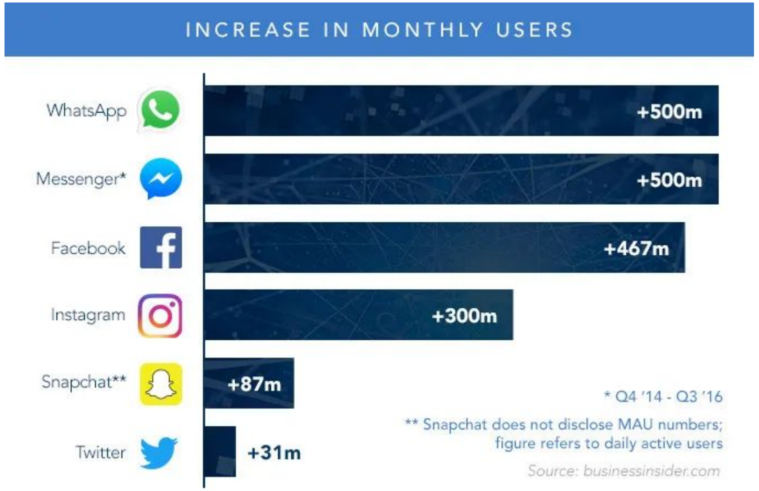
Social Network Analysis

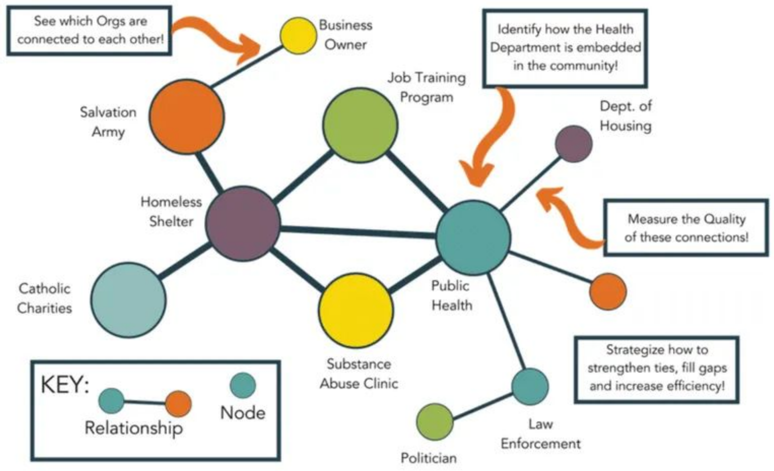
Nguyen The Duy Khanh - 005019181

Problem Statement

- A social network is incredibly useful for maintaining friendships, reaching out, and following your favorite celebrity, friends (e.g Twitter, Facebook)
- Twitter and Facebook offer many features that allow interactions between end-users and followers, and one of them is retweeting. Retweet is an important concept as it determines how trends form and how tweets go viral.
- Social network analysis can help us understand the why and how the feature's behavior makes great impact on users behind the scenes at a larger scale.



Overview



See which Orgs are connected to each other!
Business Owner
Identify how the Health Department is embedded in the community!
Job Training Program
Salvation Army
Dept. of Housing
Homeless Shelter
Measure the Quality of these connections!
Catholic Charities
Public Health
Substance Abuse Clinic
Strategize how to strengthen ties, fill gaps and increase efficiency!
KEY:
Relationship
Node
Law Enforcement
Politician

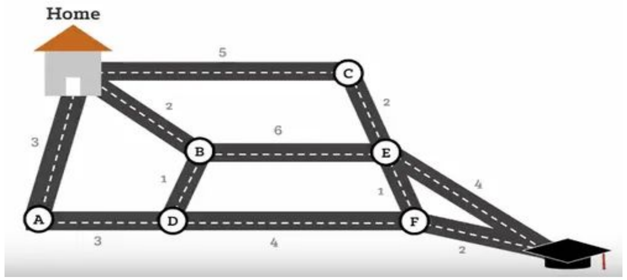
Social network analysis is the process of investigating social structures using networks and graph theories. It combines various techniques for analyzing the structure of social networks and theories that aim to explain the underlying dynamics and patterns observed in these structures. It is an inherently interdisciplinary field originally from social psychology, statistics, and graph theory.

