

Total time: 0.164s

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	4896	1.4ms	.	0.8%	for i in range(len(board.data)):
51	12306	3.4ms	.	2.1%	for j in range(len(board.data[i])):
52	9858	2.9ms	.	1.8%	if board.data[i][j] == '*':
53	1224	0.3ms	.	0.2%	location = [i,j];
54	1224	0.3ms	.	0.2%	break
55					
56	1224	0.3ms	.	0.2%	actions = []
57	4498	17.4ms	.	10.6%	for move in possible_actions(constants.board, location):
58	3274	137.6ms	.	84.0%	actions.append([result(location, move, board.data) , move])
59					
60	1224	0.3ms	.	0.2%	return actions

Total time: 0.008s

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	1224	0.4ms	.	4.7%	actions = ["RIGHT", "LEFT", "UP", "DOWN"]
65	1224	0.3ms	.	3.7%	actionstopeform = []
66					
67	6120	1.5ms	.	17.9%	for x in actions:
68					# for moving right
69	4896	1.2ms	.	14.1%	if x == "RIGHT":
70	1224	0.4ms	.	5.2%	if location[1]+1 < len(board):
71	776	0.3ms	.	4.2%	actionstopeform.append([x,location[0],location[1]+1])
72					# for moving left
73	3672	0.9ms	.	10.8%	elif x == "LEFT":
74	1224	0.4ms	.	4.6%	if location[1]-1 >= 0:
75	842	0.3ms	.	4.2%	actionstopeform.append([x,location[0],location[1]-1])
76					# for moving up
77	2448	0.6ms	.	6.8%	elif x == "UP":
78	1224	0.4ms	.	4.3%	if location[0]-1 >= 0:
79	806	0.3ms	.	3.7%	actionstopeform.append([x,location[0]-1,location[1]])
80					# for moving down
81	1224	0.3ms	.	3.4%	elif x == "DOWN":
82	1224	0.4ms	.	4.8%	if location[0]+1 < len(board):
83	850	0.3ms	.	4.1%	actionstopeform.append([x,location[0]+1,location[1]])
84					
85	1224	0.3ms	.	3.5%	return actionstopeform

Total time: 0.130s

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	3274	106.5ms	.	81.9%	newBoard = copy.deepcopy(board)
91	3274	7.9ms	.	6.1%	temp = copy.deepcopy(newBoard[action[1]][action[2]])
92	3274	7.6ms	.	5.8%	newBoard[action[1]][action[2]] = copy.deepcopy('*')
93	3274	7.4ms	.	5.7%	newBoard[location[0]][location[1]] = copy.deepcopy(temp)
94					# return new board after moving * - NIL to the new location
95	3274	0.7ms	.	0.5%	return newBoard

Total time: 0.086s

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	1930	0.9ms	.	1.0%	n = len(board)
181	1930	0.7ms	.	0.8%	linear_conflicts = 0
182					
183					# Find the linear conflicts in rows
184	7720	3.3ms	.	3.9%	for i in range(n):
185	5790	2.4ms	.	2.8%	row = board[i]
186	5790	2.3ms	.	2.7%	goal_row = goal[i]
187	23160	9.8ms	.	11.4%	for j in range(n):
188	17370	8.5ms	.	9.9%	if row[j] != '*' and row[j] in goal_row:
189	15240	7.3ms	.	8.5%	for k in range(j+1, n):
190	7987	4.7ms	.	5.5%	if row[k] != '*' and row[k] in goal_row and goal_row.index(row[j])
191	231	0.1ms	.	0.1%	linear_conflicts += 2
192					
193					# Find the linear conflicts in columns
194	7720	3.3ms	.	3.9%	for j in range(n):
195	5790	6.3ms	.	7.4%	column = [board[i][j] for i in range(n)]
196	5790	6.2ms	.	7.2%	goal_column = [goal[i][j] for i in range(n)]
197	23160	9.9ms	.	11.5%	for i in range(n):
198	17370	8.5ms	.	9.9%	if column[i] != '*' and column[i] in goal_column:
199	14336	6.8ms	.	7.9%	for k in range(i+1, n):

```

200      6851      3.9ms      .      4.5%      if column[k] != '*' and column[k] in goal_column and goal_column.i
201      964      0.4ms      .      0.5%      linear_conflicts += 2
202
203      1930      0.7ms      .      0.9%      return linear_conflicts

