**CS 342 : SOCIAL NETWORK ANALYSIS**

**MINI PROJECT**

**Topic : Citation Graph of professors**

**INSTITUES: IIT INDORE AND NIT BHOPAL**

**Submitted By:**

Name: **Hardik Gondwal and Hanusha Jain**

Roll No: **U22CS061, U22CS066**

Branch: **CSE**

Semester: **6th Sem** **A**

**Department of Computer Science and Engineering**

****

**SV NATIONAL INSTITUTE OF TECHNOLOGY SURAT**

Data Collection

**1. Extracting Professor Details**

The Google Scholar API was used to retrieve information about each professor, including:

* Scholar ID
* Name
* Affiliation
* Email domain (if available)
* Number of citations
* H-index
* i10-index
* Research interests
* Google Scholar profile URL

CODE:

import networkx as nx

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

# Load Professors from CSV

professors\_df = pd.read\_csv("google\_scholar\_profiles.csv")

scholar\_ids = professors\_df["Scholar ID"].tolist()

professor\_names = professors\_df["Name"].tolist()

graph\_metrics\_list = []

for i, scholar\_id in enumerate(scholar\_ids):

    try:

        # Get author details

        author = scholarly.search\_author\_id(scholar\_id)

        author = scholarly.fill(author)

        professor\_name = author.get("name", f"Professor {i+1}")

        # Create a Directed Graph for the professor

        G = nx.DiGraph()

        # Add professor as a node

        G.add\_node(professor\_name)

        # Add co-authors as nodes and edges

        for coauthor in author.get("coauthors", []):

            coauthor\_name = coauthor["name"]

            G.add\_node(coauthor\_name)

            G.add\_edge(professor\_name, coauthor\_name)  # Directed edge

        # Compute Graph Measures

        num\_nodes = G.number\_of\_nodes()

        num\_edges = G.number\_of\_edges()

        density = nx.density(G)

        degree\_centrality = nx.degree\_centrality(G)

        betweenness\_centrality = nx.betweenness\_centrality(G)

        closeness\_centrality = nx.closeness\_centrality(G) if num\_nodes > 1 else {}

        avg\_degree = sum(dict(G.degree()).values()) / num\_nodes if num\_nodes > 0 else 0

        clustering\_coefficient = nx.average\_clustering(G) if num\_nodes > 1 else 0

        # Save metrics

        metrics = {

            "Professor": professor\_name,

            "Scholar ID": scholar\_id,

            "Nodes": num\_nodes,

            "Edges": num\_edges,

            "Density": density,

            "Avg Degree": avg\_degree,

            "Clustering Coefficient": clustering\_coefficient,

            "Degree Centrality": degree\_centrality.get(professor\_name, 0),

            "Betweenness Centrality": betweenness\_centrality.get(professor\_name, 0),

            "Closeness Centrality": closeness\_centrality.get(professor\_name, 0),

        }

        graph\_metrics\_list.append(metrics)

        # Print metrics

        print(metrics)

        # Visualize Graph

        plt.figure(figsize=(8, 6))

        nx.draw(G, with\_labels=True, node\_size=2000, node\_color="lightblue", edge\_color="gray", font\_size=10)

        plt.title(f"Citation Graph - {professor\_name}")

        plt.savefig(f"{professor\_name}\_citation\_graph.png")

        plt.show()

    except Exception as e:

        print(f"Error processing {professor\_names[i]}: {e}")

# Save Graph Metrics to CSV

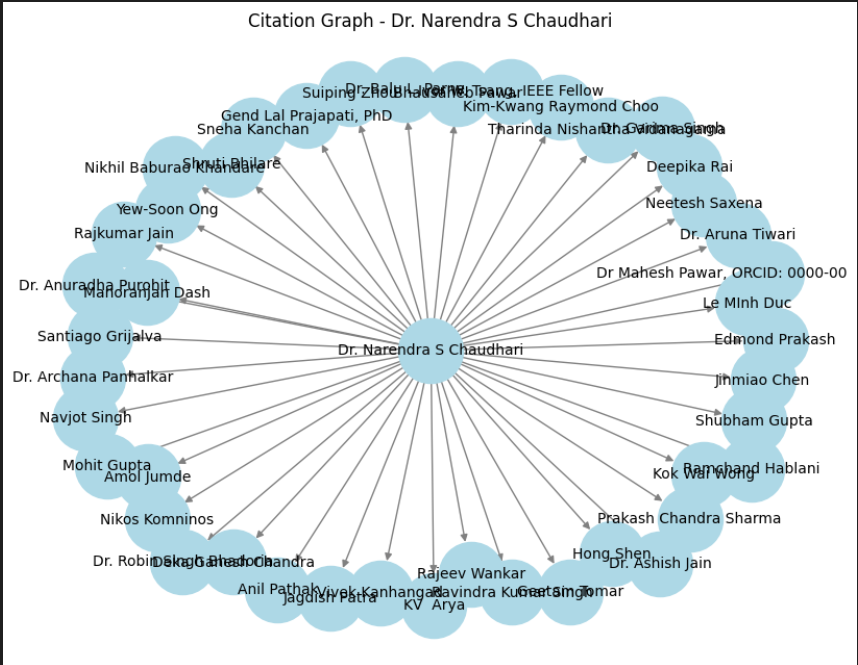
metrics\_df = pd.DataFrame(graph\_metrics\_list)

