CAP 6635 Artificial Intelligence
Homework 1 | 6/9/2023

Matthew Acs (Z23536012)

1. [1 pt] What is PEAS task environment description for intelligent agent? For the following agents, develop a PEAS description of their task environment

- Assembling line part-picking robot
- Robot soccer player

The PEAS task environment description for intelligent agents specifies the task environment in terms of performance measures, environment, actuators, and sensors. The performance measure is the function that the agent is optimizing via maximization or minimization. The environment is a representation of the states that the word can take. This representation is a formal description, such as a tuple of variables that take certain values according to the current world state. The actuators perform actions that change the current world state according to a model that describes the successor state given the current state and action. Finally, the sensors gather observations that allow the agent to deduce the state of the world. These observations form the inputs to the agent function that dictates the intelligent behavior of an agent.

Agent: Assembling Line Part-Picking Robot

- **Performance Measures:** Percentage of parts in the correct boxes
- **Environment:** Conveyer belt with the parts, boxes
- **Actuators:** Jointed arm with hand
- **Sensors:** Camera, joint angle sensors

Agent: Robot Soccer Player

- **Performance Measures:** Maximize goals scored for the team, minimize goals scored against the team, maximize successful passes, maximize successful dribbles
- **Environment:** Soccer field, soccer ball, opposing soccer team, teammates, referee, fans
- **Actuators:** Jointed legs with feet, jointed arms, jointed head, jointed torso
- **Sensors:** Camera, joint angle sensors

2. [1 pt] Figure 1 shows the relationship between an AI agent and its environment. Please explain the names and functionalities of the parts marked as A, B, C, and D, respectively.

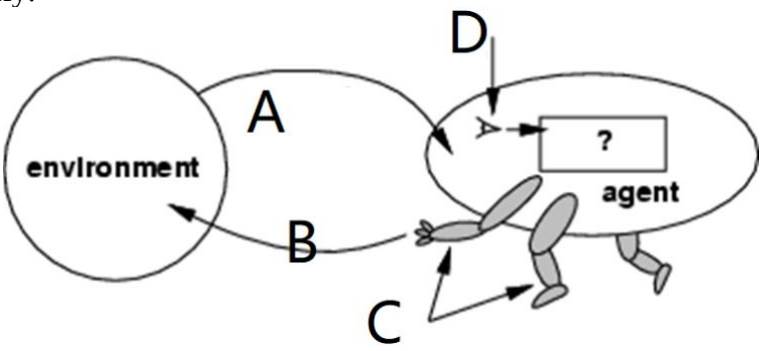


Figure 1

A - Percepts: (A) represents the percepts part of the relationship between an AI agent and its environment. In this part, the agent receives information from the environment as input in the form of perceptions. The history of percepts dictates the agent's actions. The percepts inform the agent about the current world state.

B - Actions: (B) represents the actions part of the relationship between an AI agent and its environment. In this part, the agent interacts with the environment through actions that modify the world state. These actions are determined by the agent function with the percept history as input.

C - Actuators: (C) represents the actuators part of the relationship between an AI agent and its environment. In this part, the agent utilizes actuators to be able to carry out actions such as agent movements. Actuators are parts of the agent, such are hardware components, that allow the agent to be able to complete actions and interact with the environment.

D – Sensors: (D) represents the sensors part of the relationship between an AI agent and its environment. In this part, the agent utilizes sensors to be able to understand the world environment and convey the environment percepts to the agent function. The sensors can be hardware components that are capable of transforming environmental inputs into signals that can be relayed to the agent function.

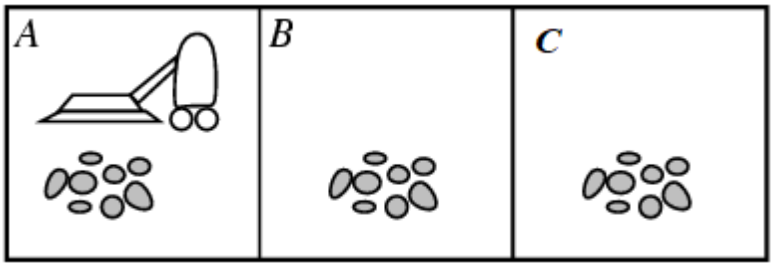


Figure 2

3. [2 pts] Please design pseudo-code of an energy efficient model-based vacuum-cleaner agent as follows:

