Hollow Man

COMP2611 Artificial Intelligence Coursework 1 Report

1. State representation

I use value 2 to represent a queen, value 1 represents that this square has already been covered, and if the square isn't covered, it will have a 0 in value. So, all the initial state will have a broad filled with 0 values.

Take a 3x3 broad as an example.

This is the initial state:

0	0	0
0	0	0
0	0	0

And this is the goal state:

1	1	1
1	2	1
1	1	1

2. Possible actions

Check every square, if the square isn't queen (not equals to value 2), then it's one of the possible actions.

Screenshot of qc_possible_actions(state) function:

```
# Get all the possible moves
def qc_possible_actions(state):
      moves = []
      for y in range(BOARD_Y):
            for x in range(BOARD_X):
                  # If the quare isn't queen, append it to possible moves
                  if state[y][x] != 2:
                        moves.append((y,x))
      return moves
```

3. Successor state function

Set a queen on one specific square (give value 2), then Cover the square (give value 1) where there's a queen in the same column, or in the same row, or in the same 45 degree diagonal line in any direction.

Screenshot of qc_successor_state(action, state) function:

```
# Get covered when queen moves
def qc_successor_state(action, state):
	RightDown_Y=LeftDown_Y = RightUp_Y = LeftUp_Y = action[0]
	RightDown_X=LeftDown_X = RightUp_X = LeftUp_X = action[1]
	newstate = deepcopy(state)
	newstate[action[0]][action[1]] = 2
	# Cover every square on y axis
	for x in range(BOARD_X):
		newstate[action[0]][x] = 1
	# Cover every square on x axis
	for y in range(BOARD_Y):
		newstate[y][action[1]] = 1
	# Cover every square on right down diagonal line
	while RightDown_Y < BOARD_Y-1 and RightDown_X < BOARD_X-1:
	    newstate[RightDown_Y+1][RightDown_X+1] = 1
	    RightDown_Y += 1
	    RightDown_X += 1
	# Cover every square on right up diagonal line
	while RightUp_Y < BOARD_Y-1 and RightUp_X > 0:
	    newstate[RightUp_Y+1][RightUp_X-1] = 1
	    RightUp_Y += 1
	    RightUp_X -= 1
	# Cover every square on left down diagonal line
	while LeftDown_Y > 0 and LeftDown_X < BOARD_X-1:
	    newstate[LeftDown_Y-1][LeftDown_X+1] = 1
	    LeftDown_Y -= 1
	    LeftDown_X += 1
	# Cover every square on left up diagonal line
	while LeftUp_Y > 0 and LeftUp_X > 0:
	    newstate[LeftUp_Y-1][LeftUp_X-1] = 1
	    LeftUp_Y -= 1
	    LeftUp_X -= 1
	return newstate
```

4. Test for goal state

Go through the whole board and check if any square hasn't been covered (equals to value 0). If any square hasn't been covered, the goal state hasn't been reached, the function will return false, else it has been reached, the function will return true.

Screenshot of qc_test_goal_state(state) function:

```
# Go through the whole board and check if any square hasn't been covered
# If any square hasn't been covered, the goal state hasn't been reached,the function will return false, else it has.
def qc_test_goal_state(state):
    for x in range(BOARD_Y):
        for y in range(BOARD_X):
            if state[x][y] == 0:
                return False
    return True
```