YANGI O‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI
Volume 1 Issue 3 04.06.2024
https://phoenixpublication.uz/ Online ISSN: 3030-3494

NETWORK FORMATION IN SOCIAL MEDIA PLATFORMS

Azimov Mirvohid
student of inha university

Title	Page No

- **Problem Introduction**
- **Operations included in the application**
- **Choice of Data Structures**
- **Code**
- **limitations and Future Scope**
- **Subject Importance**
- **References**

Problem Introduction

Social media has provided us with incredible opportunities to move our social interactions into the digital world. Common fact is that people are very social species and they can not stand without their network of relations. Every person living an adequate life has a network consisting of at least two friends. And, as we know well, in the last couple of decades the digital world and especially social media has become one the most rapidly developing and becoming more accessible spheres in the world. Developers, like myself, while developing great softwares to bring people together, have encountered interesting problems. Puzzling thing is, when people used to make friends in a traditional way(face to face), it was easy to keep track of common relations between each other. Even if people had a lot of relationships with people from different parts of their life, like family, work, childhood friends, and etc., they used some notes or schemes to fix their networking with others. But with the massive development of social media, this process became nearly impossible to do manually,

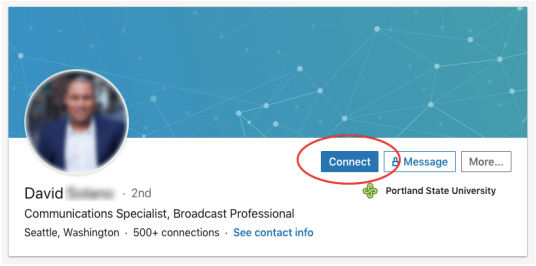
because of several factors, such as the big amount of online and not only relations, growth of connectivity between different spheres etc. This task has leaned on developers from all around the world to solve the problem of navigating and comprehending the intricate web of connections. The thing is, to solve this problem the algorithm should be created to do all this task for people. In this report form I will provide an algorithmic solution for this dilemma. I named the algorithm “Social Media Network”.

Operations Included In the Application

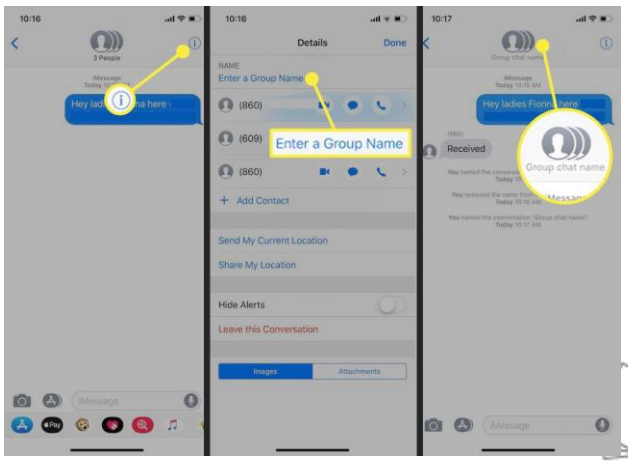
So the Social Media Networks main goal is to provide needed analyses for simple users to automate the process of finding common connections between friends. If you have noticed there is no such condition as “existing friends” in the previous statement, but it is very crucial for the algorithm to function properly, so the algorithm will include the ability to include friends and create a network.

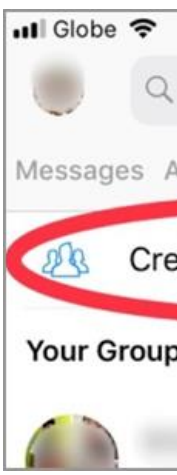
Including friends means following their social media or creating personal chat with them etc., and creating a network means creating some group chats, creating some grouped contacts etc(pictures below).The Social Media Network algorithm is mostly designed not to interact with users directly, but work invisibly, i.e. integrated with software applications of choice. So it is software developers who will choose which actions will trigger the operations that will be provided below.

Instagram’s follow and message



Linkedin's Connect and message





Imessage’s create group contact and chat Telegram’s create group

So here are that three operations to be integrated by developer into their software:

1) **Adding Friends:**

   This functionality should allow users to be able to seamlessly add their new friends with just a click of button to interests by providing the algorithm with essential data to use for analysis, like name, age, interests etc.

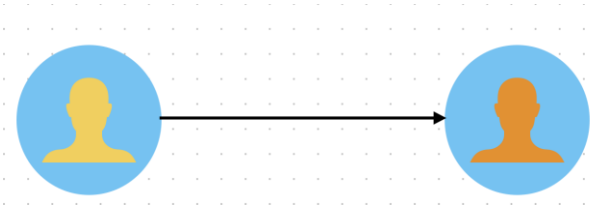
   Create Connection:

   The main feature of this functionality is to create a so-called spider net, where the spider web will connect the added friends, allowing algorithms to analyze not only one one by one, but also the whole groups of people.