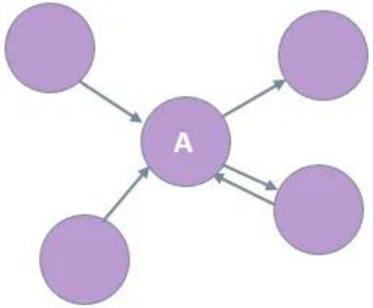
Core functionalities

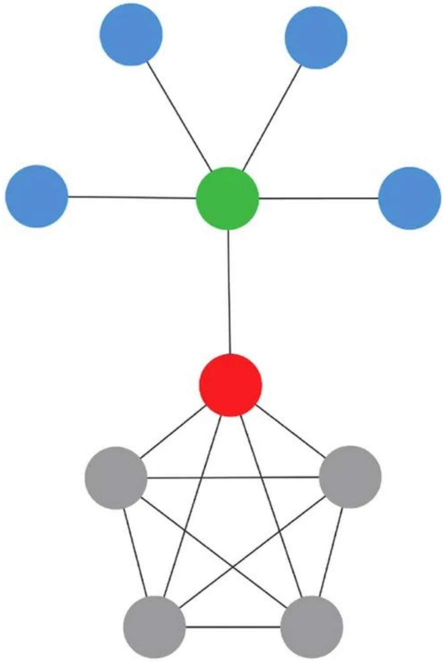
a. **Degree centrality**
- Find the popularity of each user in the social network based on the number of connections

b. **Shortest Path**
- To find the shortest route between two users or a user and a clique when sharing a similar characteristics (e.g influencer outreach, content promotion)

c. **Betweenness centrality**
- Find the bottleneck users or the individuals who play a “bridge-spanning” role in social network

