**SRM Institute of Science and Technology**

Mode of Exam

**OFFLINE**

**SET C**

**College of Engineering and Technology**

**School of Computing**

**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE**

SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamil Nadu

**Academic Year: 2023 - 24 (EVEN)**

**Test: CLAT- 2** **Date: 02.04.2023**

**Course Code & Title: 18CSC305J / Artificial Intelligence** **Duration:** **1 hr 40 mts**

**Year & Sem:** **III / VI** **Max. Marks:** **50**

**Course Articulation Matrix: *(to be placed)***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S.No. | Course Outcome | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| 1 | CO1 | 3 | 2 | 3 | - | - | - | - | - | - | - | - | - |
| 2 | CO2 | 3 | 2 | 3 | - | - | - | - | - | - | - | - | - |
| 3 | CO3 | 2 | 3 | 3 | - | - | - | - | - | - | - | - | - |
| 4 | CO4 | 2 | 3 | 2 | - | - | - | - | - | - | - | - | - |
| 5 | CO5 | 2 | 3 | 3 | 2 | - | - | - | - | - | - | - | - |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part - A**  (5 x 10 = 50 Marks)  Instructions: Answer any 5 Questions | | | | | | |
| Q. No | Question | Marks | BL | CO | PO | PI Code |
|  | Solve the given blocks world problem using the hill climbing algorithm.  Start Goal    Answer: Global heuristics:  For each block that has the correct support structure: + 1  For each block that has the wrong support structure: - 1  The following is a  basic outline of how hill climbing can be applied to the Block World Problem:  State Representation: Represent the current state of the block world. This could be done using a data structure such as a list or a graph, where each block is represented along with its current position.  Define the Objective Function: In the context of the Block World Problem, the objective function would measure how close the current state is to the goal state. This could be the number of blocks that are in the correct position or some other measure of distance.  Generate Neighbors: Define the possible moves that can be made from the current state. In the Block World Problem, this would involve moving one block at a time to an adjacent position.  Evaluate Neighbors: For each neighbor generated, evaluate its score using the objective function.  Select Next Move: Choose the neighbor with the highest score as the next move.  Repeat: Continue generating neighbors and selecting moves until either the goal state is reached or no further improvements can be made.  Local Optima: Hill climbing can get stuck in local optima, where it cannot find a better solution because all neighboring states have lower scores. To address this, you can employ techniques like random restarts or simulated annealing.  Termination: The algorithm terminates when it reaches the goal state or when a predefined number of iterations have been performed.  Given the objective function values:  Block A: -3  Block D: -2  Block C: -1  Block B: 0  According to the objective function,  each block receives a score of +1 for every correct block below it. Let's  recalculate:  For the final state "abcd":  Sample moves:  1.Move Block D to the floor. New state: "ACB | D" (Objective Function Value: -3)  Objective Function Value:  Block A: -3 (No blocks below it)  Block D: 0 (On the floor)  Block C: 0 (No blocks below it)  Block B: 0 (No blocks below it)  Total: -3 + 0 + 0 + 0 = -3  2.Move Block C to the floor. New state: "AB | D | C" (Objective Function Value: +2)  Objective Function Value:  Block A: +1 (Block B is correctly below it)  Block D: 0 (On the floor)  Block C: +1 (No blocks below it)  Block B: +1 (Block A is correctly below it)  Total: +1 + 0 + 1 + 1 = +3  Block A: +3 (Block D is correctly below it)  Block B: +2 (No blocks below it)  Block C: +1 (Block D is correctly below it)  Block D: 0 (No blocks below it) | 10 | 3 | 2 | 3 | 3.2.1 |
|  | Implement the AO\* search algorithm on the provided tree structure. How do the branching factor and depth of the provided tree structure impact the implementation and efficiency of the AO\* search algorithm? Briefly explain. | 10 | 3 | 2 | 3 | 3.2.1 |
