**✨ 1. Recursive DFS (Graph from CSV)**

**📚 Theory:**

Depth First Search (DFS) is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It uses recursion and a "visited" set to avoid cycles. DFS can be applied to find paths, detect cycles, or explore graph structures. The graph is first read from a CSV file into an adjacency list.

**❓ Viva Questions & Answers:**

* **Q:** What is DFS?  
  **A:** It’s a graph traversal that goes deep before backtracking.
* **Q:** What data structure is used internally?  
  **A:** Stack (call stack through recursion).
* **Q:** When is DFS preferred over BFS?  
  **A:** When depth is more important than breadth.
* **Q:** What are applications of DFS?  
  **A:** Cycle detection, path finding, solving puzzles.

**✨ 2. Non-Recursive DFS (Graph from User)**

**📚 Theory:**

Non-recursive DFS uses an explicit stack to simulate the recursion stack. Nodes are pushed and popped manually until all nodes are visited. This helps avoid recursion depth issues and gives more control over the order of traversal.

**❓ Viva Questions & Answers:**

* **Q:** How is non-recursive DFS implemented?  
  **A:** Using an explicit stack.
* **Q:** What if stack is not used?  
  **A:** Traversal would not happen correctly.
* **Q:** Advantages over recursive DFS?  
  **A:** Better memory control.
* **Q:** DFS traversal order?  
  **A:** Deepest nodes are visited first.

**✨ 3. BFS (Graph from User)**

**📚 Theory:**

Breadth First Search (BFS) explores the graph level-by-level using a queue. It starts from a given node, visits all its neighbors before moving to the next level. It is ideal for finding the shortest path in unweighted graphs.

**❓ Viva Questions & Answers:**

* **Q:** What data structure is used in BFS?  
  **A:** Queue.
* **Q:** BFS is best suited for which problems?  
  **A:** Finding shortest path in unweighted graphs.
* **Q:** BFS vs DFS?  
  **A:** BFS goes level-by-level; DFS goes deep first.
* **Q:** Can BFS detect cycles?  
  **A:** Yes, using visited tracking.

**✨ 4-7. Best First Search Variants**

**📚 Theory:**

Best First Search chooses the node with the lowest heuristic value to expand first. It uses a priority queue (min-heap) ordered by heuristic values. It’s faster but may not find the optimal path always unless heuristic is perfect.

**❓ Viva Questions & Answers:**

* **Q:** What is heuristic?  
  **A:** Estimate of cost to reach the goal.
* **Q:** Data structure used?  
  **A:** Priority queue.
* **Q:** Is Best First Search complete?  
  **A:** Yes, if search space is finite.
* **Q:** Is it guaranteed to find optimal solution?  
  **A:** No, unless heuristic is perfect.

**✨ 8-11. A\* Algorithm Variants**

**📚 Theory:**

A\* Algorithm finds the most cost-effective path by combining actual cost (g) and estimated cost (h). It’s optimal and complete when using an admissible heuristic. Priority queue is used with f(n) = g(n) + h(n).

**❓ Viva Questions & Answers:**

* **Q:** What is f(n) in A\*?  
  **A:** f(n) = g(n) + h(n).
* **Q:** What is admissible heuristic?  
  **A:** Never overestimates the cost.
* **Q:** Difference between A\* and Best First Search?  
  **A:** A\* uses both g and h; Best First uses only h.
* **Q:** Applications of A\*?  
  **A:** Games, pathfinding, GPS navigation.

**✨ 12. Fuzzy Set Operations**

**📚 Theory:**

Fuzzy sets allow partial membership values between 0 and 1. Operations are defined as:

* Union: max(membership values)
* Intersection: min(membership values)
* Complement: 1 - membership value

**❓ Viva Questions & Answers:**

* **Q:** What is a fuzzy set?  
  **A:** A set with degrees of membership.
* **Q:** How is union defined?  
  **A:** Maximum of memberships.
* **Q:** How is intersection defined?  
  **A:** Minimum of memberships.
* **Q:** Example application?  
  **A:** Control systems, AI decision-making.

