**Assignment for Search Strategies**

**Level 1 – Basic Understanding**

**🔹 Task 1: Define a Search Problem**

define:

* Initial state
* Possible actions
* Goal test
* Path cost

for

* Maze
* City map (from A to B)
* Robot vacuum cleaner

➡️ **Classify** if it is:

* Goal-based?
* Deterministic?
* Observable?

**🔹 Task 2: BFS vs DFS Comparison Table**

Fill in a table comparing:

* Data structure used
* Time and space complexity
* Completeness
* Optimality
* When to use

**🔹 Task 3: Manual BFS and DFS Tracing**

Draw a **binary tree (depth = 3)**.

For BFS:

* List visited nodes at each level
* Track queue at each step

For DFS:

* List order of node visits
* Show stack at each step

**🔁 Level 2 – Code Implementation**

**🔹 Task 4: Write BFS and DFS in Python**

Use an **adjacency list** representation.

* Find a goal node in a graph
* Trace visited nodes

Add:

* Print statement to show queue/stack
* Print visited nodes in order

**🔹 Task 5: Add Depth-Limited DFS (DLS)**

Modify your DFS code to support depth-limiting.

* Try depth = 2, 3
* Observe how it avoids infinite loops

**🔹 Task 6: Compare Time and Space Complexity**

For a binary tree of depth 5:

* Count nodes visited in BFS and DFS
* Estimate time complexity O(b^d)
* Estimate space complexity

**🧠 Level 3 – Problem Solving & Challenges**

**🔹 Task 7: Maze Solver with BFS and DFS**

Design a 2D grid maze (start to goal). Use:

* 0 for free cell, 1 for wall
* Start = (0,0), Goal = (n-1,n-1)

Implement:

* BFS to find shortest path
* DFS to find **any** path

Add visualization:

* Print grid with path traced

**🔹 Task 10: Design Your Own Search Problem**

Come up with a **real-world inspired** search problem. Examples:

* AI agent in a warehouse
* Navigation on a game map
* Package delivery route optimization

Define:

* State representation
* Action model
* Goal test
* Search strategy you'd use and why