```

Total time: 0.048s

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	1930	0.6ms	.	1.2%	state = state.data
208	1930	0.6ms	.	1.2%	goal_state = constants.goalBoard
209	1930	0.5ms	.	1.0%	distance = 0
210					
211					# Create a dictionary that maps each value to its position in the goal state
212	1930	0.5ms	.	1.0%	goal_dict = {}
213	7720	2.3ms	.	4.8%	for i in range(len(goal_state)):
214	23160	6.6ms	.	13.8%	for j in range(len(goal_state[0])):
215	17370	5.0ms	.	10.4%	if goal_state[i][j] != '*':
216	15440	5.0ms	.	10.6%	goal_dict[goal_state[i][j]] = (i, j)
217					
218					# Calculate Manhattan distance
219	7720	2.2ms	.	4.6%	for i in range(len(state)):
220	23160	6.4ms	.	13.4%	for j in range(len(state[0])):
221	17370	6.9ms	.	14.4%	if state[i][j] != '*' and state[i][j] != goal_state[i][j]:
222	11712	3.2ms	.	6.7%	value = state[i][j]
223	11712	3.0ms	.	6.3%	row, col = goal_dict[value]
224	11712	4.6ms	.	9.6%	distance += abs(row - i) + abs(col - j)
225					
226	1930	0.5ms	.	0.9%	return distance

Total time: 0.249s

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	1930	84.8ms	.	34.0%	manhattan_distance = manhattan(board)
232					# Add linear conflict distance
233	1930	163.9ms	0.1ms	65.8%	manhattan_distance += linear_conflict(board.data, constants.goalBoard)
234	1930	0.5ms	.	0.2%	return manhattan_distance + board.depth

Total time: 0.543s

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	0.2ms	0.2ms	.	frontier.append((f(initialNode), initialNode))
241					# frontier-a priority queue
242	1	.	.	.	reached = {str(initialProblem): initialNode}
243					# reached-a lookup table, w
244	1225	0.8ms	.	0.2%	while not frontier.empty():
245	1225	1.0ms	.	0.2%	node = frontier.get()
246					
247	1225	0.9ms	.	0.2%	if constants.goalBoard == node[1].data:
248	1	.	.	.	print('Max queue size:', frontier.getSize())
249	1	.	.	.	return node[1]
250					# then return node
251	4498	174.1ms	.	32.0%	for child in expand(node[1]):
252					# s=child.STATE
253	3274	4.1ms	.	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	3274	7.0ms	.	1.3%	if str(s.data) not in reached or s.depth < reached[str(s.data)].depth:
257	1929	2.6ms	.	0.5%	reached[str(s.data)] = s
258	1929	352.6ms	0.2ms	64.9%	frontier.append((f(s), s))
259					# add child to frontier
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	24	.	.	3.1%	while solution.prev != None:
269	23	.	.	3.5%	stateSequence.insert(0, solution.data)

270	23	.	.	3.1%	actionSequence.insert(0, solution.move)
271	23	.	.	2.0%	solution = solution.prev
272	23	.	.	2.4%	pathCost += 1
273					
274	1	.	.	3.1%	print('Action sequence:')
275	1	.	.	16.5%	print(*actionSequence, sep='\n')
276					
277	1	.	.	1.2%	print('\nState sequence:')
278	1	0.2ms	0.2ms	62.7%	print(*stateSequence, sep='\n')
279					
280	1	.	.	2.4%	print('\nPath cost:', pathCost)