Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology, Pune-48
(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Department of Artificial Intelligence and Data Science

(AY-2023-24)
Semester-V

Artificial Intelligence Laboratory Manual

(ADUA31201)

Class:- TY(B.Tech)

Pattern: 2020

Examination Scheme:

Practical: 2 Hrs/week
OR: 25 Marks

---------- **********----------

Prepared By

Prof. Vijaykumar R. Ghule
[Course Coordinator]

---------- **********----------

---------- **********----------

List of Assignments: (Python)

1	Write a program for Automatic Nought and Crosses using random number
2	Two jugs are having capacity 4 and 3 respectively. Both the jugs do not have markings on them to measure smaller quantities. Measure 2 litres of water using the two jugs
3	Given an input n, print a n X n matrix consisting of numbers form 1 to n each appearing exactly once in each row and each column (Constraint satisfaction problem)
4	Write a program to implement breadth first search (Heuristic search).
5	Write a program to apply simulated annealing algorithm to a simple 1-D x^2 objective function with the bounds [-5,5]
6	A salesperson visits a number of cities exactly once and return to the first city, find the shortest route (Hill Climbing)
7	In a competition in which a contestant must choose one of three doors, one of which conceals a price. The show's host unlocks an empty door and asks the contestant if he wants to swap to the other door after the contestant has chosen one. The decision is whether to keep the current door or replace it with a new one. It is preferable to enter by the other door because the price is more likely to be higher. Write a program to come out from this ambiguity. (Inferencing using Bayesian network).
8	Write a program which uses Q-values to iteratively improve the behaviour of learning agent (Reinforcement learning).

---------- **********----------

---------- **********----------

Assignment No:-1

Title:- Automatic Nought and Crosses (Tic-Tac-Toe) Using Random Number

Problem Statement:-Write a program for Automatic Nought and Crosses using random number

Objective: Develop a program that allows two players to play the game of Noughts and Crosses (Tic-Tac-Toe) against each other. The computer will make random moves for one of the players.

Instructions:

- Create a 3x3 grid for the game.
- Alternate turns between the two players, one of which will be the computer.
- The computer's move should be chosen randomly from the available empty spaces.
- Display the updated grid after each move.
- Determine the winner or declare a draw according to the game's rules.

Algorithm Steps:-

1. Create a 3x3 grid to represent the Tic-Tac-Toe board.
2. Initialize a variable to keep track of the current player (X or O).
3. Repeat the following steps until the game is over:
  a. Display the current state of the board.
  b. If it's the human player's turn (X), prompt them to make a move.
    - Validate the input to ensure it's a valid empty cell.
    - Place an X in the chosen cell.
  c. If it's the computer player's turn (O):
    - Generate a random number between 0 and 8 (inclusive) to represent a cell index.
    - Check if the chosen cell is empty, if not, generate a new random number.
    - Place an O in the chosen cell.
  d. Check for a win or a draw condition:
    - If the current player has won, display the winner and end the game.
    - If the board is full and no player has won, display a draw message and end the game.
  e. Switch the current player for the next turn (X to O or O to X).

4. Display the final state of the board.
5. Display the winner or a draw message based on the game outcome.
6. End the game.

---------- **********----------

---------- **********----------

Assignment No:-2

Title:- Measuring 2 Litres Using Two Jugs

Problem Statement:-Two jugs are having capacity 4 and 3 respectively. Both the jugs do not have markings on them to measure smaller quantities. Measure 2 litres of water using the two jugs

Objective: Implement a solution to measure exactly 2 litres of water using two jugs with capacities 4 and 3 liters, respectively. The jugs have no markings for measuring smaller quantities.

Instructions:

- Define functions to represent filling, emptying, and pouring water from one jug to another.
- Develop a strategy to achieve the desired 2 liters using the jugs.
- Execute the strategy and print the steps taken to achieve the result.

