Total time: 0.050s
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	1504	0.4ms	.	0.9%	for i in range(len(board.data)):
51	3771	1.1ms	.	2.1%	for j in range(len(board.data[i])):
52	3019	0.8ms	.	1.7%	if board.data[i][j] == '*':
53	376	0.1ms	.	0.2%	location = [i,j];
54	376	0.1ms	.	0.2%	break
55					
56	376	0.1ms	.	0.1%	actions = []
57	1413	5.3ms	.	10.5%	for move in possible_actions(constants.board, location):
58	1037	42.4ms	.	84.3%	actions.append([result(location, move, board.data) , move])
59					
60	376	0.1ms	.	0.1%	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	376	0.1ms	.	4.4%	actions = ["RIGHT","LEFT","UP","DOWN"]
65	376	0.1ms	.	3.7%	actionstopeform = []
66					
67	1880	0.5ms	.	18.1%	for x in actions:
68					# for moving right
69	1504	0.4ms	.	14.2%	if x == "RIGHT":
70	376	0.1ms	.	5.3%	if location[1]+1 < len(board):
71	254	0.1ms	.	4.1%	actionstopeform.append([x,location[0],location[1]+1])
72					# for moving left
73	1128	0.3ms	.	11.5%	elif x == "LEFT":
74	376	0.1ms	.	4.1%	if location[1]-1 >= 0:
75	265	0.1ms	.	4.3%	actionstopeform.append([x,location[0],location[1]-1])
76					# for moving up
77	752	0.2ms	.	6.7%	elif x == "UP":
78	376	0.1ms	.	5.2%	if location[0]-1 >= 0:
79	265	0.1ms	.	3.7%	actionstopeform.append([x,location[0]-1,location[1]])
80					# for moving down
81	376	0.1ms	.	2.9%	elif x == "DOWN":
82	376	0.1ms	.	4.5%	if location[0]+1 < len(board):
83	253	0.1ms	.	4.1%	actionstopeform.append([x,location[0]+1,location[1]])
84					
85	376	0.1ms	.	3.1%	return actionstopeform

Total time: 0.040s
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	1037	32.8ms	.	81.8%	newBoard = copy.deepcopy(board)
91	1037	2.4ms	.	6.1%	temp = copy.deepcopy(newBoard[action[1]][action[2]])
92	1037	2.4ms	.	6.0%	newBoard[action[1]][action[2]] = copy.deepcopy('*')
93	1037	2.3ms	.	5.7%	newBoard[location[0]][location[1]] = copy.deepcopy(temp)
94					# return new board after moving * - NIL to the new location
95	1037	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	634	0.2ms	.	1.2%	state = state.data
207	634	0.2ms	.	1.3%	goal_state = constants.goalBoard
208	634	0.1ms	.	1.0%	distance = 0
209					
210					# Create a dictionary that maps each value to its position in the goal state
211	634	0.2ms	.	1.0%	goal_dict = {}
212	2536	0.8ms	.	5.1%	for i in range(len(goal_state)):
213	7608	2.1ms	.	14.1%	for j in range(len(goal_state[0])):
214	5706	1.6ms	.	10.7%	if goal_state[i][j] != '*':
215	5072	1.6ms	.	10.9%	goal_dict[goal_state[i][j]] = (i, j)
216					
217					# Calculate Manhattan distance
218	2536	0.7ms	.	4.8%	for i in range(len(state)):
219	7608	2.1ms	.	14.0%	for j in range(len(state[0])):
220	5706	2.2ms	.	14.8%	if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
221	3294	0.9ms	.	6.0%	value = state[i][j]
222	3294	0.8ms	.	5.6%	row, col = goal_dict[value]
223	3294	1.3ms	.	8.5%	distance += abs(row - i) + abs(col - j)
224					

```
   225       634        0.1ms            .      1.0%        return distance
```

```
Total time: 0.091s
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       377        0.2ms            .      0.3%        while not frontier.empty():                            # while not IS-EMPTY(fronti
   243       377        0.2ms            .      0.3%            node = frontier.get()                              # node←POP(frontier)
   244
   245       377        0.2ms            .      0.2%            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      1413       53.4ms            .     58.9%            for child in expand(node[1]):               # for each child in EXPAND(problem
   250                                                              # s←child.STATE
   251      1037        1.3ms            .      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      1037        2.0ms            .      2.2%                if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
   255       633        0.8ms            .      0.9%                    reached[str(s.data)] = s                   # reached[s]←child
   256       633       32.4ms        0.1ms     35.7%                    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            .            .      0.4%        pathCost = 0
   263         1            .            .         .        stateSequence = []
   264         1            .            .         .        actionSequence = []
   265
   266        50            .            .      6.0%        while solution.prev != None:
   267        49            .            .      5.6%            stateSequence.insert(0, solution.data)
   268        49            .            .      6.4%            actionSequence.insert(0, solution.move)
   269        49            .            .      5.6%            solution = solution.prev
   270        49            .            .      5.6%            pathCost += 1
   271
   272         1            .            .      1.3%        print('Action sequence:')
   273         1        0.1ms        0.1ms     28.3%        print(*actionSequence, sep='\n')
   274
   275         1            .            .      0.9%        print('\nState sequence:')
   276         1        0.1ms        0.1ms     38.6%        print(*stateSequence, sep='\n')
   277
   278         1            .            .      1.3%        print('\nPath cost:', pathCost)
```