Question 1

1.1.

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuator.

Single Agent

A single agent environment is an environment that is explored by a single agent. All actions are performed by a single agent in the environment. An example of this would be playing tennis against a ball where there is only one player.

Mutli-Agent

A multi-agent environment is one where two or more agents are present, perceiving and performing actions in the environment. The environment may be competitive, or cooperative. An example of this would be playing a soccer match

1.2.

Deterministic

These are the environments where the next state of the environment is completely predictable from the current state and the action executed by the agent. There is no uncertainty in the environment.

Stochastic

This environment is the opposite of a deterministic environment. The next state is totally unpredictable for the agent. Uncertainty exists due to randomness, lack of good environment model or lack of complete sensor coverage.

1.3. A game of chess is fully observable. In this environment, the agent sensor is able to sense/access the complete state of and agent at each point in time.

There is no need to keep track of the keep track of the history of the surrounding so the game is not necessarily partially observable.

If the agent has no sensors in all environments, the environment would be unobservable which is not the case in a chess game.

Question 2

2.1.

A well-defined problem can be described by:

|  |  |
| --- | --- |
| **A start or initial state** | initial state that the agent starts in |
| **Actions** | A description of the possible actions available to the agent. |
| **Transition model** | This is specified by a function . A transition model is a description of what each action does. A successor is any state reachable from a given state by a single action. |
| **Path cost** | function that assigns a numeric cost to a path. Cost of a path is the sum of costs of individual actions along the path |
| **Goal test** | test to determine if at goal state |

2.2.

2.3.

In a search, a graph can be divided into three parts: nodes explored, nodes to be explored next and the remaining unexplored nodes. The explored set keeps track of the nodes that have already been expanded. This aims to prevent the situation where we might get into an infinite loop, expanding the same nodes over and over.

2.4.

**Priority Queue**

This type of queue is used in a best-first search. A best-first search is a graph traversal algorithm that uses and evaluation function to decide which adjacent node is most promising and then explore.

In this type of search a priority queue is used to store the cost of nodes, with the nodes of the highest priority stored at the front of the queue.

**FIFO (First-in-first-out) queue**

This type of queue is used in a breadth-first search. A breadth-first search is a graph traversal algorithm that traverses a graph in a breadth-ward (or wide) motion: it explores the closest vertices first and moves outwards away from the source.

In this type of search, it is important to store which vertices have been visited and in what order. To facilitate this process, a FIFO queue is used to insert the nodes that have been visited first and returns the oldest element, based on the order it was inserted.

**LIFO (Last-in-first-out) queue/stack**

This type of queue is used in a depth-first search. A depth -first search is a graph traversal algorithm that traverses a graph in a depth-ward (or deep) motion. It explores as far as possible along each branch first and then backtracks.

In this type of search, a stack works best as it is LIFO. The search needs to remember where it should go when it reaches a dead end.

2.5.

List and explain the measures used to determine problem solving performance.

A certain list of criteria is used and considered to evaluate an algorithm’s performance:

**Completeness**

A complete algorithm must be capable of systematically exploring every state that is reachable from the initial state. The algorithm should be guaranteed to find a solution when there is one, and to correctly report failure when there is not. A search algorithm must be systematic in the way it explores an infinite state space, making sure it can eventually reach any state that is connected to the initial state.

**Cost optimality**

A solution should be guaranteed to be optimal. The algorithm should find a solution with the lowest path cost of all existing solutions.

**Time Complexity**

This attribute also considers the measure of difficulty of the problem. The algorithm should take the least time to find a solution, measured in

seconds, or more abstractly by the number of states and actions considered.

**Space Complexity**

This attribute also considers the measure of difficulty of the problem. The algorithm should utilize the least amount of memory needed to perform the search.

Question 3

3.1.

3.2.

3.3.

3.4.

Question 4

4.1.

4.2.

4.3.

Question 5

5.1.

Question 6

6.1.

6.2.

Question 7

7.1.

Question 8

8.1.

8.2.

8.3.

8.4.

Question 9

9.1.

9.2.

9.3.

9.4.