**✨ 13-14. De Morgan’s Laws for Fuzzy Sets**

**📚 Theory:**

De Morgan's Laws show complement relationships:

* Complement of Union = Intersection of Complements
* Complement of Intersection = Union of Complements

**❓ Viva Questions & Answers:**

* **Q:** State De Morgan’s Laws?  
  **A:** Complement of union = intersection of complements.
* **Q:** Why important?  
  **A:** Simplifies complex fuzzy expressions.
* **Q:** In fuzzy logic, how complement is computed?  
  **A:** 1 - membership value.

**✨ 15-16. Two-Player Game (Computer Win/Lose using Min-Max)**

**📚 Theory:**

Min-Max algorithm chooses the best move assuming opponent also plays optimally. It maximizes minimum gain for one player and minimizes maximum loss for the other.

**❓ Viva Questions & Answers:**

* **Q:** What is Min-Max?  
  **A:** Decision-making algorithm for two-player games.
* **Q:** What is utility value?  
  **A:** The score value at terminal nodes.
* **Q:** Computer maximizes or minimizes in Min-Max?  
  **A:** Computer maximizes normally; minimizes if forced to lose.
* **Q:** Example games?  
  **A:** Tic-Tac-Toe, Chess, Nim.

**✨ 17-18. MLP with Random Weights**

**📚 Theory:**

MLP is a type of feedforward neural network with one or more hidden layers. Weights and biases are initialized randomly before training starts.

**❓ Viva Questions & Answers:**

* **Q:** What is an MLP?  
  **A:** A multilayer neural network.
* **Q:** Why random initialization?  
  **A:** To break symmetry among neurons.
* **Q:** Number of hidden layers here?  
  **A:** Two hidden layers.

**✨ 19-21. MLP with Activation (Sigmoid, ReLU, Tanh)**

**📚 Theory:**

Activations introduce non-linearity:

* Sigmoid: Smooth, bounded (0-1)
* ReLU: Rectified Linear Unit, faster convergence
* Tanh: Bounded (-1 to 1)

**❓ Viva Questions & Answers:**

* **Q:** Why activation functions?  
  **A:** To learn complex patterns.
* **Q:** Sigmoid vs ReLU?  
  **A:** Sigmoid saturates; ReLU faster but can "die".
* **Q:** Role of backpropagation?  
  **A:** To adjust weights using error gradient.

**✨ 22. Text Cleaning, Tokenizing, Spell Checking**

**📚 Theory:**

Preprocessing improves text by cleaning unwanted parts, converting to lowercase, tokenizing into words, removing stopwords, and correcting spelling mistakes.

**❓ Viva Questions & Answers:**

* **Q:** Why text preprocessing?  
  **A:** To prepare raw text for analysis.
* **Q:** What are stopwords?  
  **A:** Common words like 'the', 'and' etc.

**✨ 23. Text Cleaning, Stemming, Lemmatization, Trigrams**

**📚 Theory:**

Stemming cuts words roughly, Lemmatization finds root meaning. Trigrams are three consecutive words used for context.

**❓ Viva Questions & Answers:**

* **Q:** Difference between stemming and lemmatization?  
  **A:** Lemmatization gives real word roots.
* **Q:** What are trigrams?  
  **A:** Three-word sequences.

**✨ 24. One-Hot Encoding (Technical Files)**

**📚 Theory:**

One-hot encoding converts text into binary vectors showing presence or absence of words.

**❓ Viva Questions & Answers:**

* **Q:** What is one-hot encoding?  
  **A:** Binary representation of words.

**✨ 25. Bag of Words (Movie Reviews)**

**📚 Theory:**

Bag of Words model counts frequency of words ignoring their order.

**❓ Viva Questions & Answers:**

* **Q:** Limitation of Bag of Words?  
  **A:** Ignores word order and context.

**✨ 26. TF-IDF (Tourist Places)**

**📚 Theory:**

TF-IDF balances term frequency and inverse document frequency to highlight important words uniquely.

**❓ Viva Questions & Answers:**

* **Q:** Purpose of TF-IDF?  
  **A:** To find unique important words.

**1. Recursive DFS (Graph from CSV)**

**Theory**:

* DFS explores as deep as possible before backtracking.
* Implemented using recursion.
* Uses visited set to avoid visiting same node again.
* Graph is loaded from a CSV file as an adjacency list.
* Good for detecting cycles and connected components.