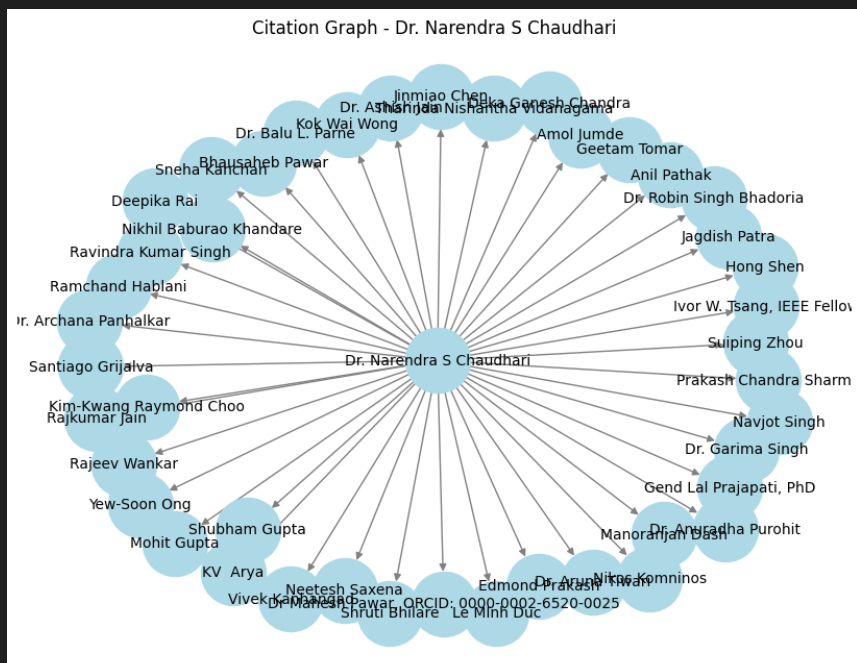
metrics\_df.to\_csv("scholar\_graph\_metrics.csv", index=False)

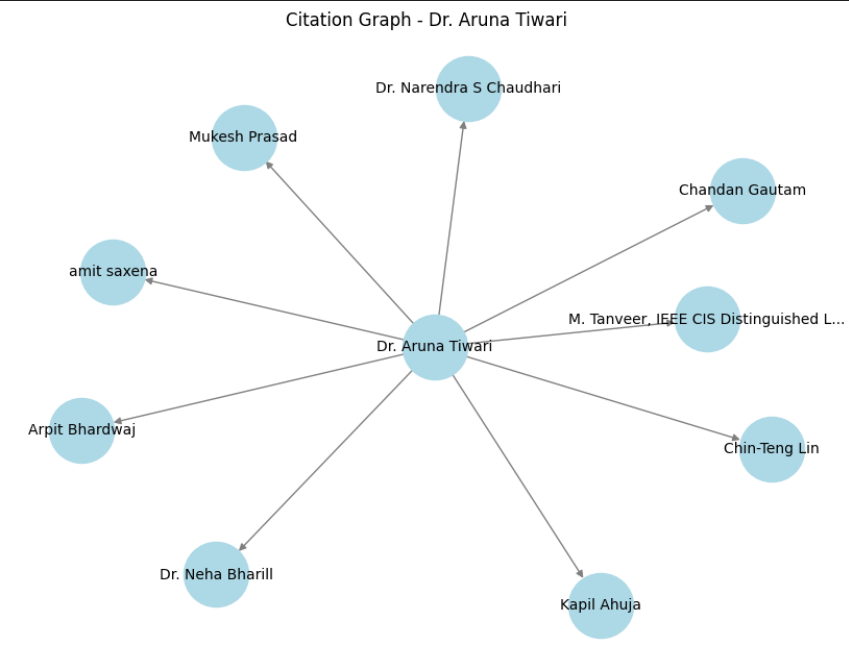
print("Graph analysis completed. Metrics saved to 'scholar\_graph\_metrics.csv'.")

