**IT159IU: Artificial Intelligence**

**Lab 1: Designing Pac-Man Agents**

Hoang Van Manh – ITDSIU21099

1. **Describe the behavior of RandomAgent from step 7:**

**A screen shot of a computer code

AI-generated content may be incorrect.**

* **Lack of Strategy:** The RandomAgent operates without any strategic planning. It selects its movement direction entirely at random.
* **Inefficient Movement:** Due to its random nature, the agent often wanders aimlessly, sometimes circling without reaching the food pellets
* **Potential for Stalling:** If the "STOP" action is selected repeatedly, the agent may remain stationary for extended periods.
* **Inconsistent Food Collection:** The agent's random movement results in inconsistent food collection, and it may fail to consume all available food, particularly in larger mazes like "mediumMaze".
* **Time Inefficiency:** In larger environments, the RandomAgent's lack of a direct strategy leads to significant time wastage.

1. **A screen shot of your myLayout environment from Step 8:**

**A screenshot of a computer

AI-generated content may be incorrect.A screen shot of a game

AI-generated content may be incorrect.**

**%:** Wall

**.:** Food

**P:** Pac-Man's starting position

* Run RandomAgent with myLayout environment:

**python pacman.py --layout myLayout --pacman RandomAgent**

* Run RandomAgent with openSearch

**python pacman.py --layout openSearch --pacman RandomAgent**

Average Score after 5 times:

* Score 1: 144
* Score 2: -200
* Score 3: 14
* Score 4: -430
* Score 5: -100
* Average = -114.4
* RandomAgent can eat all foods, but it takes a long time.

1. **Describe the behavior of RandomAgent from Step 9**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**Behavior:**

* **Improved Randomness:**
  + This agent operates randomly, but with a key improvement over the basic RandomAgent.
  + Specifically, it will never choose the 'STOP' action.
* **Valid Action Selection:**
  + At each step, the agent retrieves the list of valid actions that Pac-Man can take in the current state.
  + It then removes the 'STOP' action from this list.
  + Finally, it randomly selects an action from the remaining valid actions.
* **Continuous Movement:**
  + By eliminating the 'STOP' action, Pac-Man will move continuously, avoiding prolonged periods of inactivity.

**Advantages over the basic RandomAgent:**

* Overcomes the issue of prolonged inactivity seen in the basic random agent.
* Increases Pac-Man's movement efficiency.

**Disadvantages:**

* It still relies on random behavior, so it lacks a specific strategy to optimize food consumption.
* It can still waste time in large mazes.

1. **Describe the behavior of ReflexAgent from Step 10**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Key Behaviors:**

* **Food-First Approach:**
  + The agent's primary objective is to consume food. It actively searches for actions that will lead to eating a food pellet in the immediate next step.
  + It iterates through available legal actions and simulates the resulting state (successor state) to check if Pac-Man's new position will coincide with a food location.
  + If a food-eating action is found, it immediately returns that action.
* **Reactive Decision-Making:**
  + The agent operates on a "reflex" basis, making decisions based solely on the current state and the immediate next state. It does not plan or consider future consequences beyond the next move.
* **Random Movement (Fallback):**
  + If no immediate food-eating action is available, the agent resorts to random movement.
  + Similar to the BetterRandomAgent, it excludes the "STOP" action to ensure continuous movement.
* **State Information Printing:**
  + The agent prints out a lot of information about the current state of the game. This includes Pac-Man's position, ghost positions, wall grid, food locations, remaining food count, capsule locations, win/lose status, and current score. This print functionality is very helpful for debugging and understanding the agents behavior.

**Strengths:**

* Effective at quickly consuming nearby food.
* Avoids prolonged inactivity by eliminating the "STOP" action.

**Weaknesses:**

* Lacks long-term planning. It may get trapped or make suboptimal moves due to its limited foresight.
* Its random movement when no food is nearby can lead to inefficient exploration.
* It does not take ghost positions into consideration when making a move, so it is possible to run directly into a ghost.

1. **For each of the percepts listed in Step 10, show what command/code enables you to access it**

**Percepts and Access Code:**

* **Pac-Man's Position:**
  + gameState.getPacmanPosition()
* **Ghost Positions:**
  + gameState.getGhostPositions()
* **Walls Grid:**
  + gameState.getWalls()
* **Food Locations:**
  + gameState.getFood().asList()
* **Remaining Food Count:**
  + gameState.getNumFood()
* **Capsule Locations:**
  + gameState.getCapsules()
* **Game Won?**
  + gameState.isWin()
* **Game Lost?**
  + gameState.isLose()
* **Current Score:**
  + gameState.getScore()