Algorithm Steps:-

1. Initialize the capacities of the two jugs: Jug A (4 liters) and Jug B (3 liters).
2. Create a list to store the states of the jugs, representing the amount of water in each jug.
3. Repeat the following steps until you successfully measure 2 liters of water:
   a. Fill Jug A to its full capacity (4 liters).
   b. Pour water from Jug A into Jug B until Jug B is full or Jug A is empty.
   c. If Jug A is now empty, pour any remaining water from Jug B into Jug A.
   d. Jug B now contains the amount of water poured from Jug A in step b.
   e. Empty Jug B.
   f. Pour the remaining water from Jug A into Jug B.

4. At this point, Jug B will contain exactly 2 liters of water.

5. Display the final state of the jugs, showing that Jug B contains 2 liters of water.

6. End the algorithm.

---------- **********----------

---------- **********----------

Assignment No:-3

Title:- Constraint Satisfaction Problem - Magic Square Generator

Problem Statement:- Given an input n, print a n X n matrix consisting of numbers form 1 to n each appearing exactly once in each row and each column (Constraint satisfaction problem)

Objective:- Create a program that generates an "n x n" matrix with numbers from 1 to n, where each number appears exactly once in each row and each column.

Instructions:-

- Implement a backtracking algorithm to fill in the matrix while satisfying the constraints.
- Ensure that no two numbers repeat in the same row or column.
- Display the generated matrix as the output.

Algorithm Steps:-

1. Read the input value of n, which represents the size of the matrix.

2. Create an "n x n" matrix to store the numbers.

3. Initialize an array of size n, called "usedNumbers", to keep track of which numbers have been used in each row.

4. Repeat the following steps for each row (i):
   a. Initialize the usedNumbers array to all zeros.
   b. Repeat the following steps for each column (j):
      - Generate a random number between 1 and n that hasn't been used in the current row or column.
      - Assign the generated number to the matrix[i][j].
      - Mark the used number as used in the usedNumbers array.

5. Print the generated matrix.

6. End the algorithm.

---------- **********----------

---------- **********----------

Assignment No:-4

Title:- Breadth-First Search Implementation

Problem Statement:-Write a program to implement breadth first search (Heuristic search).

Objective:- Write a program to implement the Breadth-First Search algorithm for graph traversal.

Instructions:-

- Create a graph representation (adjacency list or matrix).
- Implement the BFS algorithm using a queue to keep track of nodes.
- Traverse and print the nodes in BFS order.

Algorithm Steps:-

1. Read the input: graph (represented as an adjacency list or matrix), start node, goal node.

2. Create an empty queue data structure for BFS.

3. Enqueue the start node into the queue.

4. Create a set or array to keep track of visited nodes.

5. While the queue is not empty:
   a. Dequeue a node from the front of the queue.
   b. If the dequeued node is the goal node, terminate and return the path.
   c. If the dequeued node is not in the visited set:
      - Mark the node as visited.
      - Enqueue all unvisited neighbors of the dequeued node into the queue.
      - Assign the path to the neighbors to include the current dequeued node.

6. If the queue becomes empty and the goal node is not found, return "Goal not reachable."

7. End the algorithm.

---------- **********----------

---------- **********----------

Assignment No:-5

Title:- Simulated Annealing for Objective Function Optimization

Problem Statement:- Write a program to apply simulated annealing algorithm to a simple 1-D x^2 objective function with the bounds [-5,5]

Objective:- Develop a program that applies the simulated annealing algorithm to optimize the simple 1-D x^2 objective function within the bounds [-5, 5].

Instructions:-

- Define the objective function: f(x) = x^2.
- Implement the simulated annealing algorithm with appropriate temperature schedule.
- Optimize the objective function within the given bounds and print the result.

Algorithm Steps:-

1. Initialize Parameters:
- Set the initial temperature (T) for the annealing process.
- Define the cooling rate (alpha), which controls how fast the temperature decreases.
- Choose the number of iterations or steps per temperature (num_steps).
- Set the initial solution as a random value within the bounds [-5, 5].