|  | Elucidate two main phases involved in a typical bidirectional search algorithm. How does it differ from traditional uninformed search algorithms like Breadth-First Search? In what scenarios might a bidirectional search be advantageous compared to a unidirectional search? Solve the given graph using a bidirectional search technique. Calculate the search complexity for the given graph.  Each iteration should be precisely elaborated stepwise using an open and closed list of nodes with a neat sketch.  Answer:  Bidirectional search tackles pathfinding in graphs with a clever twist. Unlike traditional uninformed searches that explore one direction from the starting point, bidirectional search operates in two phases (2 marks)  Phase 1: Expanding Fronts  1.  Forward Search: The algorithm initiates a search from the starting node,expanding outwards like Breadth-First Search (BFS). It explores neighboring nodes, building a frontier of nodes yet to be visited.  2.  Backward Search: Simultaneously, a second search starts from the goal node,expanding inwards. It explores neighboring nodes in the opposite direction,building another frontier.  Phase 2: Convergence and  Termination  1.  Frontier Exploration: In each iteration, both forward and backward frontiers are explored. New nodes are added to their respective frontiers.  2.  Intersection Check: The algorithm constantly checks if any node from the forward frontier overlaps with a node in the backward frontier.  3.  Path Found: If an overlap (intersection) is found, it signifies a path  has been discovered between the start and goal nodes. The search terminates  successfully.  Bidirectional search stands out from traditional uninformed searches like BFS in a few key ways (2 marks)  1.  Dual Exploration: BFS systematically explores outward from the starting point until the goal is found. Bidirectional search tackles the problem from both ends, potentially reaching the goal much faster if the start and goal are close within the graph.  2.  Reduced Search Space: By exploring from both ends, bidirectional search effectively cuts the search space in half compared to BFS. This translates to faster execution, especially for graphs where the start and goal are positioned relatively close together.  When Two Heads Are Better Than One Bidirectional search shines in specific scenarios (2 mark)  1.  Maze Solving: In mazes where the goal is visible from the starting point,a bidirectional search can quickly find the solution by exploring towards each other.  2.   Game Pathfinding: Games often have well-defined maps. Bidirectional search can be used by AI agents to navigate between points, potentially finding optimal paths faster than traditional uninformed searches.  3.     Real-World Routing: In some routing applications where the destination is known beforehand (like finding a specific address), bidirectional search can be used alongside traditional routing algorithms to potentially find alternate routes faster.  Iteration Steps(4 marks)  Path exist between 0 and 14  Intersection at: 7  \*\*\*\*\*Path\*\*\*\*\*  0 4 6 7 8 10 14 | 10 | 3 | 2 | 3 | 3.2.1 |
|  | For the following example, write the knowledge base (KB) in propositional logic. Explain how the inference would occur.  Ram is an IT professional.  IT professionals are hardworking.  All hardworking people are wealthy.  All wealthy people pay heavy income tax.  Ans:  The knowledge base (KB) can be represented in propositional logic as follows:  P1: Ram is an IT professional. (IT\_Professional(Ram))  P2: IT professionals are hardworking. (Hardworking(IT\_Professional))  P3: All hardworking people are wealthy. (Wealthy(Hardworking))  P4: All wealthy people pay heavy income tax. (IncomeTax(Wealthy))  Inference would occur by combining these statements and applying logical rules. For example, if we want to infer whether Ram pays heavy income tax, we can use the following reasoning:  Ram is an IT professional (P1).  IT professionals are hardworking (P2).  All hardworking people are wealthy (P3).  Therefore, Ram is wealthy (from P1, P2, and P3).  All wealthy people pays heavy income tax (P4).  Therefore, Ram pays a heavy income tax (from the inference in steps 4 and P4). | 10 | 3 | 2 | 3 | 3.3.1 |
|  | For each of the following arguments, explain which rules of inference are used for each step. The universe is a set of people.  a. Every student in this class owns a personal computer. Everyone who owns a personal computer can use the Internet. Therefore, John, a student in this class, can use the Internet.  b. Everyone in this class owns a personal computer. Someone in this class has never used the Internet. Therefore, someone who owns a personal computer has never used the Internet.  Let  *C(x)* be "*x* is a student in this class,  " *P(x)* be "*x* owns a personal computer, " and  *I(x)* be "*x* can use the Internet."  The premises are *x* (*C*(*x*) *P*(*x*)), *x* (*P*(*x*) *I*(*x*)) and *C*(John).  Using universal instantiation and the first premise,  *C*(John) *P*(John) follows.  Applying modus ponens to this conclusion and the last premise,  *P*(John) follows.  Using universal instantiation and the second premise,  *P*(John) *I*(John) follows.  From the last two conclusions by modus ponens  *I*(John) follows.  Hence from this conclusion and the last premise by conjunction *C*(John) *I*(John) follows.    