(1) The environment has three locations (A, B, C as shown in Figure 2) and three states ("Clean", "Dirty", "Unknown").
(2) The vacuum-cleaner has four actions "Left", "Right", "Suck", and "Idle",
(3) The vacuum-cleaner will clean the dirt as soon as it senses that the current environment is Dirty,
(4) If the vacuum-cleaner senses the current environment is "Clean" or "Unknown", it will remain Idle for one time point, and
(5) The vacuum-cleaner will change location if it senses the current environment is "Clean" for two consecutive time points in a row, and
(6) The vacuum-cleaner will change location if it senses the current environment is "Unknown" for two consecutive time points in a row.
(7) When change locations, agent can only move "right" if it is at location "A", move "left" if it is at location "C", and randomly decide to move "left" or "right" if it is location "B"

Summarize percept sequences and corresponding actions as a table [1 pt],
Write the pseudo code of the agent [1 pt] (no need to implement the pseudo code)

Percept Sequence		**Action**
Previous	**Current**	
[A, Dirty] or [B, Clean] or [B, Unknown]	[A, Clean] or [A, Unknown]	Idle
[-]	[A, Dirty]	Suck
[B, Dirty] or [A, Clean] or [C, Clean] or [A, Unknown] or [C, Unknown]	[B, Clean] or [B, Unknown]	Idle
[-]	[B, Dirty]	Suck
[C, Dirty] or [B, Clean] or [B, Unknown]	[C, Clean] or [C, Unknown]	Idle
[-]	[C, Dirty]	Suck

[A, Clean]	[A, Clean]	Right
[A, Unknown]	[A, Unknown]	Right
[B, Clean]	[B, Clean]	Left or Right
[B, Unknown]	[B, Unknown]	Left or Right
[C, Clean]	[C, Clean]	Left
[C, Unknown]	[C, Unknown]	Left

function energy-efficient-model-based-vacuum-cleaner ([location,status]) returns an action
    If c.status == dirty then return Suck
    Else if (p.location == c.location) & ((p.status == clean) or (p.status == unknown)) & (p.location == “A”) then return Right
    Else if (p.location == c.location) & ((p.status == clean) or (p.status == unknown)) & (p.location == “C”) then return Left
    Else if (p.location == c.location) & ((p.status == clean) or (p.status == unknown)) & (p.location == “B”) then return Random(Right or Left)
    Else return Idle

4. [1 pt] Summarize task environment types for the following five agents, in terms of "observable", "deterministic", "episodic", "static", "discrete", "number of agents"

Cross word puzzle Tetris Texas hold'em Robot soccer competition Vacuum Cleaner

Figure 3

Agent	observable	deterministic	episodic	static	discrete	# of agents
Crossword puzzle	fully	deterministic	sequential	static	discrete	single
Tetris	partial	stochastic	sequential	dynamic	discrete	single
Texas hold 'em	partial	stochastic	sequential	static	discrete	multi
Robot soccer competition	partial	stochastic	sequential	dynamic	continuous	multi
Vacuum cleaner	fully	deterministic	sequential	static	discrete	single

Assumptions and Rationale:

- In Tetris, the order that the blocks are going to fall in is not known. This makes the game stochastic because the next state of the environment is not completely predictable by the current state and the action executed by the agent. Furthermore, this makes Tetris partially observable because the next blocks cannot be seen.
- The previous actions of the agent (i.e. where the previous blocks are placed) are an important part of playing Tetris, making the environment sequential. In other words, what happens in the next action is dependent on what happened in the previous action.
- Tetris is dynamic because the block falls with time as the agent is deliberating, thus the environment is changing.
- For the robot soccer, I am assuming that the robots are physical (not virtual) robots in the real world and are thus constrained by sensors, actuators, and the variability of a real-world environment.
- The sensors make the robots only capable of partially observing the environment.
- The real world introduces random variability, which influences the position of the soccer ball, thus making the robot soccer stochastic rather than strategic.
- The real word is continuous, and thus a robot soccer competition would be a continuous environment.