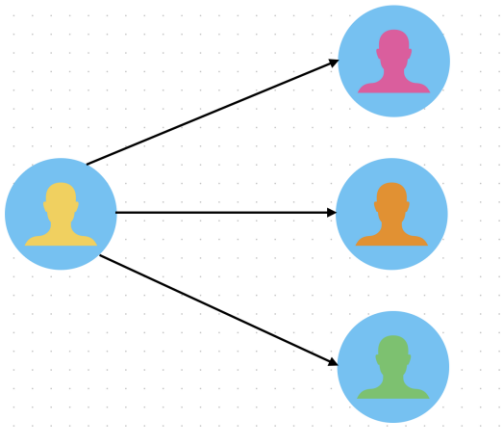
   Analyzing the Network:

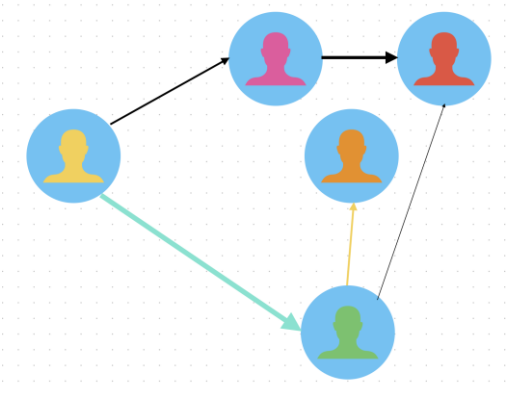
   Here is the culmination of my algorithm. Using the specific operations of the data structure I have chosen(will be discussed in the next paragraph) the algorithm will accomplish its duty.
   Algorithm will analyze social networks and offer some insights about common friends, degree of closeness and essential friends, so that developers can construct better opportunities for users to enlarge their community and create a heart warming atmosphere.

Below are the illustrations of the operations to better understand them:



1. Adding Friends





2. Creating connections

3. Analyzing the network(look to the difference between arrows, which mean the degree of closeness and essential friends)

Choice of data structure

In the data structure course we have studied many very interesting and useful data structures. But only one of them was remarkable enough to construct an idea and develop an algorithm on it. So as we continue, the fundamental choice for the Social Media Network algorithm is graph data structure. Specifically speaking, I have chosen weighted graphs, adjacency list(queue) and breadth-first traversals method respectively, for storing and retrieving essential data, because these methods are logically most suitable in my case. The chosen structure and methods will allow developers to efficiently represent connections between users and support dynamic changes within the social network. And one small detail which makes simple software more complex, location of each node in the graph representing the network can play a crucial role, because it represents the closeness of the people to each other and this can be used in many ways.

Shortly about representation in the graph: graph itself represents the network between people, so logically the nodes are people themselves, the edges of graph keeps data about connection between people, if the edges are shorter it means the relationship is stronger and vice versa. Justification: The choice of a graph, implemented using an adjacency list, is justified by its natural alignment with the structure of social networks. It balances the need for efficient storage with the dynamic and complex nature of social relationships. The adjacency list, in particular, ensures that the application can handle changes in the network structure efficiently.

Code

So let’s move to the one of the most important parts of any algorithm:

```
graph = {}
function addFriend(person_id, name, age, interests):
        graph[person_id] = {
        “name” : name
        “age” : age
```

In this pseudocode I demonstrated two simple parts of code, adding friends and creating or enlarging connections. The first function “addFriends as you can see takes some arguments: person_id which will indicate the person in the graph, their name, age and interests. And in the body of function except setting all variables, there is the friends set() part. This part ensures adding this person as a friend. And the second function “createConnection” implements the creation of connections and groups between people and different existing groups.

Now we move towards our last function analyzeNetwork:

```
function analyzeNetwork(person_id):
        visited.set()
        queue = []
        queue.enqueue(person_id)
        visited.add(person_id)
        while queue.front != -1 or queue.front > queue.rear:
                currentPerson = queue.dequeue()
                print(len(graph[currentPerson][friends]))
                print(calculateAvarageFriends(currentPerson))
                identifyCommonFriends(curretPerson)
                for friend in graph[currentPerson]['friends']:
                        if friend not in visited:
                        queue.enqueue(friend)
                        visited.add(friend)
```

