**ARTIFICIAL INTELLIGENCE AND COMPILER DESIGN**

**Part 1**

1. C) John McCarthy
2. D) All of the above
3. C) Agent and Environment
4. C) Intelligence and capability
5. B) Model-based reflex agent
6. D) Utility-based agent
7. D) Utility-based agent
8. A) Simple reflex agent
9. D) All of the above
10. A) Sensors
11. D) Goal based agent
12. A) Search space
13. A) Optimal solution
14. D) Both A and C
15. A) Iterative deepening depth-first search
16. D) All of the above
17. D) h(n) + g(n)
18. D) None of the above
19. A) Informed search
20. C) Both

**Part 2**

1. A program that is intelligent may complete tasks or make judgments depending on its surroundings, input from users, and past experiences. These applications can be used to collect data automatically on a predetermined schedule or in response to user input in real time.
2. Foundations of artificial intelligence:
3. Computer science
4. Mathematics
5. Philosophy
6. Psychology
7. Biology
8. Neural science
9. Sociology
10. Economics
11. Linguistics
12. Types of agents and their environments:
13. Reflex agents: These agents operate solely in the present, disregarding the past. The event-condition-action rule, which bases responses on the user initiating an event and the agent consulting a list of pre-set rules and pre-programmed outcomes, is used.
14. Model-based agents: These agents make decisions about what to do in the same manner that reflex agents do, but they have a more thorough understanding of their surroundings. The internal system is programmed with a world model that includes the agent's past.
15. Goal-based agents: These agents include goal information, or details about ideal circumstances, in addition to the data that model-based agents retain.
16. Utility-based agents: These agents resemble goal-based agents but include an additional utility measurement that ranks each potential scenario according to the intended outcome and selects the course of action that maximizes the result. Examples of rating criteria include the likelihood of success or the quantity of needed resources.
17. Goal-based agents are similar to utility-based agents, but utility-based agents rank each possible scenario according to the desired outcome and choose the course of action that will maximize the outcome. The chance of success or the quantity of resources required are a couple of examples of grading criteria.
18. Agent = Architecture + Agent program

The equipment on which an AI agent runs is referred to as the architecture.

In order to translate a percept into an action, we employ the agent function.

Agent program: The implementation of agent function by an agent program. For the physical architecture to work, an agent program must run.

1. Problem solving agents:

Goal-based agents that define problems and offer many solutions are the problem-solving agents in artificial intelligence. In the field of artificial intelligence, it is the culmination of a collection of methods and algorithms for tackling a certain issue. A tree, a B-tree, and heuristic algorithms are just a few of the methods it includes. These agents differ from reflex agents, which can only map states into actions when storing and learning, and cannot map when performing actions. In order to achieve a desired state or solution, problem-solving agents must go through a number of steps.

* stating or being clear about the intended objective
* Study and investigation
* Use the available algorithms to find the solution.
* Execution

1. Informed search algorithm:

Search algorithms that are informed have knowledge of the goal state, which makes searching more effective. A function that determines how close a state is to the target state estimates this information can be used.

1. Uninformed search algorithm:

Other than the information supplied in the problem statement, uninformed search algorithms have no other knowledge about the goal node. Plans to go from the starting state to the objective state simply differ in terms of the order and duration of actions.

1. Local search algorithm:

They begin with a potential solution and then shift to a nearby solution. Even if they are interrupted at any point before they finish, they can still deliver a correct answer.

1. Local search algorithm in continuous space:

* They generally apply to issues for which we simply need to know the answer itself, not the way to get there.
* They work with a single state or a small group of states and investigate the states that are nearby. Normally, they don't store the path.
* Problems with optimization, for which we seek the optimal solution in accordance with an objective function, are one such instance.
* Landscape refers to the distribution of values for the objective function in the state space.

1. Optimization problems:

The process of determining the input variables or arguments that produce a function's minimum or maximum output is referred to as optimization. Continuous function optimization is the most typical sort of optimization problem in machine learning, where the function's input parameters are real-valued numeric values, such as floating-point values. A real-valued evaluation of the input values appears in the function's output as well. In order to distinguish these problems from those that involve functions that accept discrete variables and are known as combinatorial optimization problems, we might refer to problems of this kind as continuous function optimization.

The quantity of data about the target function being optimized that may be utilized and taken advantage of by the optimization algorithm is one method for classifying optimization algorithms.

Optimization problems have some combinatorial structure to the problem. Constraints may have to be satisfied and there is also a cost function, which we want to optimize. Searching for all possible solutions is not feasible.

Characteristics of optimization problems:

• The problem is described by a set of states (configurations) and an evaluation

Function. E.g., in the Travelling Sales Person problem, a tour is a state, and the length of the tour is the evaluation function to minimize.

• The state space is too big to enumerate all states (or the evaluation may be expensive to compute for all states). E.g. in the Travelling Sales Person problem, the state space is (n − 1)!/2, where n is the number of vertices to connect.

• We are only interested in the best solution, not the path to the solution.

• Often it is easy to find some solution to the problem.

• Often it is provably very hard (NP-complete) to find the best solution.

Part 3