2. Define the Objective Function:
- Define the 1-D objective function you want to optimize, in this case, f(x) = x^2.

3. Initialization:
- Initialize the current solution as the initial random value.
- Evaluate the objective function at the current solution to get the current energy (E_current).
-

4.Annealing Loop:

A) Start a loop that iterates over different temperatures (T), decreasing the temperature using the cooling rate (alpha) in each iteration.

B) For each temperature iteration:

1. Run the following loop for num_steps:
   - Generate a random neighbor solution by perturbing the current solution.
   - This can be done by adding a random value within a certain range to the current solution.
   - Clip the neighbor solution to ensure it stays within the bounds [-5, 5].
   - Evaluate the objective function at the neighbor solution to get the neighbor's energy (E_neighbor).

- Calculate the energy difference (delta_E) between E_neighbor and E_current.
- If delta_E is negative (neighbor solution is better), accept the neighbor solution and update E_current and the current solution.
- If delta_E is positive (neighbor solution is worse), accept the neighbor solution with a probability of exp(-delta_E / T). This step introduces randomness and allows the algorithm to escape local optima.

C) After the num_steps iterations at the current temperature, the temperature is decreased using the cooling rate (alpha).

5.**Termination Condition:**

The annealing process can terminate based on various conditions, such as a predefined number of temperature iterations or when the temperature becomes very low (close to zero).

6. Output: The final solution found by the algorithm is the optimized value of x that corresponds to the global or near-global minimum of the objective function.

---------- **********----------

---------- **********----------

Assignment No:-6

Title:-Hill Climbing for Traveling Salesperson Problem

Problem Statement: A salesperson visits a number of cities exactly once and return to the first city, find the shortest route (Hill Climbing)

Objective: Write a program to find the shortest route for a salesperson visiting a number of cities exactly once and returning to the first city using the Hill Climbing algorithm.

Instructions:

- Define a representation for cities and distances between them.
- Implement the Hill Climbing algorithm to optimize the route.
- Print the optimized route and total distance traveled.

Algorithm Steps:-

1. Read the input: number of cities, distance matrix between cities.

2. Randomly generate an initial solution, which is a permutation of the cities representing the order of visit.

3. Initialize the current solution as the initial solution.

4. Initialize the current distance as the total distance traveled in the initial solution.

5. Repeat the following steps until a stopping criterion is met (e.g., a certain number of iterations or no improvement in distance):
  a. Select a pair of cities to swap in the current solution.
  b. Calculate the new distance after swapping the selected cities.
  c. If the new distance is shorter than the current distance:
    - Update the current solution with the swapped cities.
    - Update the current distance.
  d. Else, backtrack and try another pair of cities to swap.

6. Return the best solution found.

7. End the algorithm.

---------- **********----------

---------- **********----------

Assignment No:-7

Title:-Bayesian Network for Ambiguity Resolution

Problem Statement: In a competition in which a contestant must choose one of three doors, one of which conceals a price. The show's host unlocks an empty door and asks the contestant if he wants to swap to the other door after the contestant has chosen one. The decision is whether to keep the current door or replace it with a new one. It is preferable to enter by the other door because the price is more likely to be higher. Write a program to come out from this ambiguity. (Inferencing using Bayesian network).

Objective: Develop a program to resolve the ambiguity of whether to switch doors in a game show scenario using Bayesian network inference.

Instructions:

- Define a Bayesian network representing the given scenario.
- Implement the inference algorithm to make the decision on whether to switch doors.
- Provide the recommendation based on Bayesian network probabilities.

Algorithm Steps :-

1. Modeling the Bayesian Network:

a). Define nodes:

- C: Contestant's initial choice (Node with three states: Door 1, Door 2, Door 3).
- H: Location of the prize (Node with three states: Behind Door 1, Behind Door 2, Behind Door 3).
- U: Unlocked door by the host (Node with two states: Unlocked Door, Not Unlocked Door).
- S: Contestant's swap decision (Node with two states: Swap, No Swap).

b).Define conditional probabilities:

- P(C) - Uniform distribution (1/3 for each choice).
- P(H) - Uniform distribution (1/3 for each location).
- P(U|H, C) - If host reveals an empty door, P(U = Unlocked Door | H, C) = 1. If the prize is chosen, P(U = Not Unlocked Door | H, C) = 1.
- P(S|C, U) - If the door is unlocked, P(S = Swap | C, U) = 1. If the door is not unlocked, P(S = No Swap | C, U) = 1.

