F-18 AI Past Paper Solution

Qno.1(a) Describe uniform cost search using an example

Ans:
Uniform Cost Search (UCS) is a search algorithm used in computer science to find the lowest-cost path from a starting point to a goal. Let's break it down with a simple example:

Imagine you're planning a road trip from city A to city B, with several cities in between. Each road segment has a cost associated with it, representing things like distance, tolls, or travel time.

1. Starting Point: City A
2. Goal: City B

Now, you want to find the cheapest route. Uniform Cost Search works like this:

Step 1: Start at City A.
Step 2: Examine all the immediate roads (neighbors) from City A and calculate their costs.
Step 3: Choose the road with the lowest cost and move to that city.
Step 4: Repeat Steps 2-3 until you reach City B.

Let's say the costs are as follows (in arbitrary units):
A to C: 4
A to D: 2
D to B: 5
C to B: 3

Uniform Cost Search would choose the path A → D → B because the total cost (2 + 5 = 7) is the lowest among the options. It's like picking the most economical route at each step to ensure you reach your destination with the least overall cost.
Example
A simple example of a map with cities and roads, and we'll use Uniform Cost Search to find the cheapest path from the start to the goal.
Cities: A, B, C, D
Roads and Costs:
A to B: 4
A to C: 2
B to D: 5
C to D: 1

Starting Point: A
Goal: D

Uniform Cost Search Steps:
1. Start at A.
2. Examine neighbors: B (cost 4), C (cost 2).
3. Choose C (lowest cost) and move to C.
4. Examine neighbors of C: D (cost 1).
5. Choose D and reach the goal.

The path taken: A → C → D

So, using Uniform Cost Search, we found the cheapest path from A to D, which is A → C → D with a total cost of 2 + 1 = 3.

Qno.1(b) Describe depth first search algorithm.

Ans:

Depth-First Search (DFS) is a traversal algorithm used in computer science to explore and visit all the nodes in a graph or tree. Here's a simple explanation:

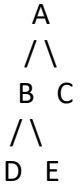
1.Start at the Initial Node: Begin at the starting point (usually called the root node).

2. Explore as Far as Possible Along Each Branch: Move as deep as you can along one branch until you reach a dead end (a node with no unvisited neighbors).

3. Backtrack: If you reach a dead end, backtrack to the most recent node with unexplored branches.

4. **Repeat:** Repeat steps 2 and 3 until all nodes are visited.

Let's illustrate this with a simple tree:



Starting at A, a Depth-First Search would visit nodes in this order: A, B, D, E, C.

1. Start at A.
2. Move to B.
3. Move to D.
4. Backtrack to B (no more unexplored branches).
5. Move to E.
6. Backtrack to B.
7. Backtrack to A.
8. Move to C.

The algorithm goes as deep as possible along one branch before backtracking. It's like exploring a maze by going as far as you can down one path before trying another.

Qno.2(a) Describe depth limited search.

Ans:

Depth-Limited Search is a variation of Depth-First Search with a twist—it sets a limit on how deep the algorithm can explore into the tree or graph. Here's a simple explanation:

1. Start at the Initial Node: Begin at the starting point (usually called the root node).

2. Explore as Far as Allowed: Move as deep as you can, but only up to a specified depth limit.

3. Backtrack When Needed: If you reach the depth limit or a dead end, backtrack to the most recent node with unexplored branches.

4. Repeat: Repeat steps 2 and 3 until all nodes within the depth limit are visited.

This is like exploring a maze but deciding beforehand how many steps you're willing to take in a single direction. If you reach that limit or a dead end, you turn back and try another path. It's a way to control the search depth in situations where you might want to avoid going too deep into a complex structure.

Qno.2(b) Differentiate between Informed and Uniformed search.

Ans:

The main difference between Informed Search and Uninformed Search lies in how much information they use to navigate and find a solution:

1. Uninformed Search:

What it does: Acts without specific knowledge about the problem.

How it works: Explores the search space blindly, typically using algorithms like Depth-First Search or Breadth-First Search.

Imagine: Like navigating a room in the dark, feeling your way around without knowing where things are.

2. Informed Search:

What it does: Utilizes specific information about the problem to make smarter decisions.

How it works: Uses heuristics or knowledge about the problem domain to guide the search. Examples include algorithms like A* (A-star).