5. [**1 pt**] The goal of the 4-queens problem is to place 4 queens on a 4x4 board (four rows and four columns), such that no two queens attach each other. One solution to the 4-queens problem is shown in Figure 4. Design a search-based solution to solve the 4-queens problem. Your solution must include following four components:

a. Define state and show examples of three states (including initial state and goal state) [0.25 pt]
b. Define a successor function, and estimate total number of states, based on the defined successor function [0.25 pt]
c. Show state graph with at least five states and their relationships (the graph must have a path from initial state to goal state) [0.25 pt]
d. Show how to find solution(s) from the state graph [0.25 pt]

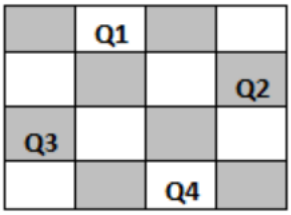
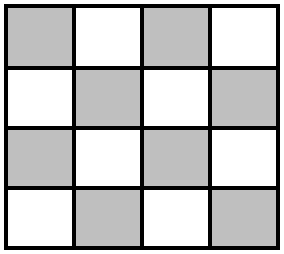


Figure 4

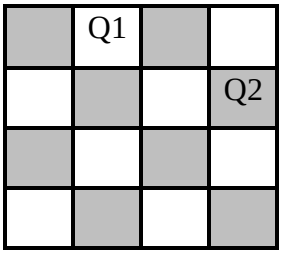
a)

State Definition: All arrangements of k = 0, 1, 2, ..., 4 queens in the k leftmost columns with no two queens attacking each other.

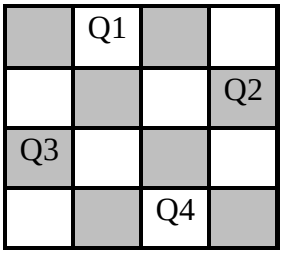
Initial State: No queens are placed on the board



Initial State



Intermediate State



Goal State

b)

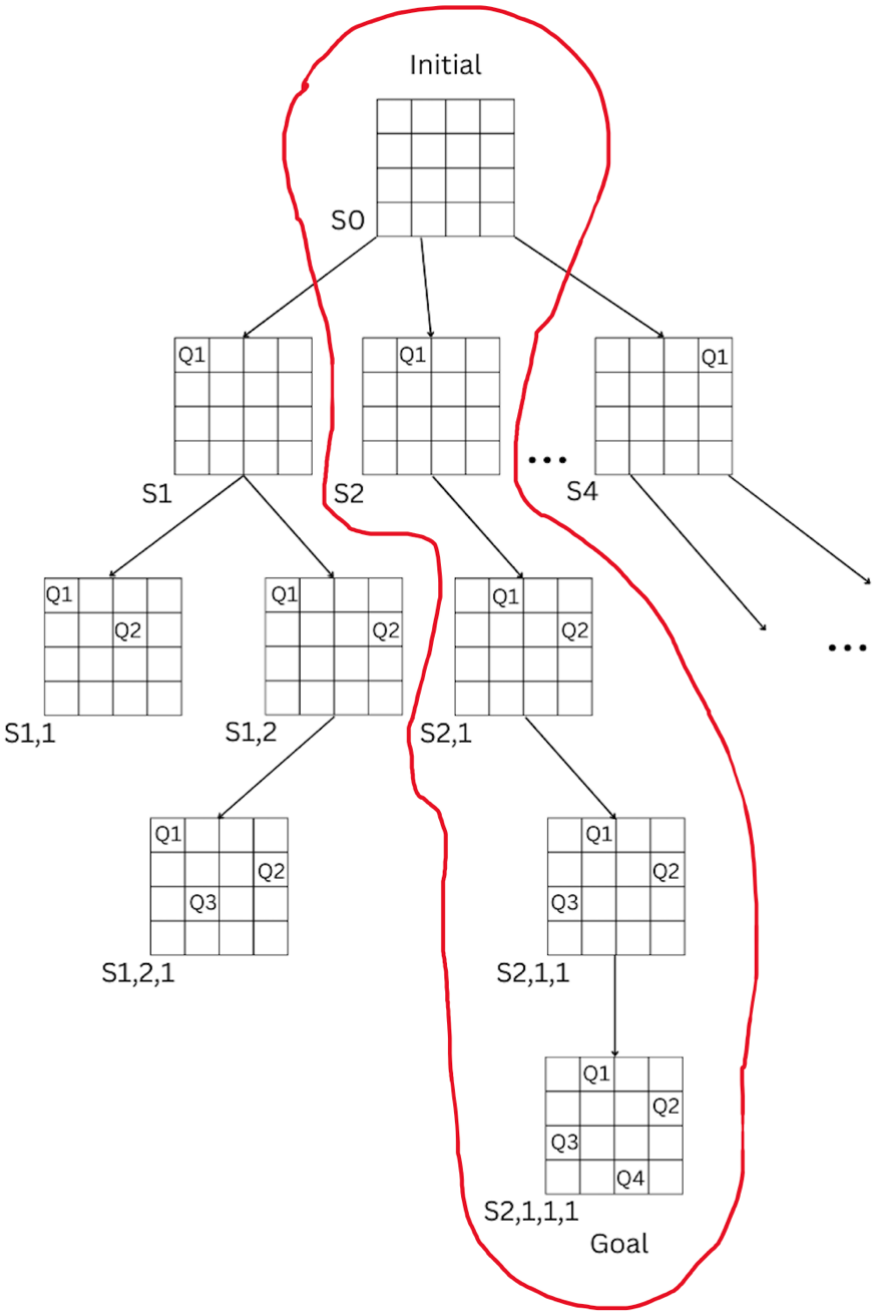
Successor function: Q_{ij}, place i^{th} queen on the j^{th} topmost empty row, such that each successor is obtained by adding one queen in any square that does not attack any queens above the current row.