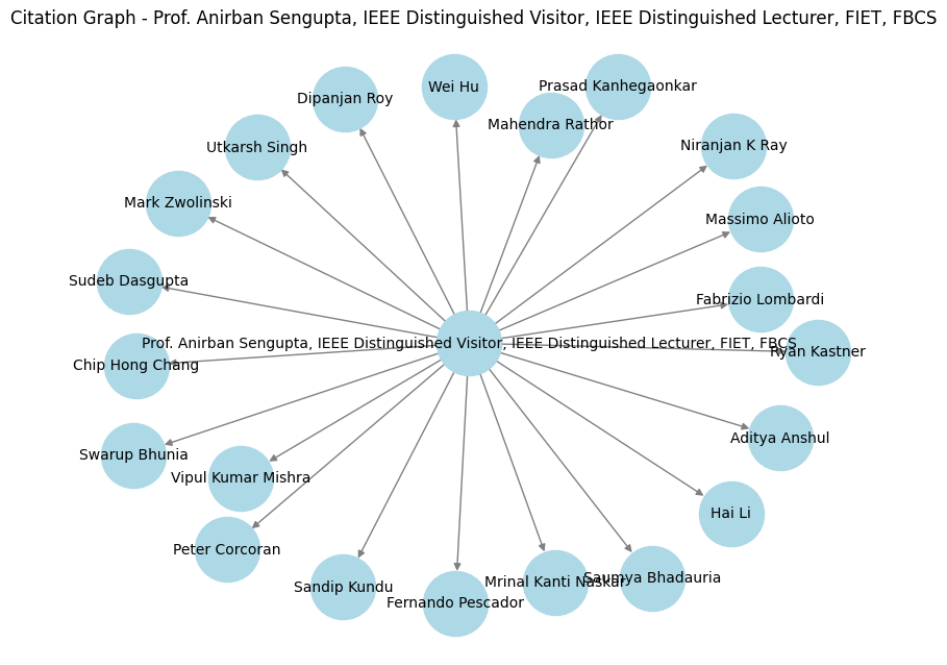
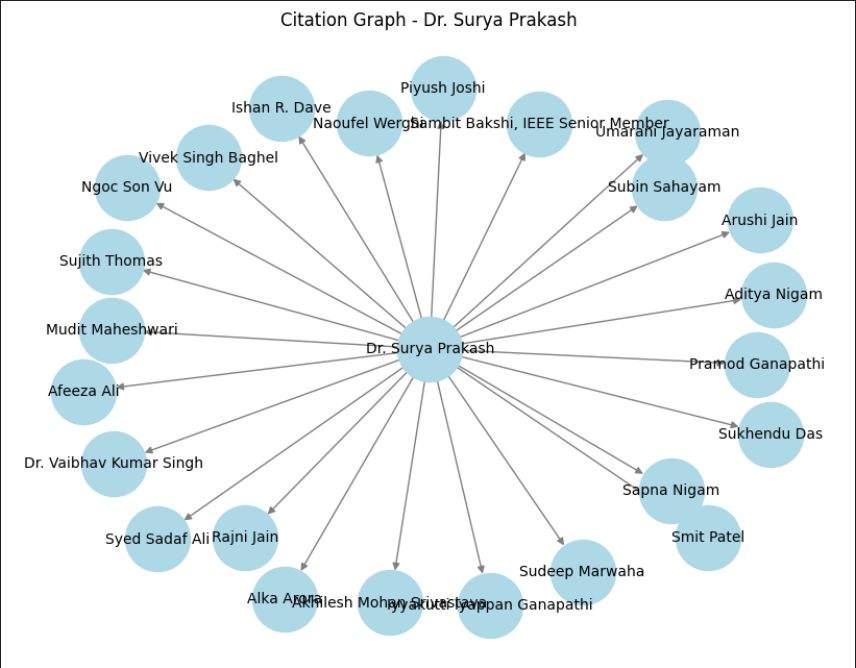
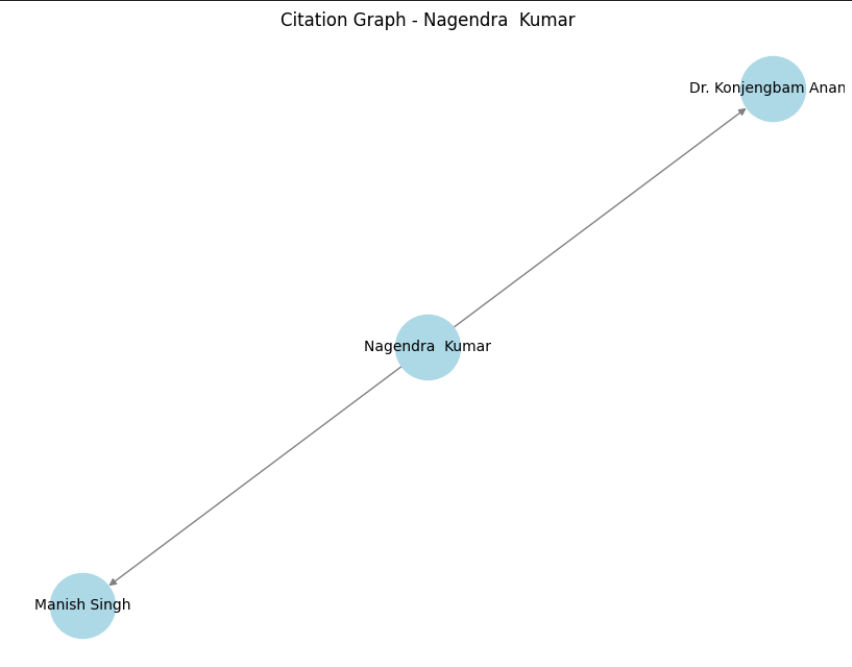
