Name: Anuska Nath
Dept: IT (UG3)
Section: A3
Roll: 002311001003

Assignment 2

Objective

To read both binary and numerical datasets from CSV files representing adjacency matrices, construct graphs based on them, and visualize the resulting graphs using Python. Binary datasets produce unweighted graphs, while numerical datasets yield weighted graphs.

Tools & Libraries Used

- NumPy – for handling matrix operations.
- CSV – to read data from CSV files.
- NetworkX – to construct and manage graph data structures.
- Matplotlib – for graph visualization.

a. Write a Python program to load and read a binary dataset from a CSV file and draw the corresponding graph considering the dataset as an adjacency matrix.

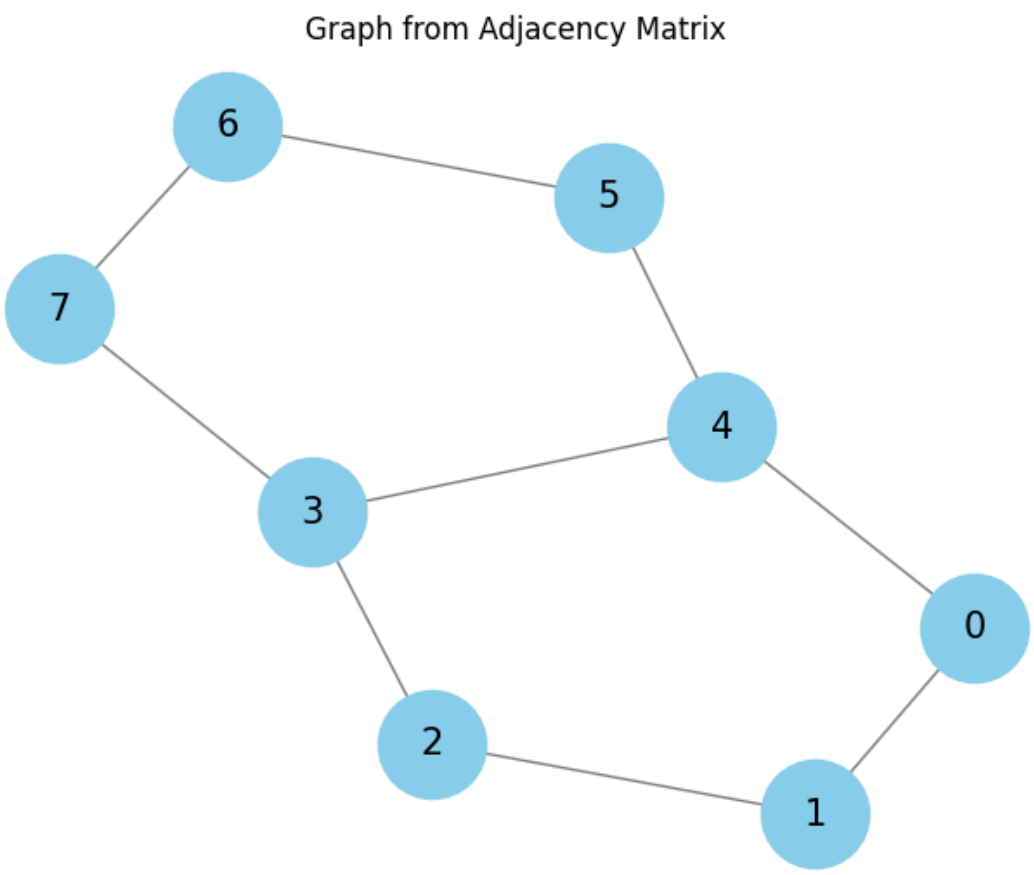
Binary Dataset as Adjacency Matrix:

Input:

- CSV file: binary_adjacency_matrix.csv
- Matrix values: Only 0 and 1
- Matrix size: 8×8

0	1	0	0	1	0	0	0
1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	0	1	0	0	1
1	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1
0	0	0	1	0	0	1	0

Output:



Implementation Details

1. Loading the Matrix

- **Function:** load_adjacency_matrix(csv_file)
- Reads the CSV file using csv.reader().
- Converts each row into a list of int values using map(int, row).
- The final matrix is stored as a NumPy array named adj_matrix.

2. Graph Creation

- **Function:** draw_graph(adj_matrix)
- Uses nx.from_numpy_array() to create the graph:
    - If the matrix is **symmetric**, an undirected graph is created using nx.Graph.
    - If the matrix is **not symmetric**, a directed graph is created using nx.DiGraph.

3. Graph Visualization

- Layout: nx.spring_layout(G) is used to compute positions for better spacing.
- Nodes: drawn using:
    - node_color='skyblue'
    - node_size=2000
    - font_size=15
- Edges: drawn in gray using edge_color='gray'
- Labels: Node numbers are displayed automatically via with_labels=True
- Title: "Graph from Adjacency Matrix" is shown with plt.title().

4. Execution

The main() function:

- Loads the adjacency matrix from adjacency_matrix.csv
- Prints the matrix to the console
- Calls draw_graph(adj_matrix) to render the graph

b. Do the same as mentioned above for a numerical dataset.

