**Agent and Problem Formulation**

**1.**

(a) Agent

An agent is an entity that performs actions or tasks on behalf of a party or individual. It is the "doer" in a scenario. Agents interact with environments and make decisions to achieve certain objectives.

(b) Agent Function

The agent function is the brain of the agent, mapping each sequence of perceptions to an action. It as the rulebook that guides the agent's behavior within its environment.

(c) Agent Program

The agent program, is the implementation of the agent function. It is the code, that turns the theory into something actionable.

(d) Rationality

Rationality, is the cornerstone of agent behavior. Rationality is the measure of how well an agent's actions align with its performance measure. The agent makes the "best" decisions given what the it knows, what it can do, and what it aims to achieve.

(e) PEAS

PEAS stands for Performance measure, Environment, Actuators, and Sensors. It is a framework for specifying the aspects that influence an agent's operation. The "Performance measure" gauges how well the agent is doing, while the "Environment" is where the agent operates. "Actuators" are the parts that perform actions, and "Sensors" gather information. Together, they shape the agent's world.

(f) Reflex Agent

Reflex agents make decisions solely based on their current percept, without considering the broader history. Reacting on instinct like a stimulus. They produce a quick response but are not always the best for complex scenarios.

(g) Model-based Agent

A model-based agent, maintains an internal model of the world. It factors in past experiences when making decisions. It's like its able to test out scenarios before taking action.

(h) Goal-based Agent

The goal-based agent is future-focused. Instead of just reacting to stimuli, this agent has goals it aims to achieve. Its actions are directed towards fulfilling these goals. It is always thinking several moves ahead.

(i) Problem Formulation

Problem formulation is defining a problem in a way that makes it solvable by an agent. It sets the criteria, detailing what the agent knows, what it needs to find out, and what actions it can take.

(j) Transition Model

The transition model describes how actions change the state of the world. It defines what happens when an agent takes a certain action in a given state. It essentially outlines the dynamics of the environment.

**2.**

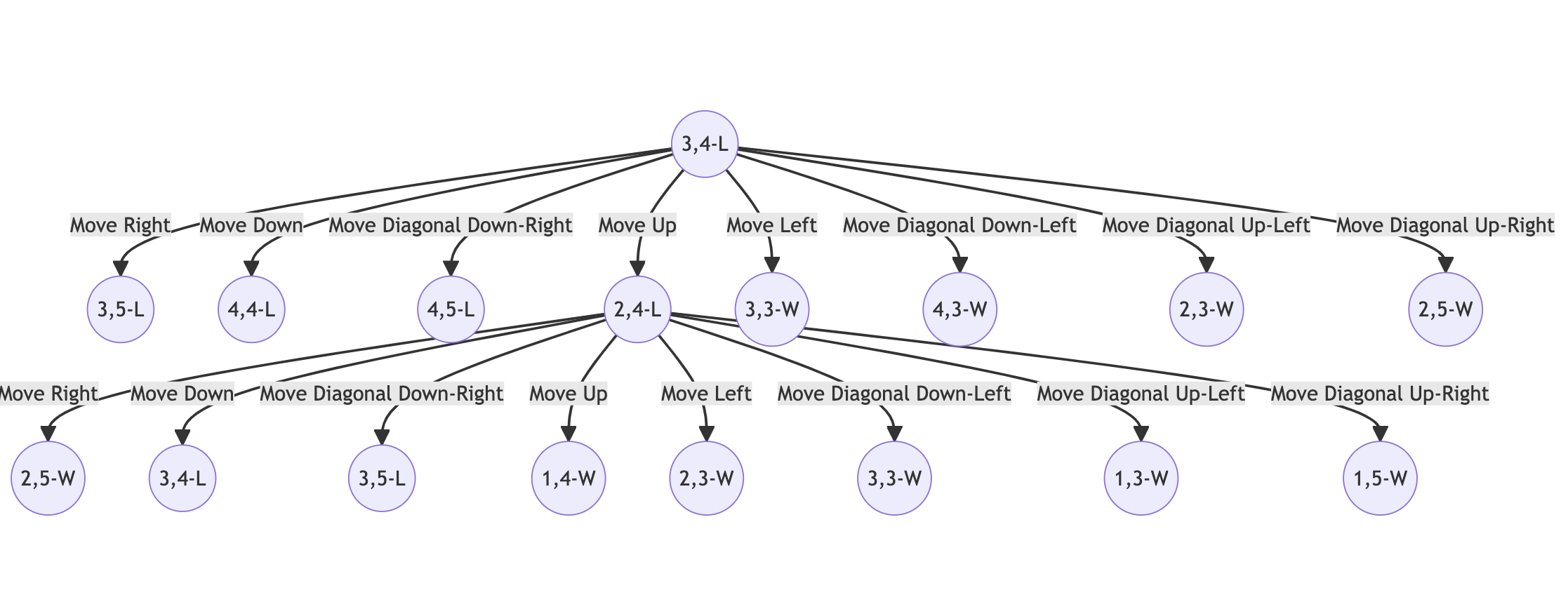
(a) In Goal 1, the agent's task is to move through the 8x13 grid to identify clusters of '1's that form an island. Each island is defined as a contiguous group of '1's connected either horizontally or vertically. The agent will have to scan through each cell in the grid, marking visited cells to avoid redundant checks.

(b) For Goal 2, the agent will not only identify islands but also calculate their areas, which is the total number of '1's in a given island. The agent will track the maximum area encountered during its search.

(c) The agent will start at the top-left corner of the grid and proceed row by row, scanning each cell. If a '1' is encountered, the agent will start a depth-first search (DFS) to explore the contiguous land mass, marking each visited cell.

(d) Once the agent locates a '1', it will start DFS to search all connected '1's, keeping a count of the number of cells it visits. The count represents the area of the island.

(e) If the agent finds one land, the number of possible states for the land in a larger island will depend on the number of connected '1's. Each '1' can have 4 connected states (up, down, left, right).

(f)  


(g) So here's the thing, when dealing with problems where you have to explore different paths, you basically have two popular choices, Depth-First Search (DFS) and Breadth-First Search (BFS). Each has its merits, but for this specific scenario, DFS really shines. First off, DFS is super space-efficient. Why? Because it only needs to store the nodes along the path from the root node to the current node. That's way less overhead compared to BFS, which keeps track of all possible next moves at each step. And let's be real, when you're working with AI agents navigating through grids, efficiency is key. Secondly, DFS is really straightforward when it comes to implementation. We're talking about a recursive function that dives deep into each path before backtracking. DFS gives you the area of each island on the fly, while you're exploring it. The algorithm navigates through connected '1's, and you can easily count the cells it goes through. That's super handy for Goal 2, where we need to find the maximum area of an island. You might argue that BFS could also work, and you wouldn't be wrong. But BFS explores all neighbors first before moving on, meaning you'd have to store way more nodes at each level. That's overkill for our scenario. Plus, calculating the area for each island becomes trickier because BFS doesn't naturally explore all connected components in one go.

(h)  attached “Assign02Q2hVE.py”