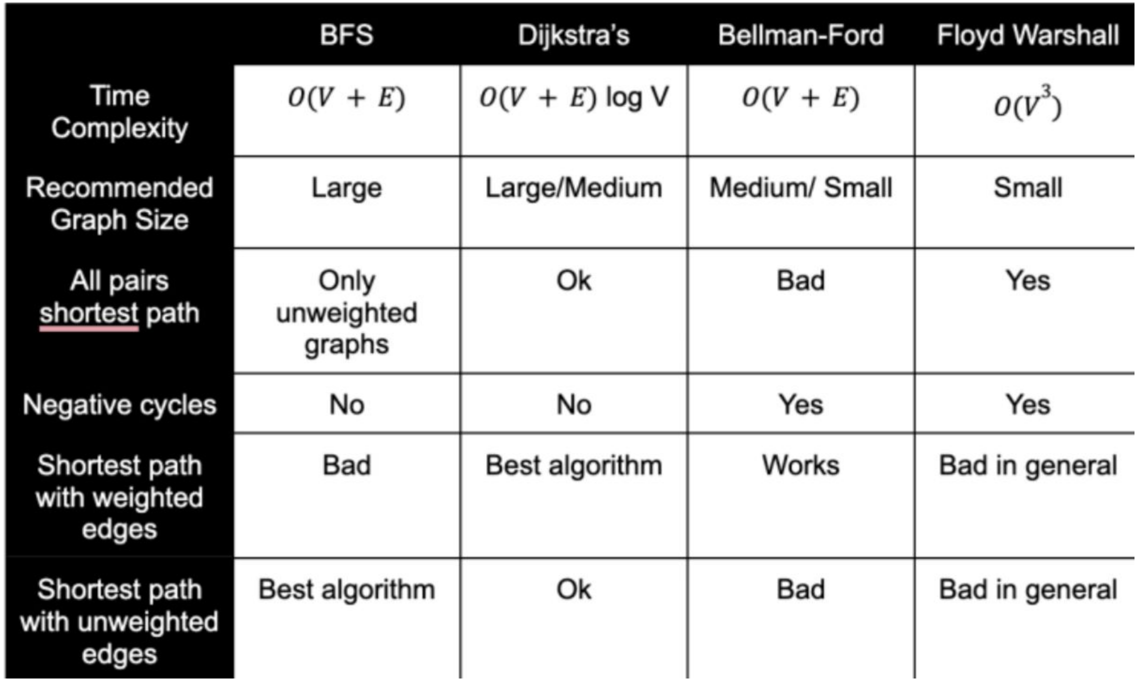






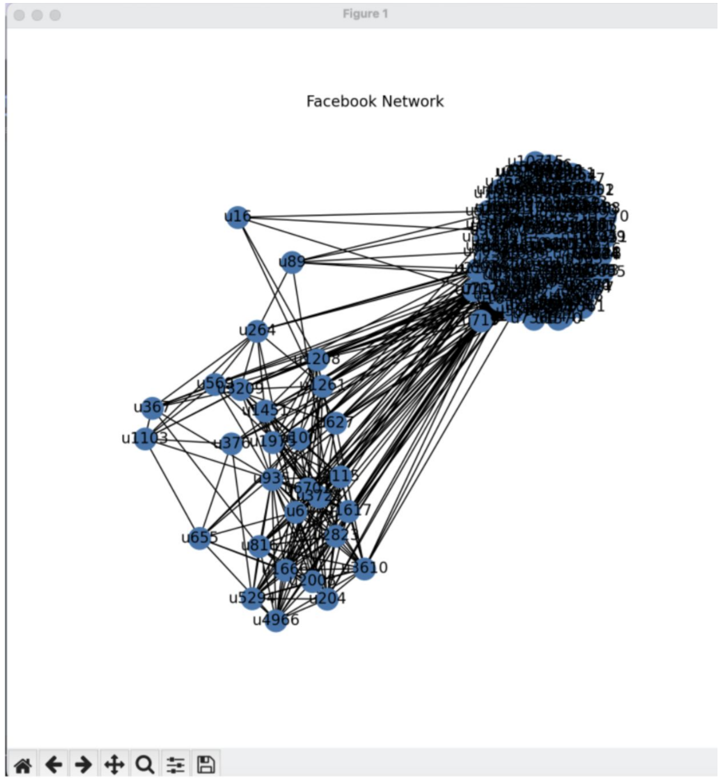
Shortest path analysis

- Facebook has 3 billions active users worldwide
- If considering the similar characteristics between users, it would be best with Dijkstra's. Otherwise, BFS would be the most suitable algorithm.
- BFS is suitable for large graph size with lowest time complexity



	BFS	Dijkstra's	Bellman-Ford	Floyd Warshall
Time Complexity	$O(V + E)$	$O(V + E)$ log V	$O(V + E)$	$O(V^3)$
Recommended Graph Size	Large	Large/Medium	Medium/ Small	Small
All pairs shortest path	Only unweighted graphs	Ok	Bad	Yes
Negative cycles	No	No	Yes	Yes
Shortest path with weighted edges	Bad	Best algorithm	Works	Bad in general
Shortest path with unweighted edges	Best algorithm	Ok	Bad	Bad in general