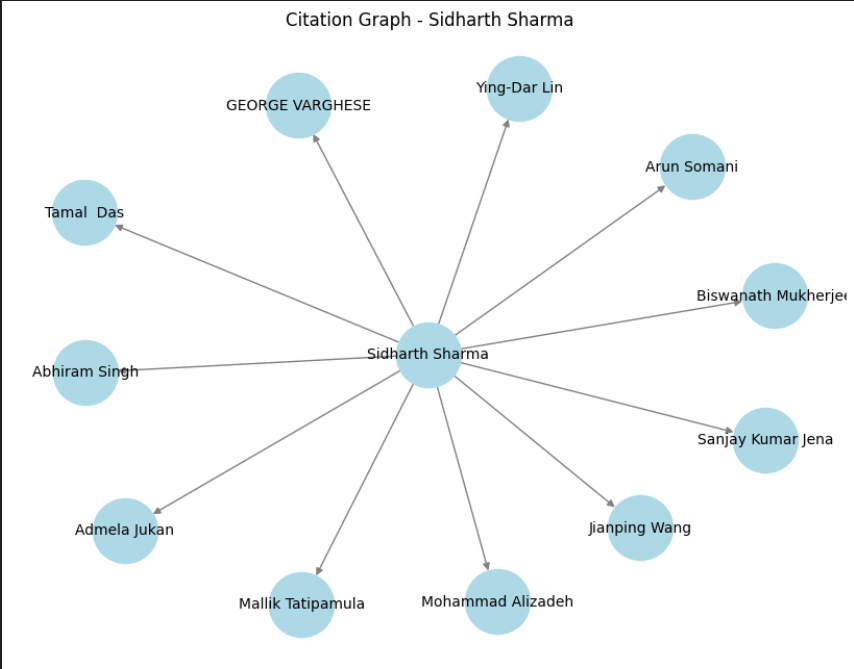
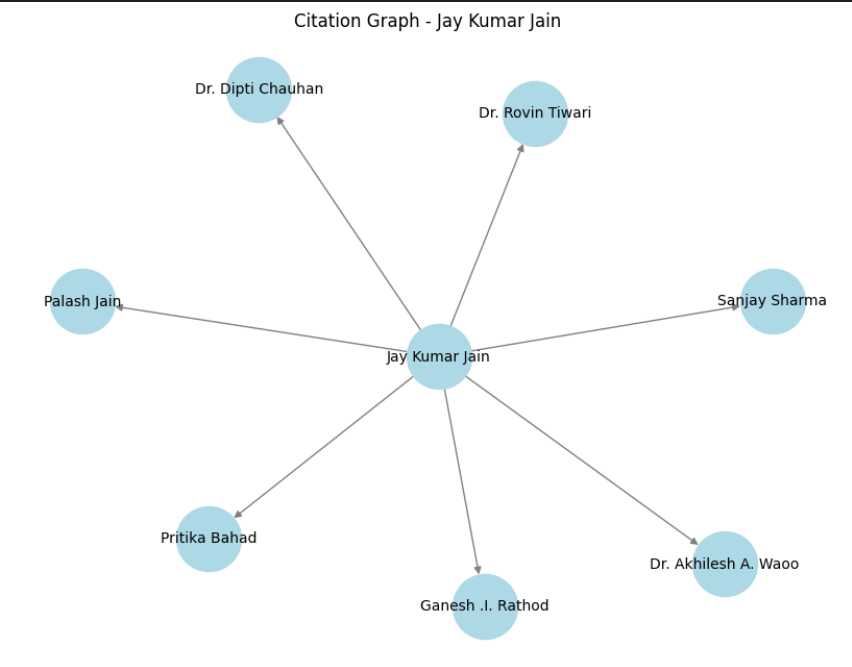
OUTPUT:

