PROJECT REPORT

For the Course of

DESIGN AND ANALYSIS OF ALGORITHMS (CSBB 202)

Submitted by

Name of the Students

DEVAGIRI NEERAJ BABU { 231210037}

OM HADKE {231210046}

M. CHAITANYA {231210063}

Semester - 3rd

In the Department of

Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, DELHI

Project Report

1.Problem Statement (Title) : Social Network Analysis

2. Problem Statement (Description) :

Developing a social network analysis tool inspired by platforms like Facebook and Instagram to implement the following core functionalities.

1. **Mutual Friends**- Identifying common connections between users, similar to Facebook's mutual friends feature.

2. **Friend Prediction**-Suggesting potential connections based on users' interests, similar to Instagram's "Suggestions for You," but enhanced by incorporating shared interests for more relevant recommendations.

3. **Community Detection-** Identifying and grouping users into communities by analyzing the network's structure, utilizing modularity optimization to detect densely connected clusters, similar to how social networks uncover interest-based groups.

4. **Centrality Measurement**- Evaluating user influence and importance within the network using centrality metrics, as seen in ranking algorithms used by various platforms.

Additionally, the system offers a web-based interface for users to interact with the graph data and view results.

3. **Tabular Representation :** The table below outlines the various functionalities of the system, providing a clear understanding of its structure. It categorizes the functionalities, explains each in detail, and specifies the techniques used for their implementation.

Functionalities	**Explanation**	**Techniques used for implementation**
Mutual friends	Returns the common neighbors shared by two specified nodes.	File handling , adjacency list *(defaultdict(list))*
Edge prediction	Predicts edges for a selected node based on its interests.	File handling, adjacency list, node_interests
Community detection	Assigns a node to the most suitable community based on modularity optimization.	Louvian method,networkx library for graph creation using edge list
Degree centrality	Identifies the most connected node in the network	Networkx library for graph creation using edge list
Closeness centrality	Identifies the node that is closest to all other nodes in the network.	
Betweenness centrality	Identifies the node that appears on the shortest paths between the most pairs of nodes in the network.	

4. Solution Approach :

4.1 Flowchart

(File Upload → Graph Creation → Analysis → Visualization → Results Display.)

4.2 Approach and results

4.2.1 Mutual Friends: Build adjacency list from the edge list file. Identify mutual friends by finding the intersection of the neighbors of the two nodes from the adjacency list.

Example of edge list file

`1 6` indicates an edge between node 1 and node 6.
`1 3` indicates an edge between node 1 and node 3.
`1 4` indicates an edge between node 1 and node 4.

4.2.2 Edge Prediction: We construct an adjacency list from an edge list file, similar to the process for finding mutual friends. However, this file includes an additional field representing the interest of the first node. The interest is encoded as a 4-bit binary number, where each bit corresponds to a specific category (e.g., music, movies, sports, etc.).An edge is predicted for a node if its neighbor's neighbor shares the same interest as the node.

Example of edge list file

`1 6 1000` indicates an edge between node 1 and node 6, where node 1 has Interest 1000

`1 3 1000` indicates an edge between node 1 and node 3, with the same interest.

`1 4 1000` indicates an edge between node 1 and node 4, also sharing the same interest.

4.2.3 Community Detection: Communities in the graph are identified by optimizing modularity, which evaluates the quality of the groupings. The process begins with building a graph from an edge list using the `build_graph` function. The `manual_community_detection` function iteratively assigns nodes to communities, enhancing modularity at each step by moving nodes to more suitable groups. The detection process halts when modularity improvement becomes negligible or after a defined number of iterations.

Example of edge list file

1 6 indicates an edge between node 1 and node 6.
1 3 indicates an edge between node 1 and node 3.
1 4 indicates an edge between node 1 and node 4.

4.2.4 centrality

Degree Centrality: Degree centrality of a node is calculated based on the number of direct connections (neighbors) it has. The degree centrality is normalized by the total number of nodes in the graph (minus one) to provide a relative score for each node. Nodes with more connections are considered more central.