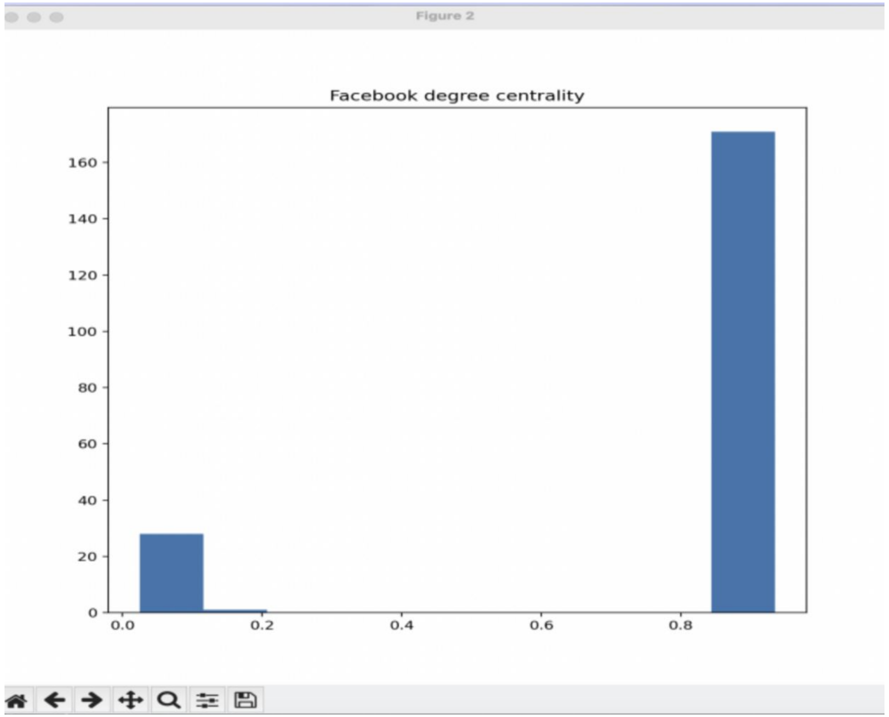
Social Network Dataset

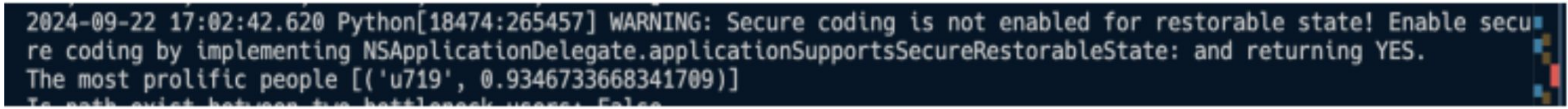


- Simulated Facebook with 56519 users with 72900 connections as an indirect graph where users can connect bidirectional.
- For better visualization, sampling the social network with 200 users which has most connections (degree)
- For performance optimization, the dataset will keep as original or at the rate of 100, 1000, 10000.

Find important people

- By using degree centrality, the most important people will be shown amongst the users
- **Result:** User 719 has the most degree centrality 0.9346733668341709