2. Inference:

- Given that the host has unlocked a door (U = Unlocked Door), calculate the posterior probabilities for the prize location (H) based on the initial choice (C).
    - ◆P(H | U, C) ∝ P(U | H, C) * P(H) * P(C)
- Calculate the posterior probabilities for the contestant's swap decision (S) based on the initial choice (C) and the unlocked door (U).
    - ◆P(S | C, U) ∝ P(U | H, C) * P(S | C, U)

3. Decision Making:

- Calculate the expected value of the prize location if the contestant chooses to swap (E[H | U, C, S = Swap]) and if the contestant chooses not to swap (E[H | U, C, S = No Swap]).

- Compare the expected values and make the decision:
    - If E[H | U, C, S = Swap] > E[H | U, C, S = No Swap], then advise the contestant to swap the door.
    - Otherwise, advise the contestant to keep the current door.

4.Output:

Output the advice to the contestant whether to swap or not based on the comparison of expected values.

5. Implementation:

You can use libraries like **"pgmpy"** (Probabilistic Graphical Models using Python) to create and manipulate Bayesian networks, perform inference, and calculate probabilities.

---------- **********----------

---------- **********----------

Assignment No:-8

Title:- Reinforcement Learning with Q-Values

Problem Statement:-Write a program which uses Q-values to iteratively improve the behaviour of learning agent (Reinforcement learning).

Objective:- Write a program that uses Q-values to iteratively improve the behavior of a learning agent.

Instructions:-

- Define the state space, action space, and rewards for a given problem.
- Initialize Q-values and implement the Q-learning algorithm.
- Train the agent over a number of episodes and observe the Q-values converge.
- Demonstrate the agent's improved behavior by selecting actions based on Q-values.

Algorithm Steps:-

1.Initialization:

- Initialize the Q-values for all state-action pairs to arbitrary values or zeros.
- Set the learning rate (alpha) to control the impact of new information.
- Set the discount factor (gamma) to balance immediate and future rewards.
- Define the number of episodes (iterations) for training.

2. Training Loop:

- For each episode:
  - Reset the environment to the initial state.
  - Repeat until the episode is completed:
    - Choose an action based on exploration-exploitation strategy (e.g., epsilon-greedy).
    - Take the chosen action and observe the new state and reward.
    - Update the Q-value of the current state-action pair using the Q-learning update rule:

Code:-

```
Q(s, a) = (1 - alpha) * Q(s, a) + alpha * (reward + gamma * max(Q(s', a')))
```

Where:

s: Current state
a: Chosen action
s': Next state after taking action a
reward: Reward received after taking action a

alpha: Learning rate
gamma: Discount factor

3. Exploration-Exploitation:

During action selection, balance exploration and exploitation using an epsilon-greedy strategy or another exploration technique.

- With probability epsilon, choose a random action.
- With probability 1 - epsilon, choose the action with the highest Q-value.

4.Termination:

Repeat the training loop for the specified number of episodes.

5. Policy Extraction:

After training, the Q-values can be used to extract a policy for the agent.

- For each state, choose the action with the highest Q-value as the recommended action.

6. Evaluation:

Evaluate the performance of the learned policy in the environment to measure its effectiveness and efficiency.

7. Iterative Improvement:

If needed, continue to refine the Q-values through more episodes of training or by adjusting hyperparameters.

Conclusion :-This algorithm outlines the basic steps for using Q-values to iteratively improve the behavior of a learning agent through reinforcement learning using the Q-learning algorithm. It's important to note that while this algorithm provides a general framework, the success of the learning process greatly depends on appropriate parameter tuning, exploration strategy, and the design of the reward structure.

----------**********----------

Note:- Remember to thoroughly test each program, provide appropriate comments, and document your approach and results in the lab report. Good luck with your assignments!