Imagine: It's like having a map or some clues to help you find your way through the room; you make more informed choices based on available information.

Qno.3(a) Describe the bi-directional search.

Ans:

Bi-directional search is like having two people searching for each other in a maze. Here's a simple explanation:

1. Start at Both Ends:

Imagine you're in a maze, and you have one person starting at the entrance, and another person starting at the exit.

2. Search Towards Each Other:

Both individuals explore the maze, moving toward the center, or a meeting point.

3. Meet in the Middle:

- The goal is for both searchers to meet somewhere in the middle of the maze.

4. Efficiency:

Bi-directional search is often more efficient than a single search because it reduces the total number of nodes or paths that need to be explored.

It's like trying to find a friend in a building by starting from different entrances and moving towards each other to meet in the middle, saving time and effort.

Qno.3(b) What is inductive learning method?

Ans:

Inductive learning is a method in machine learning and cognitive science where general principles are derived from specific examples. It involves making generalizations based on observed patterns or instances. In simpler terms:

1. Start with Examples:
Begin with a set of specific examples or observations.

2. Identify Patterns:
Look for patterns, similarities, or trends within these examples.

3. Form General Rules:
Create general rules or principles that seem to explain or predict the observed patterns.

4. Apply Rules to New Instances:
Use the derived rules to make predictions or understand new instances that were not part of the original examples.

This method is akin to learning from specific cases and then developing general rules that can be applied to similar situations in the future. It's a way of building knowledge by generalizing from specific experiences. Inductive learning contrasts with deductive learning, where specific instances are derived from general principles.

Qno.4(a) Describe supervised learning method using an example.

Ans:

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning that it learns from input-output pairs. Here's a simple example:

Example: Handwriting Recognition

1. Dataset:
You have a dataset of handwritten digits (0 to 9) along with their corresponding labels (the actual digit each image represents).

2. Training:
You train a supervised learning algorithm, let's say a neural network, using this dataset. The algorithm learns to associate certain patterns in the images with the correct digit labels.

3. Testing:
After training, you test the algorithm with new, unseen handwritten digits. The algorithm predicts the digits based on what it learned during training.

4. Evaluation:
You compare the algorithm's predictions with the actual labels to measure its accuracy. For example, if the algorithm correctly identifies 95 out of 100 digits, its accuracy is 95%.

In this scenario, the algorithm is "supervised" because it learns from labeled examples, and the goal is to make accurate predictions on new, unseen data. It's like teaching a computer to recognize handwriting by showing it many examples and telling it what each example represents.

Qno.4(b) Name some real life applications of supervised learning.

Ans:

Here are some applications of supervised learning:

1. Email Spam Filtering:
Supervised learning helps your email filter distinguish between spam and regular emails by learning from examples you label as spam or not spam.

2. Voice Assistants:
Siri or Alexa use supervised learning to understand and respond to your voice commands by learning from various examples of spoken language.

3. Online Shopping Recommendations:
Websites like Amazon recommend products based on your past purchases and behavior, which is learned through supervised learning.

4. Digital Image Recognition:
Your phone's camera can recognize faces or objects in photos through supervised learning, where it learns from labeled images.

5. Language Translation:
Apps like Google Translate use supervised learning to translate text between languages by learning from translated pairs of text.

6. Credit Score Assessment:
Banks use supervised learning to predict creditworthiness by learning from historical data on individuals' credit behavior.

7. Medical Diagnosis:
Supervised learning helps in medical fields, predicting diseases or analyzing medical images based on labeled data.

8. Autonomous Vehicles:
Self-driving cars use supervised learning to recognize and respond to different objects and situations
on the road, learning from labeled examples.

In these examples, supervised learning involves the algorithm learning from labeled data to make predictions or provide assistance in various tasks.

Qno.5(a) Describe decision trees algorithm using an example.

Ans:

A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal nodetests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction.

Examples

Certainly! Let's consider two simple examples:

Example 1: Choosing an Outfit

1. Dataset:

Imagine you have a dataset of past outfit choices with two features: weather (Sunny, Rainy) and temperature (Warm, Cold).