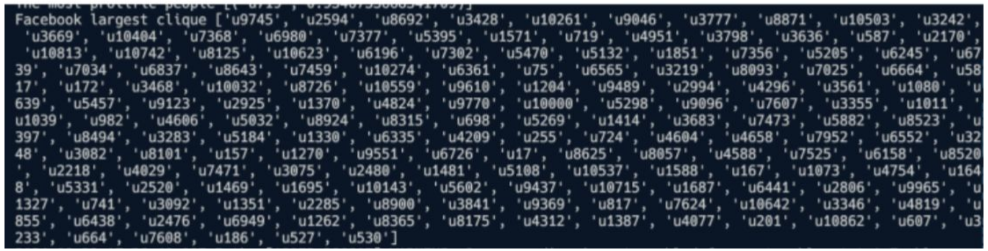




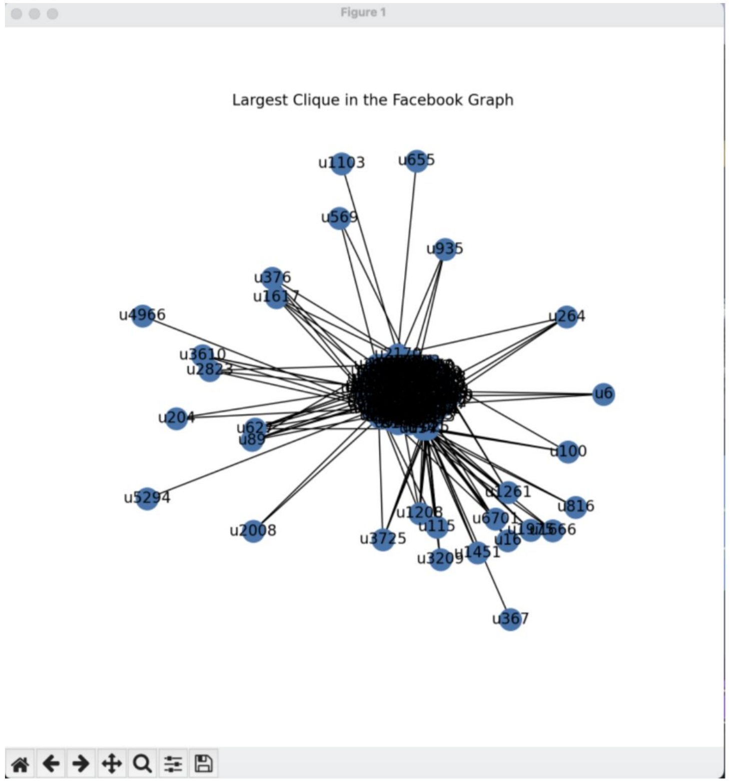
```
2024-09-22 17:02:42.620 Python[18474:265457] WARNING: Secure coding is not enabled for restorable state! Enable secu
re coding by implementing NSApplicationDelegate.applicationSupportsSecureRestorableState: and returning YES.
The most prolific people [('u719', 0.9346733668341709)]
Is path exist between two bottleneck users: False
```

Find largest community

- By finding all the cliques within a social network, the largest clique will be found by sorting it and take the maximum number of users within a clique.
- **Results:** [‘u9745’, ...] is the largest clique.



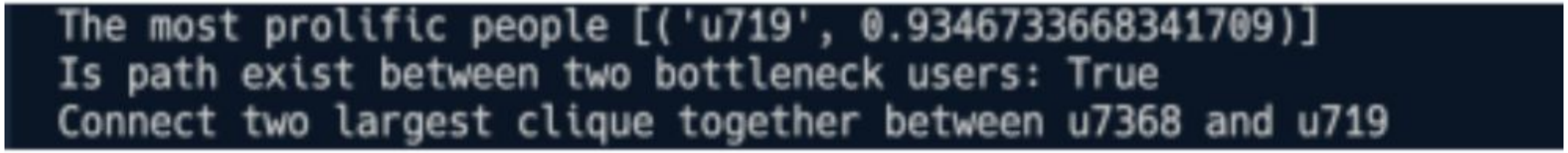
```
The most prolific people [('u719', 0.[illegible])]
Facebook largest clique ['u9745', 'u2594', 'u8692', 'u3428', 'u10261', 'u9046', 'u3777', 'u8871', 'u10503', 'u3242',
 'u3669', 'u10404', 'u7368', 'u6980', 'u7377', 'u5395', 'u1571', 'u719', 'u4951', 'u3798', 'u3636', 'u587', 'u2170',
 'u10813', 'u10742', 'u8125', 'u10623', 'u6196', 'u7302', 'u5470', 'u5132', 'u1851', 'u7356', 'u5205', 'u6245', 'u67
39', 'u7034', 'u6837', 'u8643', 'u7459', 'u10274', 'u6361', 'u75', 'u6565', 'u3219', 'u8093', 'u7025', 'u6664', 'u58
17', 'u172', 'u3468', 'u10032', 'u8726', 'u10559', 'u9610', 'u1204', 'u9489', 'u2994', 'u4296', 'u3561', 'u1080', 'u
639', 'u5457', 'u9123', 'u2925', 'u1370', 'u4824', 'u9770', 'u10000', 'u5298', 'u9096', 'u7607', 'u3355', 'u1011', '
u1039', 'u982', 'u4606', 'u5032', 'u8924', 'u8315', 'u698', 'u5269', 'u1414', 'u3683', 'u7473', 'u5882', 'u8523', 'u
397', 'u8494', 'u3283', 'u5184', 'u1330', 'u6335', 'u4209', 'u255', 'u724', 'u4604', 'u4658', 'u7952', 'u6552', 'u32
48', 'u3082', 'u8101', 'u157', 'u1270', 'u9551', 'u6726', 'u17', 'u8625', 'u8057', 'u4588', 'u7525', 'u6158', 'u8520
', 'u2218', 'u4029', 'u7471', 'u3075', 'u2480', 'u1481', 'u5108', 'u10537', 'u1588', 'u167', 'u1073', 'u4754', 'u164
8', 'u5331', 'u2520', 'u1469', 'u1695', 'u10143', 'u5602', 'u9437', 'u10715', 'u1687', 'u6441', 'u2806', 'u9965', 'u
1327', 'u741', 'u3092', 'u1351', 'u2285', 'u8900', 'u3841', 'u9369', 'u817', 'u7624', 'u10642', 'u3346', 'u4819', 'u
855', 'u6438', 'u2476', 'u6949', 'u1262', 'u8365', 'u8175', 'u4312', 'u1387', 'u4077', 'u201', 'u10862', 'u607', 'u3
233', 'u664', 'u7608', 'u186', 'u527', 'u530']
```