So let’s quickly analyze the pseudocode.| visited.set() | : this set keeps track of the nodes (individuals) that have been visited during the graph traversal to avoid redundant processing.| queue = [] | : this queue will store the nodes to be processed in a breadth-first manner.| queue.enqueue(person_id) | : the initial node (person) specified by personId is enqueued to start the breadth-first traversal.| visited.add(person_id) | : the starting node is marked as visited. Then comes the iterative approach. The while loop continues as long as

the queue is not empty. In each iteration one person is dequeued from the queue as | currentPerson = queue.dequeue() | and processed. The next line of code prints out the number of friends of currentPerson. Then the next print statement prints the result of the function ‘calculateAvarageFriends’(code of this function is provided below). This function calculates the average number of friends among the friends of the current person. And next ‘identifyCommonFriends’ function is invoked(its code is also below) , this function identifies common friends between the current person and every other person in the network.| for friend in graph[currentPerson]['friends'] | : iterates through the friends of the current person.| if friend not in visited | : checks if the friend has not been visited to avoid redundant processing. | queue.enqueue(friend) | : enqueues unvisited friends for further exploration.| visited.add(friend) | : marks the friend as visited.

Here is the code of the additional functions to perform original functions:

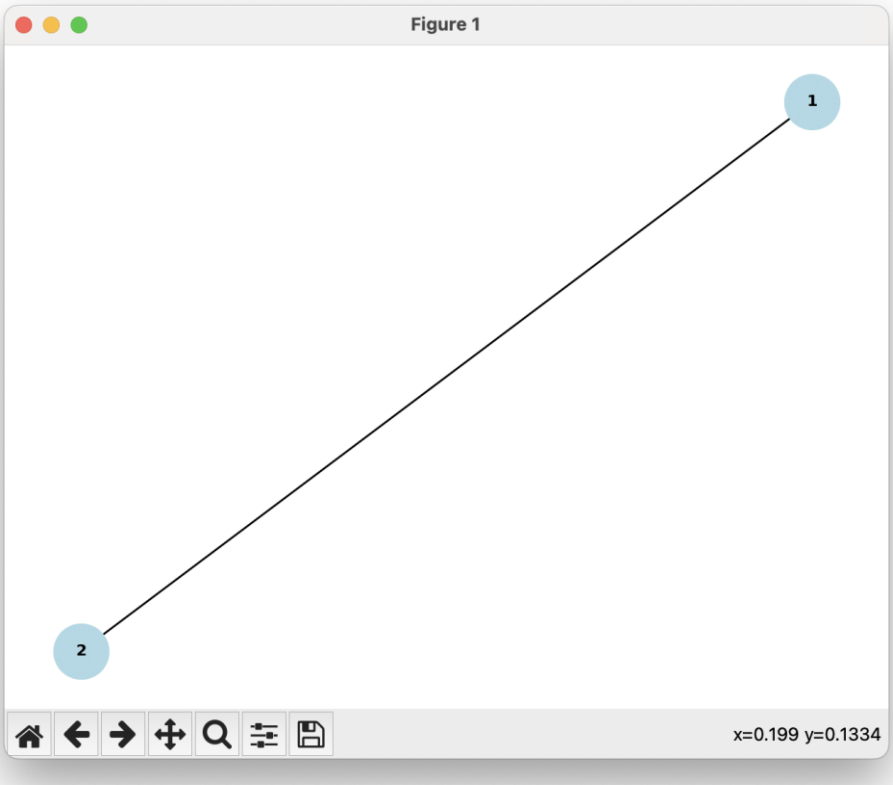
```
function calculateAverageFriends(person_id):
        totalFriends = 0
        for friend in graph[personId]['friends']:
                totalFriends += len(graph[friend]['friends'])
                return totalFriends / len(graph[person_id]['friends'])
```

```
function identifyCommonConnections(person_id):
        for otherPerson in graph.keys():
        if otherPerson != person_id:
                commonFriends = set(graph[person_id]['friends'])&set(graph[otherPerson]['friends']) if
                commonFriends:
                                print("Common friends between", person_id, "and", otherPerson, ":",
                                commonFriends)
```

I have implemented addFriend function in python language to see if the algorithm, and to see the graphical output of the program, here are the results:

```python
import networkx as nx
import matplotlib.pyplot as plt
social_network = nx.Graph()
social_network.add_node(1, name="Alice", age=25, interests=["Reading", "Travel"])
social_network.add_node(2, name="Bob", age=28, interests=["Photography", "Music"])
social_network.add_edge(1, 2)
pos = nx.spring_layout(social_network)
nx.draw(social_network, pos, with_labels=True, node_color="lightblue", font_size=8, font_color="black", font_weight="bold", node_size=800)
plt.title("Social Network Connectivity")
plt.show()
```

It is python code that uses networkx library to implement the addFriend function



and here is the graphical output of our newly created connection. So such implementations can be added to social media apps to automate the whole process.

