Artificial Intelligence

Assignment – III | Unit – III (New Syllabus - PU)

3. Problem Solving and Search Algorithms

3.1 Problem Solving

3.1.1 Problem-Solving Agents

1. Define a problem-solving agent. How is it different from a simple reflex agent?
2. Explain the role of a problem-solving agent in an AI system with examples like a GPS navigation system.

3.1.2 Problem-Solving Process

3. Describe the steps involved in the problem-solving process.
4. How does the problem-solving process vary for deterministic vs. stochastic environments?

3.1.3 Production System

5. What is a production system? Explain its components (rules, working memory, control strategy) with examples.
6. Model the **Water Jug** problem as a production system.

3.1.4 Well-Defined and Ill-Defined Problems

7. Differentiate between well-defined and ill-defined problems with examples.
8. Provide a real-world example of an ill-defined problem and suggest how AI can help in solving it.

3.1.5 Problem Formulation

10. What is problem formulation? Why is it critical in designing intelligent agents?
11. Formulate the **8-puzzle** problem, specifying the state space, initial state, and goal state.
12. Design a problem formulation for the **Rubik's Cube**.

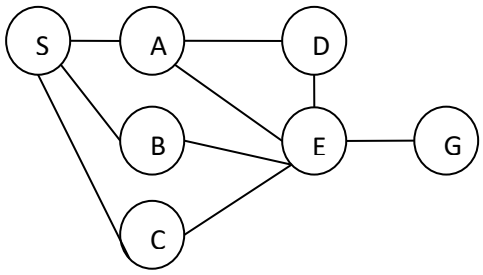
3.2 Search Algorithms

3.2.1 Uninformed Search

3.2.1.1 Breadth-First Search (BFS)

13. Explain the Breadth-First Search algorithm with an example.
14. What are the properties of BFS (completeness, optimality, time complexity, space complexity)?

15. Trace the BFS algorithm for a graph where **S** is the start node, and **G** is the goal node.



3.2.1.2 Depth-First Search (DFS)

16. Explain the Depth-First Search algorithm with an example.
17. What are the properties of DFS?
18. Illustrate DFS on a tree and explain its backtracking mechanism.

3.2.1.3 Iterative Deepening Search

19. What is Iterative Deepening Search (IDS)? How does it combine the benefits of BFS and DFS?
20. Discuss Depth Limit Search and Iterative Deepening Search with an example and compare these algorithms.

3.2.2 Informed Search

3.2.2.1 Heuristics

21. What is a heuristic? Explain its role in informed search algorithms.
22. Provide examples of heuristics for the **8-puzzle**.

3.2.2.2 Greedy Best-First Search

23. Explain the Greedy Best-First Search algorithm.
24. How does Greedy Best-First Search use heuristics? Illustrate with an example.

*3.2.2.3 A Search**

25. Explain the A* Search algorithm and its evaluation function f(n)=g(n)+h(n)
26. Using a suitable example, illustrate the steps of A* Search. How is A* better than Best First Search?
27. Using the Euclidean distance heuristic, solve the **Water Jug Problem** using A* algorithm.

3.3 Local Search and Optimization Problems

3.3.1 Hill-Climbing Search

28. State and explain Hill Climbing Search using an example.

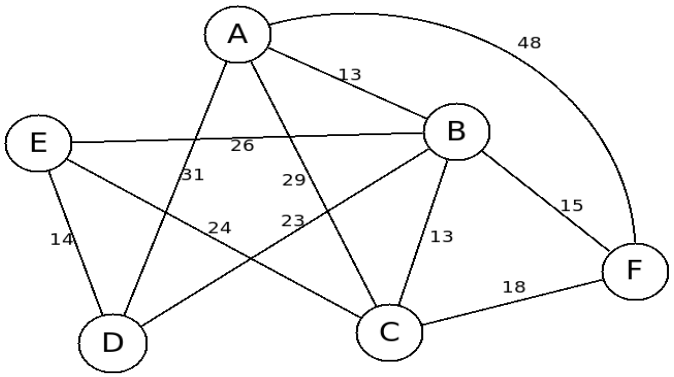
29. Plateaux, peaks, and ridges all cause problems for Hill Climbing Search. What does this mean? Explain.

3.3.2 Simulated Annealing

30. Explain the Simulated Annealing algorithm with an example.
31. Compare Simulated Annealing with Hill-Climbing Search.

3.3.3 Genetic Algorithms

32. Discuss the Genetic Algorithm with an example.
33. Illustrate the steps of Genetic Algorithm in solving an optimization problem, such as the **Traveling Salesman Problem**.



3.3.4 Gradient Descent

34. Explain Gradient Descent and its application in optimization problems.
35. Compare the use of Gradient Descent in linear regression and neural networks.

3.4 Adversarial Search and Game Playing

3.4.1 Minimax Algorithm

36. Explain the principle of Minimax Search using the example of a **Tic-Tac-Toe** game.

3.4.2 Alpha-Beta Pruning

37. Explain Alpha-Beta Pruning Search. What are the advantages of Alpha-Beta Pruning over Minimax Search?
38. In the given figure, use Minimax Search with Alpha-Beta Pruning to decide the next move by the MAX player from node A.

Game of NIM

39. The game of NIM is played as follows: Two players alternately remove one, two, or three coins from a stack initially containing five coins. The player who picks up the last coin loses.

    - Draw the full game tree.

- Show that the player who has the second move can always win.
- Execute the Alpha-Beta procedure on the game tree. How many terminal nodes are examined?

3.5 Constraint Satisfaction Problems (CSPs)

3.5.1 Representation of CSPs

40. What are Constraint Satisfaction Problems (CSPs)?
41. Explain the representation of CSPs using variables, domains, and constraints.
42. Model the **n-queens problem** as a CSP.

3.5.2 Search Algorithms for CSPs

43. Describe the Backtracking Search algorithm for CSPs.
44. Explain constraint propagation and its role in CSPs.
45. Trace the constraint satisfaction procedure for solving the **Cryptarithmatic Problem**:
    - **BASE + BALL = GAMES**

3.5.3 Optimization Technique: Min-Conflicts Heuristic

46. Consider the **Map Coloring Problem**, where a given map is to be colored using three colors in a manner so that no neighboring blocks contain the same color. Show that the CSP is satisfiable using the Min-Conflicts Heuristic.

Additional Problem-Solving and Critical Thinking Questions

1. Compare BFS, DFS, and IDS in terms of space and time complexity for a given search tree.
2. Formulate the **Water Jug Problem** and solve it using BFS or DFS.
3. Discuss heuristic search in terms of the **Traveling Salesman Problem**.
4. Explain the limitations of Hill-Climbing Search and how Simulated Annealing overcomes them.
5. Describe a real-world application of adversarial search and explain how Minimax or Alpha-Beta Pruning could be implemented.
6. For the **Sudoku Puzzle**, explain how CSP techniques can be used to find a solution.