**Graph Construction and Analysis**







# **Best Professor Selection**

import pandas as pd

# Load Metrics

graph\_metrics = pd.read\_csv("scholar\_graph\_metrics.csv")

# Fill missing values with 0

graph\_metrics.fillna(0, inplace=True)

# Normalize Graph Parameters Safely

for column in ["Density", "Avg Degree", "Clustering Coefficient", "Degree Centrality", "Betweenness Centrality", "Closeness Centrality"]:

    if column in graph\_metrics.columns:

        col\_min, col\_max = graph\_metrics[column].min(), graph\_metrics[column].max()

        if col\_max > col\_min:  # Only normalize if there's a range

            graph\_metrics[column] = (graph\_metrics[column] - col\_min) / (col\_max - col\_min)

# Compute Overall Score (Weighted Sum)

graph\_metrics["Overall Score"] = (

    graph\_metrics["Density"] \* 0.1 +

    graph\_metrics["Avg Degree"] \* 0.2 +

    graph\_metrics["Clustering Coefficient"] \* 0.1 +

    graph\_metrics["Degree Centrality"] \* 0.2 +

    graph\_metrics["Betweenness Centrality"] \* 0.2 +

    graph\_metrics["Closeness Centrality"] \* 0.2

)

# Ensure no NaN values in "Overall Score"

graph\_metrics["Overall Score"].fillna(0, inplace=True)

if not graph\_metrics.empty and graph\_metrics["Overall Score"].max() > 0:

    best\_professor\_row = graph\_metrics.loc[graph\_metrics["Overall Score"].idxmax()]

    best\_professor = best\_professor\_row["Professor"]

    print("🏆 The Best Professor based on citation and graph parameters is:")

    print(f"➡️ {best\_professor}")

else:

    best\_professor = "No valid data available"

    print("⚠️ No valid data available to determine the best professor.")

# Save Rankings

graph\_metrics.sort\_values(by="Overall Score", ascending=False).to\_csv("professor\_rankings.csv", index=False)

print("📊 Professor rankings saved to 'professor\_rankings.csv'.")

OUTPUT:



RELATIONSHIP GRAPH:

from scholarly import scholarly

import requests

from bs4 import BeautifulSoup

import pandas as pd

import time

professor\_urls = [

    "https://scholar.google.com/citations?user=HMAzliEAAAAJ",  # Dr. Sanyam Shukla (MANIT Bhopal)

    "https://scholar.google.com/citations?user=0YYnNYkAAAAJ",  # Jay Kumar Jain (MANIT Bhopal)

    "https://scholar.google.com/citations?user=fWgOgcIAAAAJ",  # Dr. Mitul Kumar Ahirwal (MANIT Bhopal)

    "https://scholar.google.com/citations?user=97HSz70AAAAJ",  # Vaibhav Soni (MANIT Bhopal)

    "https://scholar.google.com/citations?user=cPorLCwAAAAJ"   # Vijay Bhaskar Semwal (MANIT Bhopal)

]

# Dictionary to store professor data

professor\_data = {}

# Function to extract data from scholarly

def get\_professor\_basic\_data(scholar\_url):

    scholar\_id = scholar\_url.split("user=")[-1].split("&")[0]

    author = scholarly.search\_author\_id(scholar\_id)

    author = scholarly.fill(author, sections=["basics", "indices"])  # Get basic details

    name = author.get("name", "Unknown")

    interests = ", ".join(author.get("interests", []))  # Convert list to string

    citedby = author.get("citedby", 0)

    return name, interests, citedby

# Function to scrape additional data (co-authors, h-index, i10-index)

def scrape\_google\_scholar\_profile(profile\_url):

    try:

        headers = {

            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"

        }

        response = requests.get(profile\_url, headers=headers)

        if response.status\_code != 200:

            print(f"Failed to retrieve {profile\_url}")

            return set(), 0, 0

        soup = BeautifulSoup(response.text, "lxml")