- 1 Initial State
- First row has 4 placements at most
- Second row has 2 placements at most
- Third row has 1 placement at most
- Last row has 1 placement at most

(1)(4)(2)(1)(1) = at most 8 goal states
1 + (1)(4) + (1)(4)(2) + (1)(4)(2)(1) + (1)(4)(2)(1)(1) = 1 + 4 + 8 + 8 + 8 = at most 29 total states

Based on the successor function, there will be at most 8 goal states and at most 29 total states.

c)



The state graph above shows an initial state, several intermediate states, and a path to a goal state

that can be obtained from the graph as a solution to the four queens problem.

d)

To find solutions from the state graph, you need to traverse the graph to the goal state by finding a path that leads to the fourth (four queens placed) level of the tree. Any states at this level will be goal states because the successor function ensures that queens can only be placed in ways that generate valid intermediate states. Thus, any state that is generated that contains four queens is a valid goal state. These states are found in the final level of the graph. In the example above, the path circled in red shows how you can find one of the solutions to the four-queens problem. In this path, after the initial state, the first queen is placed in the second column of the first row. Then the second queen is placed in the fourth column of the second row. The third queen is placed in the first column of the third row, and finally the fourth queen is placed in the third column of the fourth row. This final queen placement generates one of the goal states.

6. [**1 pt**]. Figure 5 shows a state graph with five states. Assume that "a" is the initial state, and "G" is the goal state.

a. Use Breath First Search (BFS) to build a search tree to carry out search from the initial state to the goal state (Show complete search tree). [1 pt]

i. Mark order of the search node (on the BFS tree) being visited from initiate state to find the goal state (break tie in favor of node with a lower alphabetic order, e.g., "b" has a higher priority than "c"). If multiple nodes are created for same state, say "f", use f1, f2, f3, etc., to denote each node (the number corresponding to node being created).

ii. Show final path from "a" to the goal state. Is the solution optimal? Why or why not?