Closeness Centrality: Closeness centrality is determined by calculating the sum of the shortest path lengths from the node to every other node in the graph. The inverse of this sum (normalized) gives the closeness centrality, with higher values indicating a node is closer to other nodes.

Betweenness Centrality: Betweenness centrality is calculated by counting how often each node appears in the shortest path between any two other nodes. The centrality is normalized by dividing by the total number of node pairs in the graph. Nodes that appear frequently on shortest paths are considered more central.

Example of edge list file

1 6 indicates an edge between node 1 and node 6.
1 3 indicates an edge between node 1 and node 4.
1 4 indicates an edge between node 1 and node 4.

4.3 Algorithm Used with Complexities

4.3.1 Mutual friends:

ALGORITHM

- Check if both nodes exist in the graph.
- Find the neighbors (connections) of both nodes.
- Identify common neighbors between the two nodes.
- Return the list of mutual friends (common neighbors).

COMPLEXITY

- d1: Number of elements in set1.
- d2: Number of elements in set2.
- Conversion to sets: O(d1+d2)
- Finding the intersection: O(min(d1,d2))
- Time Complexity: O(d1+d2+min(d1,d2))

4.3.2. Edge Prediction:

ALGORITHM

- Check if the node exists in the graph and has an interest.
- Find its neighbors and check if they have interests.
- Check if the neighbors' neighbors (second-degree neighbors) share the same interest and are not already connected.
- Predict edges based on shared interests between the node and second-degree neighbors.
- Return the list of predicted edges.

COMPLEXITY

Let N: Total number of nodes in the graph.

- *Let M: Total number of edges in the graph.*
- *Let d: Average degree of a node (d=2M/N)*
- *Complexity:* O(d^2).

4.3.3. Community Detection:

ALGORITHM

- Calculate the degree of each node and total edges in the graph.
- Assign each node to its own community initially.
- For each node, check its neighbors' communities and calculate modularity gain.
- Move the node to the community with the highest modularity gain.
- Repeat until modularity stops improving or maximum iterations are reached.
- Return the final community assignment for each node.

COMPLEXITY

Time Complexity: O(V.deg(v).Iterations).

- **V**: Total number of vertices in the graph.
- **deg(v))**: Degree of a vertex v, representing the average number of edges per vertex.
- **Iterations**: Number of iterations performed for each vertex.

4. Centrality Analysis:

ALGORITHM

Degree Centrality:

- For each node
- Count the number of neighbors it has.
- Divide this count by the total number of nodes minus one.

Closeness Centrality:

- For each node Find the shortest path to every other node.
- Calculate the total distance from the node to all other nodes.
- Compute the centrality as the inverse of this distance.

Betweenness Centrality:

- For each nodeCount how many shortest paths between other nodes pass through this node.
- Normalize the count by dividing by the number of node pairs.

COMPLEXITY

- Degree: O(V + E)
- Closeness: $O(V \cdot (E+V))$
- Betweenness: $O(V^2 \cdot (V+E))$

Function	Complexity
Mutual friends	O(d1+d2+min(d1,d2))
Edge Prediction	$O(d^2)$
Community Detection	O(V.deg(v).Iterations)
Centrality Analysis	Degree: O(V + E) Closeness: O(V · (E+V)) Betweenness: O(V^2 · (V+E))

5. Platform Used (Hardware + Software) :

- Python 3.9+
- Libraries: `NetworkX`, `Matplotlib`, `Flask`.
- Operating System: Windows

6. References :

1. NetworkX Documentation: https://networkx.org/documentation/

2. Matplotlib Library Guide: https://matplotlib.org/stable/contents.html

3. Flask Framework Documentation: https://flask.palletsprojects.com/

4. Louvian method https://en.wikipedia.org/wiki/Louvain_method

7. Conclusion :

This project successfully implements a system for creating, analyzing, and visualizing network graphs. The integration of algorithms for graph property analysis, community detection, and edge prediction, along with a user-friendly web interface, makes this system versatile for real-world applications.