b. "Everyone in this class owns a personal computer. Someone in this class has never used the Internet. Therefore, someone who owns a personal computer has never used the Internet."    Let  *C(x)* be "*x* is a student in this class,  " *P(x)* be "*x* owns a personal computer, " and  *I(x)* be "*x* has used the Internet."  The premises are  *x* (*C*(*x*) *P*(*x*)) and  *x* (*C*(*x*) *I*(*x*)).  Using existential instantiation and the second premise  *C*(*d*) *I*(*d*) for some person *d*.  Hence by simplification *C*(*d*).  For that person *d*, by universal instantiation and the first premise *C*(*d*) *P*(*d*). >  From this and the previous conclusion, by modus ponens *P*(*d*).  Also from *C*(*d*) *I*(*d*) by simplification *I*(*d*).  Hence by conjunction *P*(*d*) *I*(*d*).  Hence by existential generalization *x* (*C*(*x*) *I*(*x*)). | 10 | 3 | 3 | 3 | 3.3.1 |
|  | Using the Semantic network, represent different players in the Indian Cricket Team and their properties. Write a query to determine the most prolific batsman on Australian soil using this semantic network.  ANS: Creating knowledge base using the semantic network representation of different players in the Indian Cricket Team and their properties. Relevant statements in propositional logic to determine the most prolific batsman on Australian soil.  **Virat Kohli:**  Role: Batsman, Captain  Properties:  Tremendous run in 2018.  Highest T20I score in Australia: 90 not out (Adelaide, 2016).  Man of the Match awards in T20Is against Australia.  Key player in India’s 3-0 T20I series victory in Australia (2016).  **Rohit Sharma:**  Role: Batsman, Vice-Captain  Properties:  Pale performance in T20Is on Australian soil.  Played six matches, scoring 151 runs at an average of 30.20 and a strike rate of 132+.  Two fifties, highest score of 60.  Now, let’s express some relevant statements in propositional logic:  Prolific Batsman: Predicate Prolific(x) to represent that player x is a prolific batsman.  On Australian Soil: Predicate AustralianSoil(x) to represent that player x has played on Australian soil.  Using these predicates, we can express the following statements:  Relevant statements in propositional logic to determine the most prolific batsman on Australian soil:  **Prolific Batsman:** We define a predicate Prolific(x) to represent that player x is a prolific batsman.  **On Australian Soil:** We define a predicate AustralianSoil(x) to represent that player x has played on Australian soil.  Using these predicates, we can express the following statements:  1. Virat Kohli is a prolific batsman:  Prolific(Kohli)  2. Rohit Sharma is not a prolific batsman:  ¬Prolific(Sharma)  3. Virat Kohli has played on Australian soil:  AustralianSoil(Kohli)  4. Rohit Sharma has played on Australian soil:  AustralianSoil(Sharma)  Combine these statements to determine the most prolific batsman on Australian soil:  If a player is both prolific and has played on Australian soil, they are the most prolific batsman on Australian soil:  Prolific(Kohli)∧AustralianSoil(Kohli)  Therefore, based on this semantic network, Virat Kohli is the most prolific batsman on Australian soil among these two players. | 10 | 3 | 3 | 3 | 3.3.1 |
|  | After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a serious disease and that the test is 99% accurate (i.e., the probability of testing positive when you do have the disease is 0.99, as is the probability of testing negative when you don’t have the disease). The good news is that this is a rare disease, striking only 1 in 10,000 people of your age. Why is it good news that the disease is rare? What are the chances that you actually have the disease?  ANS:  To calculate the chances that you actually have the disease, we can use Bayes' theorem. Let's denote:  D: Having the disease  +: Testing positive for the disease  We are given:  P(+∣D)=0.99 (The probability of testing positive given that you have the disease)  P(D)=1/10,000 (The prior probability of having the disease)  P(+∣¬D)=0.01 (The probability of testing positive given that you don't have the disease)  We want to find:  P(D∣+) (The probability of actually having the disease given that you tested positive)  Using Bayes' theorem:  P(D∣+)= P(+∣D)×P(D)/ P(+)  ​We can calculate P(+) using the law of total probability:  P(+)=P(+∣D)×P(D)+P(+∣¬D)×P(¬D)  P(+)=(0.99× (1/10,000)) + (0.01× (9,999/10,000))  ​P(+)=0.000099+0.009999  P(+)=0.010098  Now, substituting back into Bayes' theorem:  P(D∣+)= (0.99× (1/10,000))/ 10.010098  P(D∣+)≈ 0.000099/0.010098  P(D∣+)≈0.009801  So, the probability that you actually have the disease given that you tested positive is approximately 0.9801%, which is still quite low despite testing positive. | 10 | 3 | 3 | 3 | 3.3.1 |