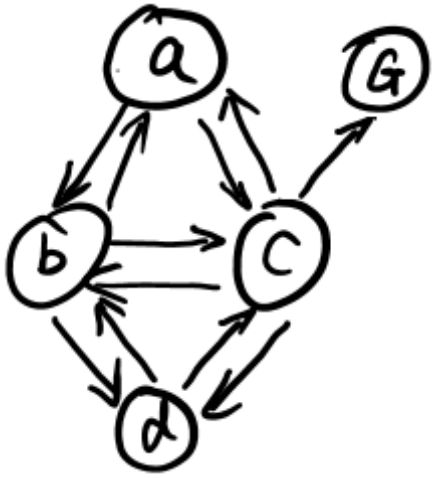
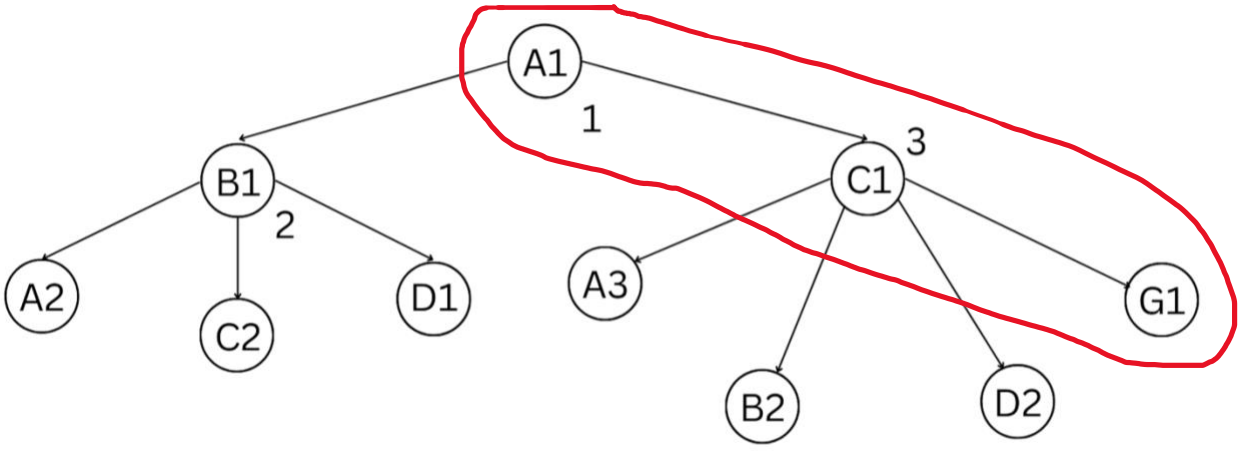


Figure 5

i)



The order that the nodes were visited is A1, B1, and C1 and the path from the initial node to the goal node is A1 -> C1 -> G1.

ii)

The final path from “a” to the goal state is A -> C -> G and the solution is optimal. This is because breadth first search always finds the minimum cost path if the step cost is one for all edges. In this case, breadth first search is used, and all step costs are one unit, thus it is optimal. Breadth first search explores all the nodes at a certain depth, before moving onto the next level. This ensures that if a solution is possible, the search strategy will find the minimum cost path to it. In the case of the search tree generated for the graph above, the path A -> C -> G is optimal because breadth first search is optimal for a step cost of one. It can also be seen from looking at the graph that the shortest path from A to G is the path that was found, thus verifying the result from an intuitive perspective.

7. [**1 pt**]. Figure 6 shows a state graph with five states. Assume that "a" is the initial state, and "G" is the goal state. The value above each line shows the arch cost.

a. Use Uniform Cost Search (UCS) to build a search tree to carry out search from the initial state to the goal state (Show complete search tree). [1 pt]

iii. Mark order of the search node (on the search tree) being visited from initiate state to find the goal state (break tie in favor of node with a lower alphabetic order, e.g., "b" has a higher priority than "c".). If multiple nodes are created for same state, say "f", use f1, f2, f3, etc., to denote each node (the number corresponding to node being created).

iv. Show final path from "a" to the goal state. Is the solution optimal? Why or why not?