**Viva Questions**:

* Q1: What is DFS? ➔ Depth First Search for graph traversal.
* Q2: DFS uses which data structure? ➔ Stack (recursion).
* Q3: Where is DFS useful? ➔ Cycle detection, maze solving.
* Q4: Why use visited set? ➔ To prevent infinite loops.
* Q5: DFS time complexity? ➔ O(V + E).

**2. Non-Recursive DFS (Graph from User)**

**Theory**:

* DFS without recursion by manually using a stack.
* Reads graph edges from user input.
* Useful for controlling stack overflow issues.
* Similar traversal order to recursive DFS.
* Helps in understanding internals of recursion.

**Viva Questions**:

* Q1: What replaces recursion? ➔ Stack.
* Q2: Is result same as recursive DFS? ➔ Yes.
* Q3: When is non-recursive DFS better? ➔ Deep graphs.
* Q4: Stack operations used? ➔ Push, pop.
* Q5: DFS time complexity? ➔ O(V + E).

**3. BFS (Graph from User)**

**Theory**:

* BFS explores level-by-level.
* Uses a queue to explore all neighbors first.
* Reads graph edges from user input.
* Best suited for shortest path in unweighted graphs.
* Detects connected components.

**Viva Questions**:

* Q1: BFS uses which structure? ➔ Queue.
* Q2: Where is BFS used? ➔ Shortest path, broadcasting.
* Q3: BFS time complexity? ➔ O(V + E).
* Q4: BFS vs DFS? ➔ BFS: breadth first, DFS: depth first.
* Q5: Cycle detection possible? ➔ Yes.

**4-7. Best First Search Variants**

**Theory**:

* Greedy search using heuristic value only.
* Chooses the node with minimum heuristic h(n).
* Can be used in directed/undirected, weighted/unweighted graphs.
* Uses a priority queue.
* May not find the optimal path unless heuristic is good.

**Viva Questions**:

* Q1: What is heuristic? ➔ Estimate to goal.
* Q2: Complete or incomplete? ➔ Complete.
* Q3: Guaranteed optimal? ➔ No.
* Q4: Best First Search structure? ➔ Priority Queue.
* Q5: Example? ➔ Pathfinding in AI.

**8-11. A\* Algorithm Variants**

**Theory**:

* Combines cost from start g(n) and estimated cost to goal h(n).
* f(n) = g(n) + h(n).
* Uses a priority queue.
* Optimal if heuristic is admissible.
* Reads graph from CSV or user.

**Viva Questions**:

* Q1: What is g(n)? ➔ Actual cost to node.
* Q2: What is h(n)? ➔ Heuristic estimate.
* Q3: Why A\* better than Best First? ➔ Considers total cost.
* Q4: Optimal when? ➔ Admissible heuristic.
* Q5: Data structure? ➔ Priority Queue.

**12. Fuzzy Set Operations (3 sets)**

**Theory**:

* Fuzzy sets allow partial membership [0,1].
* Union = max(), Intersection = min(), Complement = 1 - value.
* Used in systems where boundaries are not crisp.
* Supports approximate reasoning.

**Viva Questions**:

* Q1: Fuzzy union rule? ➔ max(a, b).
* Q2: Fuzzy intersection rule? ➔ min(a, b).
* Q3: Complement rule? ➔ 1 - value.
* Q4: Where used? ➔ Washing machines, control systems.
* Q5: Is membership binary? ➔ No, partial.

**13-14. De Morgan’s Laws (Fuzzy Sets)**

**Theory**:

* De Morgan’s Laws for Fuzzy Sets:
  + Complement of Union = Intersection of Complements.
  + Complement of Intersection = Union of Complements.
* Proves logical consistency.

**Viva Questions**:

* Q1: State De Morgan Law 1. ➔ ~(A U B) = ~A ∩ ~B.
* Q2: State De Morgan Law 2. ➔ ~(A ∩ B) = ~A U ~B.
* Q3: Application? ➔ Simplifying logic expressions.
* Q4: Needed in Fuzzy Logic? ➔ Yes.
* Q5: Fuzzy vs Classical sets? ➔ Fuzzy allows degrees.