        # Get h-index and i10-index

        indices = soup.find\_all("td", class\_="gsc\_rsb\_std")

        h\_index = int(indices[2].text) if len(indices) > 2 else 0

        i10\_index = int(indices[4].text) if len(indices) > 4 else 0

        # Get co-authors from publications

        coauthors\_set = set()

        papers = soup.find\_all("tr", class\_="gsc\_a\_tr")

        for paper in papers:

            paper\_authors = paper.find("div", class\_="gs\_gray")

            if paper\_authors:

                authors\_list = paper\_authors.text.split(", ")

                coauthors\_set.update(authors\_list)

        return coauthors\_set, h\_index, i10\_index

    except Exception as e:

        print(f"Error scraping {profile\_url}: {e}")

        return set(), 0, 0

# Fetch data for all professors

for url in professor\_urls:

    print(f"Processing: {url}")

    name, interests, citedby = get\_professor\_basic\_data(url)

    coauthors, h\_index, i10\_index = scrape\_google\_scholar\_profile(url)

    professor\_data[name] = {

        "Interests": interests,

        "Citations": citedby,

        "H-Index": h\_index,

        "i10-Index": i10\_index,

        "Coauthors": list(coauthors)

    }

    time.sleep(3)  # To avoid rate limiting

# Convert to DataFrame

df = pd.DataFrame.from\_dict(professor\_data, orient="index")

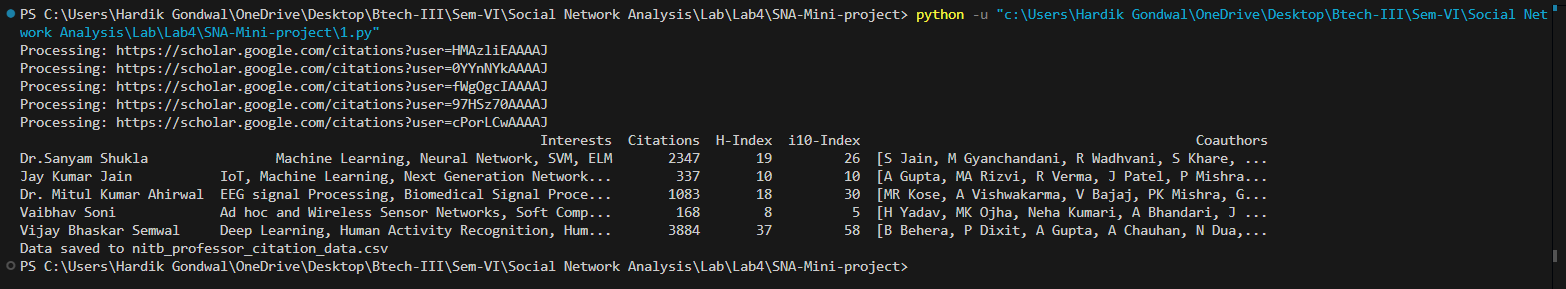
print(df)

# Save to CSV

df.to\_csv("nitb\_professor\_citation\_data.csv", index=True)

print("Data saved to nitb\_professor\_citation\_data.csv")

OUTPUT:



2.py

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Load IIT and NIT professor data

iit\_df = pd.read\_csv("iitb\_professor\_citation\_data.csv")

nit\_df = pd.read\_csv("nitb\_professor\_citation\_data.csv")

# Create graph

G = nx.Graph()

# Add IIT professors (Red)

iit\_profs = set(iit\_df.iloc[:, 0])

for prof in iit\_profs:

    G.add\_node(prof, color="red")

# Add NIT professors (Blue)

nit\_profs = set(nit\_df.iloc[:, 0])

for prof in nit\_profs:

    G.add\_node(prof, color="blue")

# Extract and add co-authors

iit\_coauthors = set()

for coauthors in iit\_df.iloc[:, -1]:  # Last column contains co-authors

    if isinstance(coauthors, str):

        coauthors = eval(coauthors) if coauthors.startswith("[") else []

        iit\_coauthors.update(coauthors)

nit\_coauthors = set()

for coauthors in nit\_df.iloc[:, -1]:

    if isinstance(coauthors, str):

        coauthors = eval(coauthors) if coauthors.startswith("[") else []

        nit\_coauthors.update(coauthors)

# Common co-authors (Green)

common\_coauthors = iit\_coauthors.intersection(nit\_coauthors)

for author in common\_coauthors:

    G.add\_node(author, color="green")

# Other co-authors (Gray)

other\_coauthors = (iit\_coauthors.union(nit\_coauthors)) - common\_coauthors - iit\_profs - nit\_profs

for author in other\_coauthors:

    G.add\_node(author, color="gray")

