🗂️ Graph Representation

Graph representation means how a graph is stored in memory so that we can perform operations like traversal, search, and updates efficiently

🔢 Adjacency Matrix

- Adjacency Matrix is a 2D array used to represent a graph.
- Rows and columns represent vertices.
- If there is an edge between vertex i and j, the cell contains 1 (or weight).
- If there is no edge, the cell contains 0.
- Works well for dense graphs (many edges).
- Simple to understand and implement.
- Consumes more memory even if graph has few edges.

- Time Complexity
- Check edge existence: O(1)
- Space complexity: O(V^2)

📋 Adjacency List

- Adjacency List represents a graph using an array of lists.
- Each vertex stores a list of connected vertices.
- Stores only existing edges, not unnecessary data.
- Best suited for sparse graphs (few edges).
- More memory efficient than adjacency matrix.
- Slightly complex to implement compared to matrix.

- Time Complexity
- Check edge existence: O(degree of vertex)
- Space complexity: O(V + E)

⚖️ Adjacency Matrix vs Adjacency List

- Adjacency Matrix uses more memory, Adjacency List is memory efficient.
- Adjacency Matrix is faster for edge lookup, Adjacency List is faster for iterating neighbors.
- Adjacency Matrix is ideal for small or dense graphs.
- Adjacency List is ideal for large or sparse graphs.
- Most real-world applications prefer Adjacency List.

🎯 Which One to Use?

Use Adjacency Matrix when:

- Graph is small
- Many edges exist
- Fast edge lookup is required

Use Adjacency List when:

- Graph is large
- Few edges exist
- Memory efficiency matters