Connecting Communities

- With the combination of finding cliques, betweenness centrality and shortest path, connect communities can be implemented with following steps:
  - **Step 1:** Find the cliques of the social network
  - **Step 2:** Find the bottleneck user which serves as a "bridge-spanning role" in the network using betweenness centrality
  - **Step 3:** Calculate the minimum distance between the two bottleneck users to see if there are any connections.
- **Result:** u7368 and u719 can connect to each other as a two bottleneckusers.

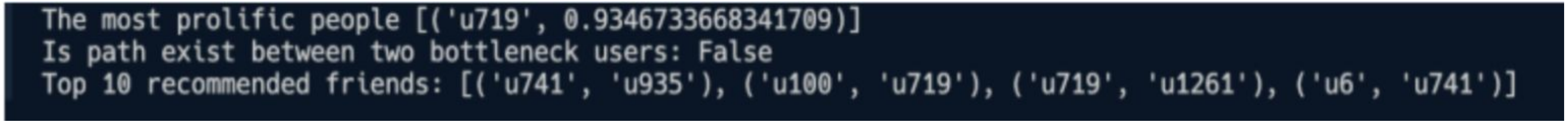
$$B(u) = \sum_{u \neq v \neq w} \frac{\sigma_{v,w}(u)}{\sigma_{v,w}}$$



```
The most prolific people [('u719', 0.9346733668341709)]
Is path exist between two bottleneck users: True
Connect two largest clique together between u7368 and u719
```

Recommend friends

- With shortest path, connect communities can be implemented with following steps:
  - **Step 1:** Find all the combinations of all the user’s friends
  - **Step 2:** If there is no connection between two neighbor nodes, we would recommend them and increase further if there are more intermediate node
  - **Step 3:** Prioritize the top 10 pairs of users that have the highest intermediate users
- **Result: (**u741, u935), (u100,u719), (u719,u1261), (u6,u741) are the top 4 recommended to be friends.