# Add edges between professors and co-authors

for \_, row in iit\_df.iterrows():

    prof = row.iloc[0]

    if isinstance(row.iloc[-1], str):

        coauthors = eval(row.iloc[-1]) if row.iloc[-1].startswith("[") else []

        for coauthor in coauthors:

            G.add\_edge(prof, coauthor)

for \_, row in nit\_df.iterrows():

    prof = row.iloc[0]

    if isinstance(row.iloc[-1], str):

        coauthors = eval(row.iloc[-1]) if row.iloc[-1].startswith("[") else []

        for coauthor in coauthors:

            G.add\_edge(prof, coauthor)

# Draw the graph

plt.figure(figsize=(12, 8))

node\_colors = [G.nodes[node]["color"] for node in G.nodes]

pos = nx.spring\_layout(G, seed=42)  # Positioning for better visualization

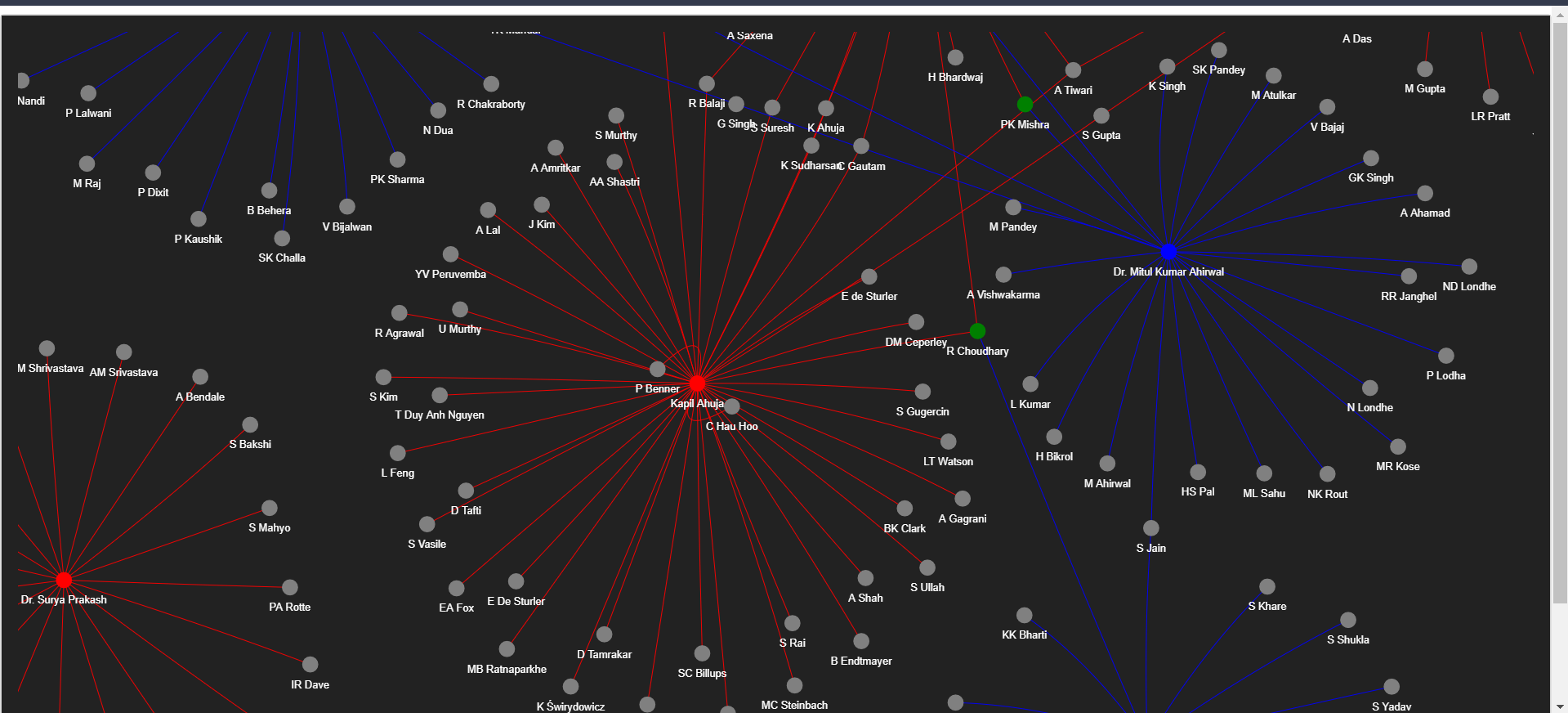
nx.draw(G, pos, with\_labels=True, node\_size=50, font\_size=6, node\_color=node\_colors, edge\_color="gray", alpha=0.6)