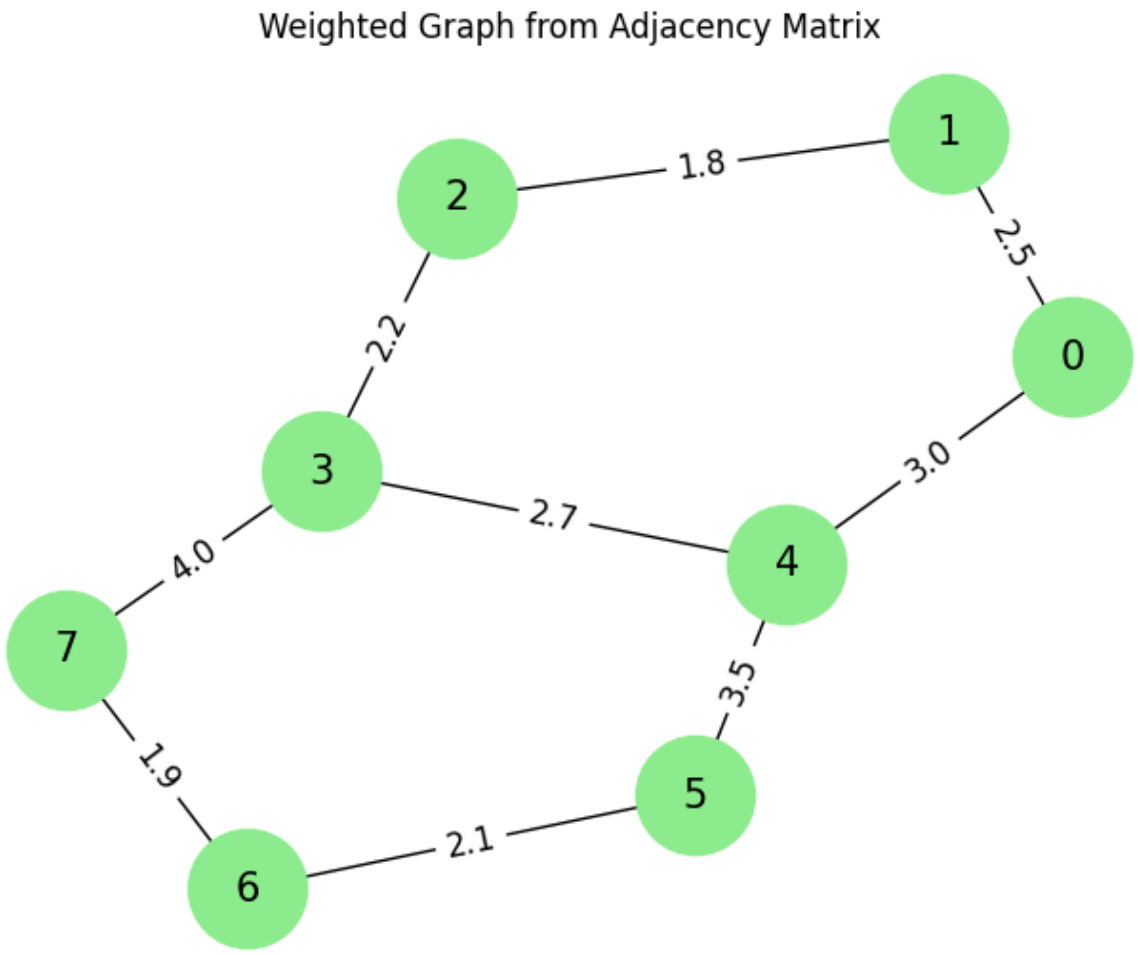
Numerical Dataset as Weighted Adjacency Matrix:

Input:

- CSV file: weighted_adjacency_matrix.csv
- Matrix values: Floating-point numbers (weights)
- Matrix size: 8×8

0	2.5	0	0	3.0	0	0	0
2.5	0	1.8	0	0	0	0	0
0	1.8	0	2.2	0	0	0	0
0	0	2.2	0	2.7	0	0	4.0
3.0	0	0	2.7	0	3.5	0	0
0	0	0	0	3.5	0	2.1	0
0	0	0	0	0	2.1	0	1.9
0	0	0	4.0	0	0	1.9	0

Output:



Implementation Details

1. Loading the Matrix

- Function: load_adjacency_matrix(csv_file)
- Opens the CSV file using Python's csv.reader.
- Each row is read and converted into a list of float values.
- All rows are combined into a NumPy array adj_matrix for graph processing.

2. Graph Creation

- Function: draw_weighted_graph(adj_matrix)

- Uses nx.from_numpy_array() to create the graph:
    - If the matrix is symmetric (np.allclose(adj_matrix, adj_matrix.T)), an undirected graph (nx.Graph) is created.
    - Otherwise, a directed graph (nx.DiGraph) is constructed.
- Edges with weight 0 are removed to represent the absence of a connection

3. Graph Visualization

- Node positions are determined using nx.spring_layout(G) for even spacing.
- Nodes are drawn with:
    - Color: lightgreen
    - Size: 2000
    - Font size: 15
- Edges are drawn with default styles using nx.draw().
- Edge weights are shown using nx.draw_networkx_edge_labels()
- A plot title is added via plt.title("Weighted Graph from Adjacency Matrix").

4. Execution

The main() function:

- Loads the matrix from weighted_adjacency_matrix.csv
- Prints the matrix to the console
- Calls draw_weighted_graph(adj_matrix) to generate the plot