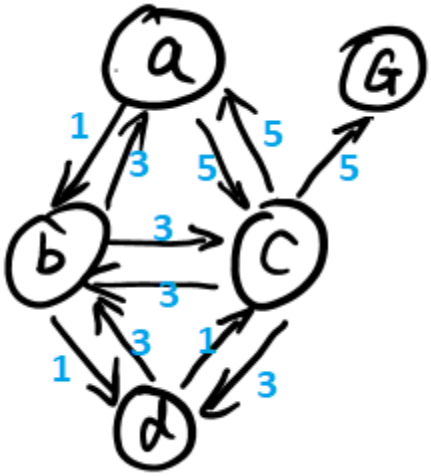
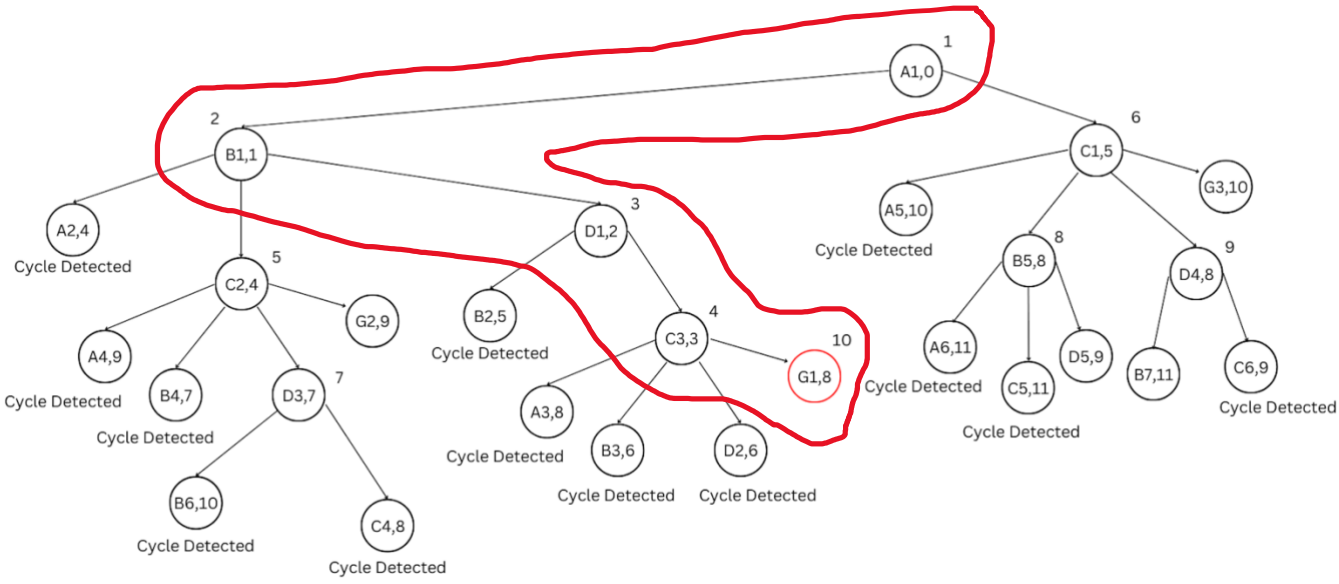


Figure 6

a)



The order that the nodes were visited is A1, B1, D1, C3, C2, C1, D3, B5, D4, and G1 and the path from the initial node to the goal node is A1 -> B1 -> D1 -> C3 -> G1. The search tree above names the nodes in the form (A || B || C || D || G)(# generated), (Path cost from initial node to this node), thus it can be seen what order the nodes were generated in and what the path cost to get to each node is. When a cycle was detected, the tree was pruned because cycles do not advance the solution in this case.

b)

The solution is optimal. This is because uniform cost search is an optimal search strategy and always finds the solution of lowest cost if a solution exists. It considers the cost of each path as it decides which node to explore from the fringe. Thus, it always explores the next lowest cost path, leading to finding the solution of lowest cost rather than a solution of higher cost. The path to the optimal solution is circled in red in the search tree above. The optimal path is A -> B -> D -> C-> G and the cost is 8. This makes sense from an intuitive standpoint as the path of least cost follows the path of 1s that go from A to B to D to C and then finally G.

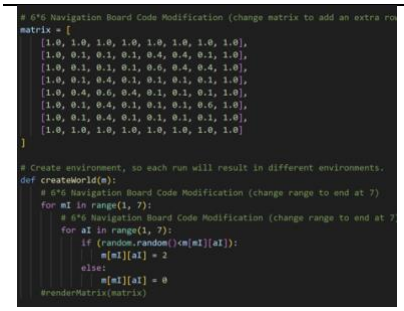
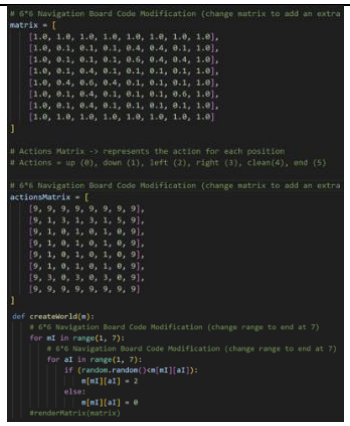
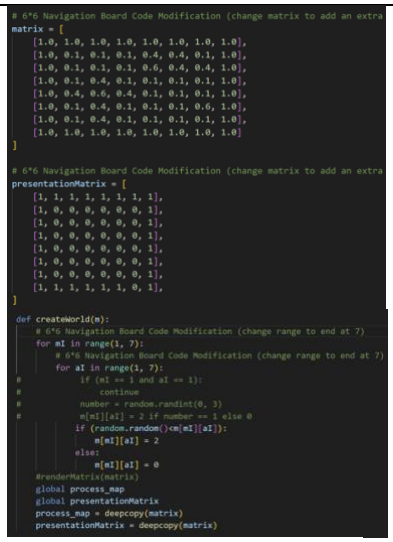
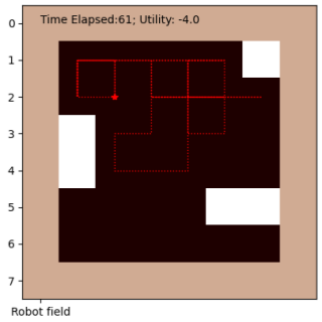
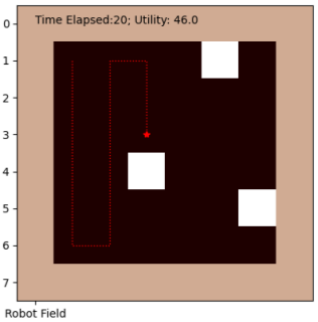
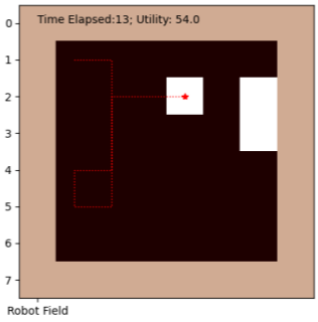
8. [2 pts] The "Source Code" module in Canvas lists three python programs. "simple.py" is a simple reflex agent, "model.py" is a reflex agent with a model to ensure agent walking through all locations. "goal.py" is a goal-based agent aiming to clean all dirty spots with minimum steps of moves.