Key Achievements:

Mutual Friends: Quickly finds shared connections to enhance user interaction.

Edge Prediction: Incorporating interests improves the accuracy of predicting new connections.

Community Detection and Centrality: Helps identify key groups and influential nodes in the network.

Efficient Analysis and Visualization: visualize graph for a better understanding

8. Results

8.1 Graph Analysis and visualization

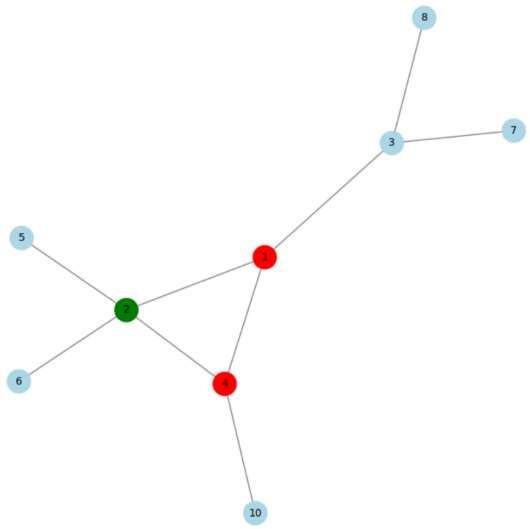
8.1.1 Mutual Friends

Upload and Find Mutual Friends

Mutual Friends between 1 and 4 are

2

Visualization



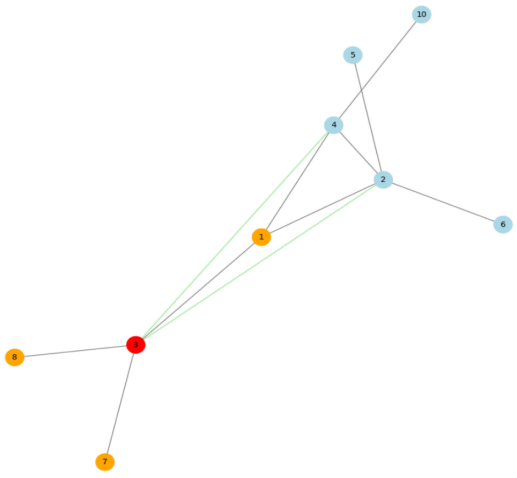
8.1.2 Edge prediction

Predict Edges

Predicted Edges

3 - 2
3 - 4

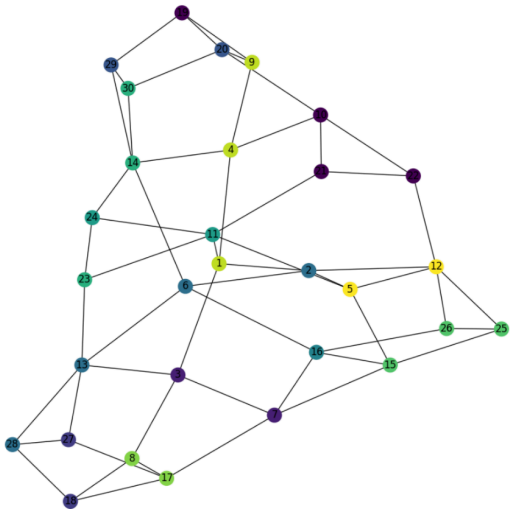
Visualization



8.1.3 Community detection

Node 25 is in Community E
Node 26 is in Community E
Node 27 is in Community I
Node 28 is in Community H
Node 29 is in Community L
Node 30 is in Community G

Community Visualization



8.1.4 Centrality

Calculate Centrality

Top Node by Degree Centrality

Node 11= 0.1724137931034483

Top Node by Closeness Centrality

Node 6= 0.4393939393939394

Top Node by Betweenness Centrality

Node 1= 0.20566502463054187

Centrality Visualization