Day	Weather	Temperature	Outfit
Day1	Sunny	Warm	T-shirt
Day2	Rainy	Cold	Jacket
Day3	Sunny	Hot	Shorts
Day4	Rainy	Cold	Sweater
Day5	Sunny	Warm	T-shirt

2. Decision Tree Building:

The decision tree might look like this:

- Weather
  - Sunny
    - Temperature
      - Warm: T-shirt
      - Hot: Shorts
  - Rainy
    - Cold: Jacket or Sweater

3. Making Predictions:

If it's a sunny day with warm temperature, the decision tree predicts "T-shirt."

Example 2: Buying a Laptop

1. Dataset:
Consider a dataset for buying laptops with features like brand (Dell, HP) and price range (Low, High).

Laptop	Brand	Price Range	Decision
Laptop1	Dell	Low	Buy
Laptop2	HP	High	Don't Buy
Laptop3	Dell	Low	Buy
Laptop4	HP	High	Don't Buy
Laptop5	Dell	High	Don't Buy

2. Decision Tree Building:
- The decision tree might look like this:
  - Brand
    - Dell
      - Price Range
        - Low: Buy
        - High: Don't Buy
    - HP
      - High: Don't Buy

3. Making Predictions:
If you are considering a Dell laptop with a low price range, the decision tree predicts "Buy."

In these examples, decision trees help make decisions based on features, learning patterns from past data to guide future choices.

Qno.5(b) What is the difference between classification and regression problem?

Ans:

The main difference between classification and regression problems lies in the type of output they produce:

1. Classification Problem:

Nature: In a classification problem, the output variable is a category or label. The goal is to predict which category or class the input belongs to.

Example: Predicting whether an email is spam or not (binary classification) or classifying images of animals into different species (multi-class classification).

2. Regression Problem:
Nature: In a regression problem, the output variable is a continuous value. The goal is to predict a quantity, typically a real number.
Example: Predicting house prices based on features like size and location, or estimating the temperature based on time of day.
In summary:
Classification: Deals with predicting categories or labels.
Regression: Deals with predicting a continuous value.

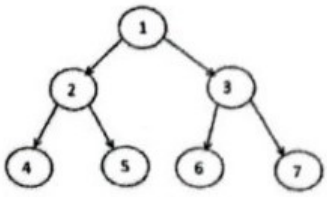
Imagine you're trying to predict whether an email is spam (classification) or trying to predict the actual temperature tomorrow (regression). The nature of the output (category or continuous value) determines whether the problem is a classification or regression problem.

19 AI Past Paper Solution

Qno.1(a) & Qno.1(b) same in F-18

Qno.2(a) same in F-18

Qno.2 (b) Determine the best path using Best First Search algorithm and a queue for the given graph.



Ans:

The best path using Best First Search algorithm and a queue for the given graph is:

A - C - E

The algorithm works by expanding the nodes of the graph in order of increasing distance from the starting node (A) until the goal node (E) is reached.

The following table shows the steps of the algorithm:

Step	Node	Queue	Visited
1	A	A	
2	C	C	A
3	E	E	A, C

Since node E is the goal node, the algorithm terminates.

Here is a Python implementation of the Best First Search algorithm:

Python

```
def best_first_search(graph, start_node, goal_node):
```

```
  """Performs a best first search on a graph.

  Args:
    graph: A graph represented as a dictionary of lists, where
each key is a node
      and each list is the neighbors of that node.
    start_node: The starting node.
    goal_node: The goal node.

  Returns:
    A list of nodes representing the path from the start node to
the goal node, or None
    if no path is found.
  """

  queue = []
  visited = set()

  queue.append((start_node, 0))

  while queue:
    node, cost = queue.pop(0)

    if node == goal_node:
      return [node]

    if node in visited:
      continue

    visited.add(node)

    for neighbor in graph[node]:
      queue.append((neighbor, cost + 1))

  return None
```

To use the algorithm, we simply pass the graph, start node, and goal node to the `best_first_search()` function. The function will return a list of nodes representing the path from the start node to the goal node, or None if no path is found.

For example, to find the best path from node A to node E in the given graph, we would use the following code:

Python

```
graph = {
  "A": ["C"],
  "C": ["A", "D", "E"],
  "D": ["C"],
  "E": ["C"]
}

start_node = "A"
goal_node = "E"

path = best_first_search(graph, start_node, goal_node)

print(path)
```

Output:

```
['A', 'C', 'E']
```

Qno.3(a) & (b) same in F-18
Qno.4(a) & (b) same in F-18
Qno.5(a) & (b) same in F-18