Than I implemented analyzeNetwork function again using python:

```
Analyzing network starting from node 1:
Common friends between 1 and 2: set()
Common friends between 1 and 3: set()

Degree Centrality:
Node 1: 0.5
Node 2: 0.5
Node 3: 0.5
Node 4: 0.75
Node 5: 0.25
```

```python
import networkx as nx
def analyze_network(graph, start_node):
    print(f"Analyzing network starting from node {start_node}:")
    find_common_friends(graph, start_node)
    calculate_degree_centrality(graph)

def find_common_friends(graph, node):
    for neighbor in graph.neighbors(node):
        common_friends = set(graph.neighbors(node)) & set(graph.neighbors(neighbor))
        print(f"Common friends between {node} and {neighbor}: {common_friends}")

def calculate_degree_centrality(graph):
    degree_centrality = nx.degree_centrality(graph)
    print("\nDegree Centrality:")
    for node, centrality in degree_centrality.items():
        print(f"Node {node}: {centrality}")
G = nx.Graph()
G.add_edges_from([(1, 2), (1, 3), (2, 4), (3, 4), (4, 5)])
analyze_network(G, start_node=1)
```

And if we briefly talk about possible use cases: the most intended one is finding common friends and presenting them to the user. Also developers can use this algorithm to determine the shortest path or the number of connections between the specified person and others. Also this algorithm can be used to evaluate the influence or centrality of nodes based on their connections within the network.

Limitations And Future Scope

Limitations:

So as we have decided this algorithm will graph to get and return the main points of the process. But here is one of the problems: scalability. As the size of people's social network is growing, and the amount of people using and interacting rocketing currently, the efficiency of graph traversal and analysis algorithms may decrease. This could result in longer processing times and potentially affect the responsiveness of the application. The solution of this problem is very possible, such as parallel processing or more advanced graph algorithms, but still this limitation should be considered for the long term.

Also algorithm may suffer from the diverse and dynamic nature of social interactions. Such factors, such as cultural differences, varying user behavior, and day-to-day changing trends on social media may need complex amounts of variables to be analyzed by the algorithm, may create real trouble and big complexity to the algorithm, so that it can’t andress them all.

Continuing talking about dynamic behavior of the social interactions and social media, real-time updates may challenge our algorithm. The algorithm may not handle dynamic changes in the social network efficiently, especially if there are frequent updates or additions. This problem is also solvable in perspective by adding more real-time data structures or event driven mechanisms to the initial algorithm, but is worth taking into consideration.

Future scope:

Implementing this algorithm in the social media app can boost its performance and user-friendliness. Providing more advanced analytics to the developers, algorithms can change the good way interaction between users and developers. These features open up new stages to grow for social media companies, and secures great perspectives. Enriching the data set of the algorithm by collaborating with other social media apps, may create an incredible atmosphere for users and great profit for business owners.

Creating machine learning models based on this algorithm will bring much more detailed and essential analytics for developers. Based on machine learning predictions, this algorithm may offer users the best recommendations and can form a more personalized network.

Also implementing a feedback mechanism is a must in the future. For the algorithm to perform better, feedback should be passed as arguments for analysis. This can help to avoid the second problem discussed in the limitations part.

In conclusion, the future scope of the Social Media Network is very bright, offering opportunities for growth, innovation, and enhanced user satisfaction. By strategically addressing these areas of development, the application can remain relevant, competitive, and well-received by its user base.

Subject Importance

This algorithm’s importance is not measures not only in the profit of social media apps, but also its contribution to the development and growth of social interactions on the digital landscape.

With the Social Media Network algorithm, any social media application serves as a platform for users to communicate and interact with their social circles. In an increasingly digital world, having a centralized and user-friendly space for social connections is vital for maintaining relationships, sharing updates, and fostering communication.

Also, the algorithm is very community supporting. It facilitates the creation and nurturing of communities. Users can form groups based on shared interests, affiliations, or common goals. This community-building aspect is essential for fostering a sense of belonging and creating spaces where users can engage with like-minded individuals.

REFERENCES :

One of the biggest appreciation goes to HBO and their “Silicon Valley” series. I got the idea to create this algorithm from there. In one of the episodes main characters were discussing the implementation of neural networks to solve a problem that is close to mine. And I decided to go easier and test my knowledge of data structures.

Where I revised all the lectures:

Link to youtube playlist

Link to youtube playlist

Links that helped:

Graph Data Structure And AlgorithmsGeeksforGeekshttps://www.geeksforgeeks.org › …

Graph Data StructureProgramizhttps://www.programiz.com › …

Introduction to Graphs - Data Structure and Algorithm ...GeeksforGeekshttps://www.geeksforgeeks.org › ...