

EXPERIMENT 1

PROBLEM TITLE

Implement a Simple Reflex Agent and a Model-Based Reflex Agent in a Simulated Environment

OBJECTIVE OF THE PROGRAM

To understand the basic principles of intelligent agents by implementing a simple reflex agent and a model-based reflex agent in a simulated environment.

DESCRIPTION

An intelligent agent perceives its environment through sensors and acts upon it through actuators. A simple reflex agent selects actions based on the current percept, ignoring the rest of the percept history. In contrast, a model-based reflex agent maintains an internal state to keep track of the world and make decisions based on the current state and percept.

ALGORITHM

Simple Reflex Agent:

1. Initialize the agent with a set of condition-action rules.
2. Observe the current percept.
3. Apply the rule that matches the current percept.
4. Execute the corresponding action.

Model-Based Reflex Agent:

1. Initialize the agent with a set of condition-action rules and an internal state.
2. Observe the current percept.
3. Update the internal state based on the percept.
4. Apply the rule that matches the current state.
5. Execute the corresponding action.

EXPERIMENT 2

PROBLEM TITLE

Write a Program to Implement Breadth-First Search (for Tic-Tac-Toe Problem).

OBJECTIVE OF THE PROGRAM

To implement the Breadth-First Search (BFS) algorithm and use it to explore the state space of the Tic-Tac-Toe problem.

DESCRIPTION

Breadth-First Search (BFS) is an uninformed search algorithm that explores the state space level by level. It is particularly useful in scenarios where the solution is not too deep from the initial state, as it guarantees the shortest path to the goal.

ALGORITHM

1. Initialize the BFS queue with the initial state.
2. Loop until the queue is empty:
 - a. Dequeue the front node.
 - b. If the node represents a goal state, return the solution.
 - c. Otherwise, enqueue all possible child nodes (next states).
3. If the queue is empty and no solution is found, return failure.

EXPERIMENT 3

PROBLEM TITLE

Write a Program to Implement Depth-First Search (for Water Jug Problem).

OBJECTIVE OF THE PROGRAM

To implement the Depth-First Search (DFS) algorithm to solve the Water Jug Problem, where the goal is to measure a specific amount of water using two jugs of different capacities.

DESCRIPTION

Depth-First Search (DFS) is an uninformed search algorithm that explores as far as possible along a branch before backtracking. It's useful for problems with deep solutions but doesn't guarantee the shortest path.

ALGORITHM

1. Start with the initial state (empty jugs).
2. Mark the current node as visited and explore its unvisited neighbours recursively.
3. If a goal state is reached (desired amount of water), return the solution.
4. If no solution is found, backtrack to explore other paths.

EXPERIMENT 4

PROBLEM TITLE

Write a Program to Implement A* Search.

OBJECTIVE OF THE PROGRAM

To implement the A* search algorithm and use it to solve a pathfinding problem in a grid, demonstrating the efficiency of informed search strategies.

DESCRIPTION

A* search is an informed search algorithm that uses both the cost to reach the node (g) and a heuristic estimate of the cost to reach the goal (h). The algorithm expands the node with the lowest estimated total cost $f(n) = g(n) + h(n)$. A* is optimal and complete when the heuristic used is admissible (never overestimates the cost).

ALGORITHM

1. Initialize the open list with the start node.
2. Loop until the open list is empty:
 - a. Select the node with the lowest $f(n)$ value.
 - b. If this node is the goal, reconstruct the path and return it.
 - c. Otherwise, generate the node's neighbors and calculate their g , h , and f values.
 - d. Add the neighbors to the open list if they have not been visited or if a lower-cost path to the node is found.
3. If the open list is empty and no solution is found, return failure.