**15-16. Two-Player Game (Min-Max Algorithm)**

**Theory**:

* Min-Max chooses optimal moves assuming opponent also plays best.
* Maximizing player tries to maximize score.
* Minimizing player tries to minimize computer's score.
* Modified for forcing loss by reversing strategies.

**Viva Questions**:

* Q1: What is Min-Max? ➔ Algorithm for optimal moves.
* Q2: Two roles? ➔ Maximizer and Minimizer.
* Q3: Application? ➔ Tic-Tac-Toe, Chess AI.
* Q4: Game tree? ➔ All possible moves.
* Q5: Depth limit needed? ➔ Yes for large trees.

**17-18. MLP (Random Weights)**

**Theory**:

* MLP has input, hidden, and output layers.
* Weights and biases randomly initialized.
* No learning (backprop) — just forward pass.

**Viva Questions**:

* Q1: Why random weights? ➔ Break symmetry.
* Q2: Hidden layers in MLP? ➔ At least one.
* Q3: Activation function needed? ➔ Yes.
* Q4: Purpose of bias? ➔ Shifts activation.
* Q5: Forward pass? ➔ Input → Hidden → Output.

**19-21. MLP (Sigmoid, ReLU, Tanh)**

**Theory**:

* Activation functions bring non-linearity.
* Sigmoid: 0-1, smooth.
* ReLU: max(0, x), fast convergence.
* Tanh: -1 to 1, better for centered data.

**Viva Questions**:

* Q1: Why activation? ➔ Non-linearity.
* Q2: Sigmoid saturation? ➔ Yes, at extremes.
* Q3: ReLU dying neuron problem? ➔ Yes.
* Q4: Tanh better where? ➔ Centered outputs.
* Q5: Backpropagation purpose? ➔ Update weights.

**22. Text Preprocessing (Cleaning, Spell Check)**

**Theory**:

* Removes noise (punctuation, numbers).
* Lowercase conversion.
* Tokenizes text.
* Removes stopwords.
* Corrects spelling errors.

**Viva Questions**:

* Q1: What is text cleaning? ➔ Removing unwanted parts.
* Q2: Why lowercase? ➔ Standardization.
* Q3: Example stopwords? ➔ 'the', 'is'.
* Q4: Tokenization? ➔ Breaking into words.
* Q5: Spell correction tool? ➔ SpellChecker.

**23. Stemming, Lemmatization, Trigrams**

**Theory**:

* Stemming cuts suffixes crudely.
* Lemmatization uses dictionary root forms.
* Trigrams are sequences of 3 consecutive words.

**Viva Questions**:

* Q1: Stemming example? ➔ playing → play.
* Q2: Lemmatization example? ➔ better → good.
* Q3: Why trigrams? ➔ Capture local context.
* Q4: Which is more accurate: stem or lemma? ➔ Lemmatization.
* Q5: Use of trigrams? ➔ Text prediction.

**24. One-Hot Encoding (Technical files)**

**Theory**:

* Converts words into 0/1 vectors.
* Each word becomes an independent feature.
* Useful for machine learning models.

**Viva Questions**:

* Q1: What is one-hot encoding? ➔ Binary vector representation.
* Q2: Disadvantage? ➔ Sparse vectors.
* Q3: Advantage? ➔ Simple, interpretable.
* Q4: Used in? ➔ Text classification.
* Q5: Libraries? ➔ sklearn.

**25. Bag of Words (Movie Reviews)**

**Theory**:

* Counts word frequency.
* No importance to order.
* Simple but lacks context understanding.

**Viva Questions**:

* Q1: BoW meaning? ➔ Bag of Words.
* Q2: BoW matrix? ➔ Word frequency matrix.
* Q3: Context captured? ➔ No.
* Q4: When useful? ➔ Text classification.
* Q5: Alternatives? ➔ TF-IDF, Word Embeddings.

**26. TF-IDF (Tourist Places)**

**Theory**:

* TF-IDF = Term Frequency \* Inverse Document Frequency.
* Highlights important and unique words.
* Reduces common word dominance.

**Viva Questions**:

* Q1: TF meaning? ➔ Term Frequency.
* Q2: IDF meaning? ➔ Inverse Document Frequency.
* Q3: Why IDF? ➔ Penalize common words.
* Q4: TF-IDF library? ➔ sklearn.
* Q5: Use case? ➔ Text mining.