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

Assignment 2

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:

Objective:
Load a binary dataset from a CSV file, interpret it as an adjacency matrix representing an unweighted graph, and visualize the corresponding graph.

Process:

1. **Data Loading:**
   The binary CSV file is read using Python's CSV module. Each row is converted into a list of integers (0s and 1s), and the entire dataset is stored as a NumPy array for convenient matrix operations.

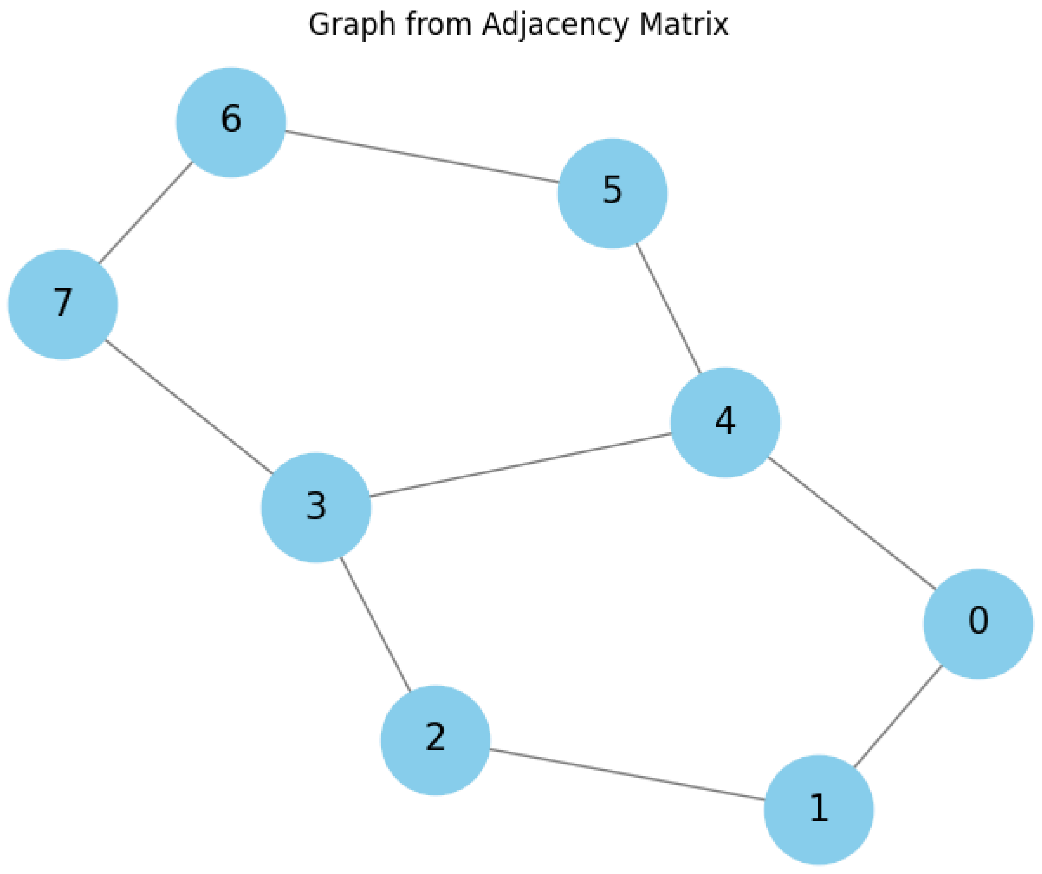
2. **Graph Construction:**
   Using the NetworkX library, the adjacency matrix is converted into an undirected graph (nx.Graph) if the matrix is symmetric, or a directed graph (nx.DiGraph) otherwise. Edges are added where the matrix entry is 1, indicating a connection.

3. **Graph Visualization:**
   Matplotlib and NetworkX's drawing utilities are used to visualize the graph. The spring_layout algorithm positions nodes to minimize edge crossings and evenly space them for clarity. Nodes are styled with colors and sizes to improve readability. Since the graph is unweighted, edges are drawn uniformly without labels.

Input:

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:



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

Numerical Dataset as Weighted Adjacency Matrix:

Objective:
Load a numerical dataset from a CSV file representing weighted edges between nodes and visualize the weighted graph accordingly.

Process:

1. **Data Loading:**
   The CSV file is loaded similarly, but entries are converted to floats to capture edge weights. The matrix is stored in a NumPy array.