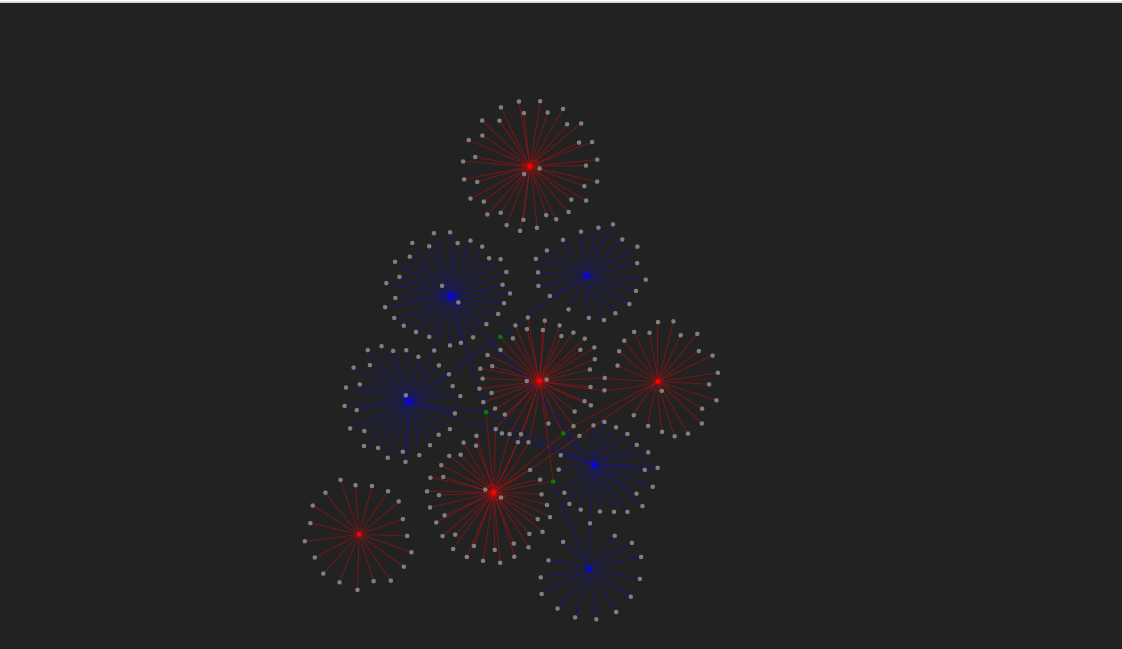
# Show the plot

plt.title("Co-authorship Network between IIT and NIT Professors")

plt.show()

Pyvis is further used here for the data visualisation to get the information about the relationship and the common nodes in a clean and conside way:-





***Summary***

**Introduction:-**

This project aims to analyze co-author relationships among professors from NITs and IITs by constructing a collaboration network based on their joint research publications. The network will be evaluated using various graph-based centrality measures, such as **degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality**, to quantify the influence and involvement of researchers within the academic community. The insights derived from this analysis can help assess interdisciplinary collaborations, identify key contributors, and uncover patterns in research partnerships.

**Use Case:-**

* **Identifying Key Researchers:** Helps in recognizing highly influential researchers based on their centrality in the collaboration network.
* **Collaboration Strength Analysis:** Determines the strength of academic partnerships between professors based on the frequency of co-authored publications.
* **Departmental and Institutional Collaboration:** Evaluates the extent of research collaboration across different institutions and departments.
* **Research Trend Analysis:** Provides insights into how research collaborations evolve over time and which domains attract the most co-authorship.
* **Funding and Policy Recommendations**: Assists policymakers and funding agencies in allocating resources by identifying strong research networks and potential interdisciplinary opportunities.

**Data Collection**

We utilized **Scholarly Number**, an in-built resource for extracting publication details such as **author names, affiliations, and co-authorship information**. The extracted data is then structured into a **CSV file**, serving as the foundation for further analysis.

**Graph Construction**

* A network graph was created using NetworkX.
* Nodes represent professors and co-authors, with different colors assigned:
  + **Red**: IIT Professors
  + **Blue**: NIT Professors
  + **Green**: Common co-authors between IIT and NIT
  + **Gray**: Other co-authors
* Edges represent co-authorship relationships.

**Network Visualization**

* The graph was drawn using Matplotlib.
* The spring\_layout algorithm was used for positioning.
* Labels and colors were applied to enhance clarity.

**Summary**

This project successfully visualized and analyzed the co-authorship network between IIT and NIT professors. It highlights collaboration trends, identifies common researchers, and provides insights into academic networking. Future work can incorporate additional metrics like citation impact and research field categorization to enhance the analysis.