e. Use proper python IDE environment (FAU HPC environment will allow students to use Jupyter Notebook for python coding) to run each of the three python programs. Change environment as a 6x6 navigation board. Capture one screenshot (or plot) showing that the program is properly running (0.5 pt).

f. Explain how the intelligent behaviors are improved from simple.py, model.py, and goal.py? (0.5 pt).

g. Propose a solution to design a fourth agent program which may outperform "goal.py". Explain why your proposed solution can perform better (1 pt). (no need to implement the agent. Just description your idea using textual descriptions or pseudo-code).

e)

simple.py	model.py	goal.py
		
		

f)

Initially, in simple.py the agent moves randomly until it finds dirt. Then it cleans the tile and moves randomly again. This process is repeated until all the dirt is gone. This is the lowest level of intelligence exhibited by the three agents because it does not guarantee that all tiles will be visited and cleaned within a specified time frame. The random behavior could lead to the agent moving between a subset of the tiles and not cleaning dirty tiles for a longer time. However, the random behavior ensures that the agent does not get stuck in a pattern of behavior. Model.py is more intelligent because the agent follows a pattern of behavior that reaches all the tiles without repeating a tile. This a better than randomly moving because all the tiles are ensured to be reached, and no tiles are repeated, increasing the efficiency of the agent. This guarantees that all dirty tiles will be cleaned, however, it is not as efficient as possible because it visits many more tiles than necessary to clean the dirty tiles, which are a subset of all the tiles. Finally, goal.py is the most intelligent agent because it searches for the closest dirt and then moves to the dirt to clean it. This process is then repeated until all the dirt is cleaned. This is an improvement from the model-based agent because it is more efficient due to it not visiting every tile. It guarantees that all dirty tiles are cleaned, and it does not require all tiles to be visited, which is an improvement from a behavior pattern that requires every tile to be visited. This maximizes the utility function and thus leads to a more intelligent agent.

g)

A solution that may outperform goal.py is a breadth-first search-based agent. Based on the map, dirt locations, and the agent starting position, you generate a search tree that identifies the shortest path to go to every dirt using a breadth first search approach. You create the search tree by exploring all nodes at a certain depth before you move to the next level. This is repeated until a goal state is found. Then the agent follows the path identified in the goal state. Since the cost to move between each tile is 1, breadth first search will identify the optimal solution, thus giving the most efficient agent movement to clean up all the dirt. This may be more efficient than the goal.py agent because that agent only moves to the closest dirt, but it does not consider the entire path that it will take. The BFS based agent will always follow the optimal path to clean up all the dirt. The BFS agent will be the best/most intelligent solution because it is optimal, and no solution can be better than optimal regarding behavior. However, the BFS agent may have a greater run time than the goal.py agent because BFS search take $O(b^d)$. At large environment sizes, the BFS agent may take significantly longer due to the time complexity.