ms	.	0.2%	while not frontier.empty(): # while not IS-EMPTY(fronti
245	553	0.4ms	.	0.2%	node = frontier.get() # node←POP(frontier)
246					
247	553	0.4ms	.	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	2003	77.0ms	.	37.2%	for child in expand(node[1]): # for each child in EXPAND(problem
252					# s←child.STATE
253	1451	1.8ms	.	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	1451	3.0ms	.	1.4%	if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
257	841	1.2ms	.	0.6%	reached[str(s.data)] = s # reached[s]←child
258	841	122.7ms	0.1ms	59.3%	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	.	.	.	pathCost = 0
265	1	.	.	.	stateSequence = []
266	1	.	.	.	actionSequence = []
267					
268	25	.	.	5.8%	while solution.prev != None:
269	24	.	.	5.8%	stateSequence.insert(0, solution.data)

```
270        24            .            .       5.8%            actionSequence.insert(0, solution.move)
271        24            .            .       4.3%            solution = solution.prev
272        24            .            .       5.8%            pathCost += 1
273
274         1            .            .       2.9%        print('Action sequence:')
275         1            .            .      30.2%        print(*actionSequence, sep='\n')
276
277         1            .            .       2.2%        print('\nState sequence:')
278         1            .            .      35.3%        print(*stateSequence, sep='\n')
279
280         1            .            .       2.2%        print('\nPath cost:', pathCost)
```