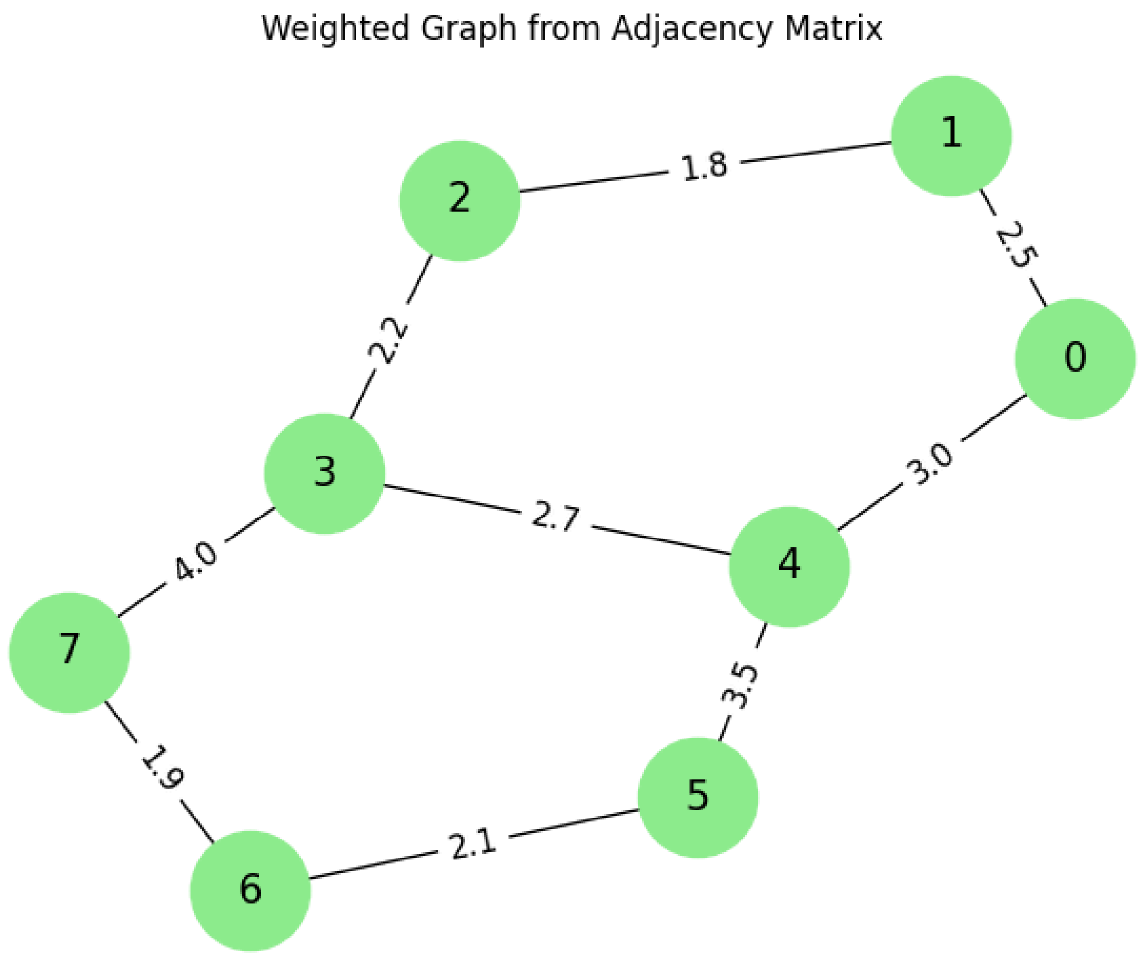
2. **Graph Construction:**
   Using NetworkX, the numerical adjacency matrix is converted to a graph. Edges with zero weights are removed to reflect the absence of a connection. The graph is treated as weighted, preserving edge weights as attributes.

3. **Graph Visualization:**
   The graph is plotted with node placement via the spring_layout algorithm for optimal visual clarity. Edges are drawn with thickness proportional to their weights to visually represent strength of connections. Edge weight values are displayed as labels along the edges. Nodes are styled distinctly to enhance presentation.

Input:

```
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:


Weighted Graph from Adjacency Matrix
0
1
2
3
4
5
6
7
1.8
2.5
2.2
3.0
2.7
4.0
3.5
1.9
2.1