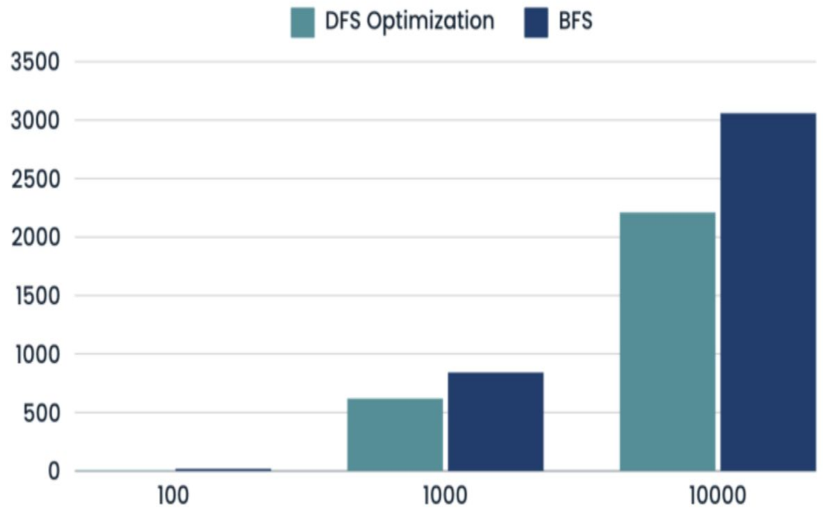


```
The most prolific people [('u719', 0.9346733668341709)]
Is path exist between two bottleneck users: False
Top 10 recommended friends: [('u741', 'u935'), ('u100', 'u719'), ('u719', 'u1261'), ('u6', 'u741')]
```

Optimization

a. **Connect communities**

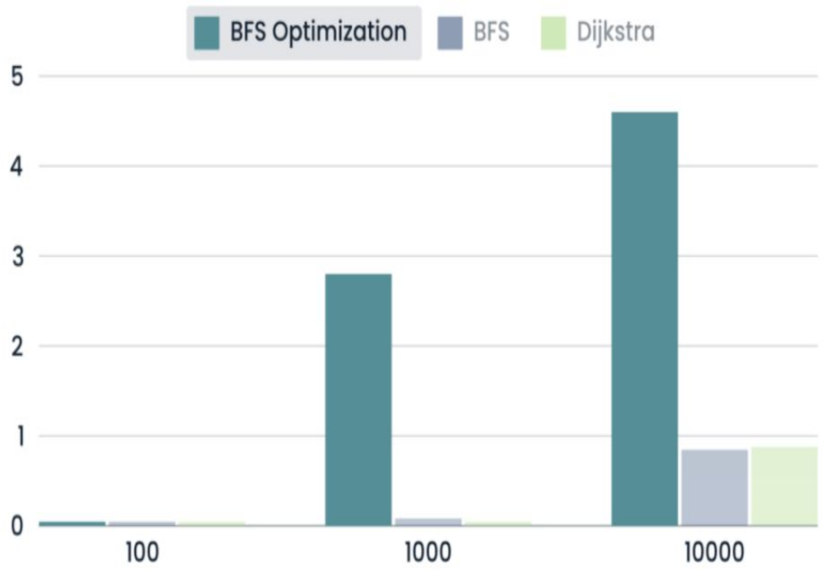
- Instead of finding whether there aren't any connections between the node's neighbors, traversing the path during the initialization, recording the path between two nodes via cache if there is a connection.
- Time complexity: $O(V^2 x(V + E))$
- Space complexity: $O(V * P)$
- **Result:** decreasing execution time from 3200 second to 2221 second



Optimization

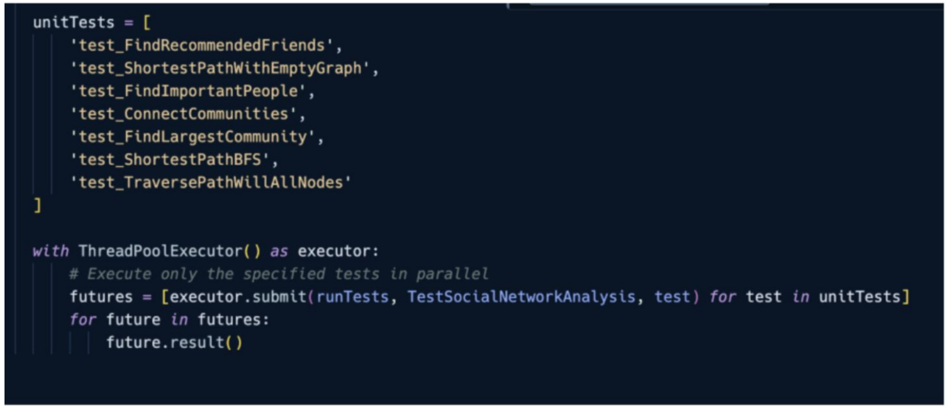
b. Recommended friends

- Instead of using Dijkstra or BFS Optimization with cache traversal path, using BFS would be an optimal choice for indirect un-weight graph.
- If using other algorithm, the time complexity will be higher
- **Result:** the execution time is 0.96 for 10000 nodes, which is the least execution time amongst the three algorithms.



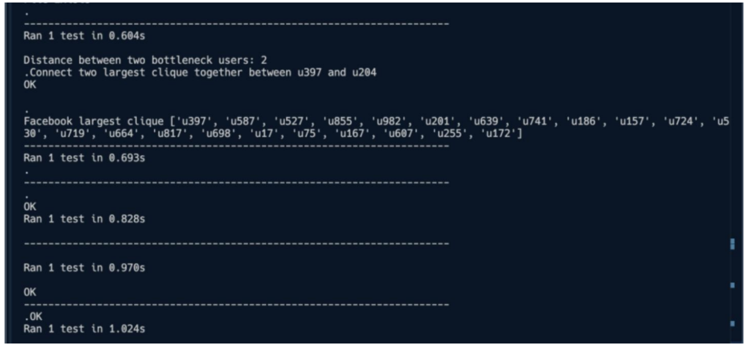
Advance testing

- **Unit Test:** Maintain backward compatibility and documentation for future use. Therefore, creating a set of unit tests involving the Facebook’s features with edge case to avoid breaking backward compatibility..
- **Stress test:** a performance test in the future to give a user transparency of how the algorithm uses their allocation resource. Therefore, creating a set of stress testing involving the Facebook’s features.



```python
unitTests = [
    'test_FindRecommendedFriends',
    'test_ShortestPathWithEmptyGraph',
    'test_FindImportantPeople',
    'test_ConnectCommunities',
    'test_FindLargestCommunity',
    'test_ShortestPathBFS',
    'test_TraversePathWillAllNodes'
]

with ThreadPoolExecutor() as executor:
    # Execute only the specified tests in parallel
    futures = [executor.submit(runTests, TestSocialNetworkAnalysis, test) for test in unitTests]
    for future in futures:
        future.result()
```



```
.
----------------------------------------------------------------------
Ran 1 test in 0.604s

Distance between two bottleneck users: 2
.Connect two largest clique together between u397 and u204
OK

.
Facebook largest clique ['u397', 'u587', 'u527', 'u855', 'u982', 'u201', 'u639', 'u741', 'u186', 'u157', 'u724', 'u5
30', 'u719', 'u664', 'u817', 'u698', 'u17', 'u75', 'u167', 'u607', 'u255', 'u172']
----------------------------------------------------------------------
Ran 1 test in 0.693s
.
----------------------------------------------------------------------
.
OK
Ran 1 test in 0.828s

----------------------------------------------------------------------

Ran 1 test in 0.970s

OK
----------------------------------------------------------------------
.OK
Ran 1 test in 1.024s
```

Future direction / Limitation

- The dataset from social networks is static; therefore, finding a way to always append the new traversal path when there is a new user joining it.
- The optimization technique is not suitable for features that only need to determine a connection between two users (e.g connect communities).
- The optimizing solutions only work towards an unweighted graph. However, if there are any weights or different data structures, the optimization and the current implementation won’t work.
- In a constrained memory environment, the space complexity of O(V * P) when comparing the original space complexity.

Questions?