1. Facebook network

Qikai Feng 106395231
Yi Han 606335602
Xinxin Chang 306405941

1. Facebook network

QUESTION 1: A first look at the network:

QUESTION 1.1: Report the number of nodes and number of edges of the Facebook network.

QUESTION 1.2: Is the Facebook network connected? If not, find the giant connected component (GCC) of the network and report the size of the GCC.

```
Number of nodes: 4039
Number of edges: 88234
Is the network connected? True
Size of the Giant Connected Component (GCC): 4039
```

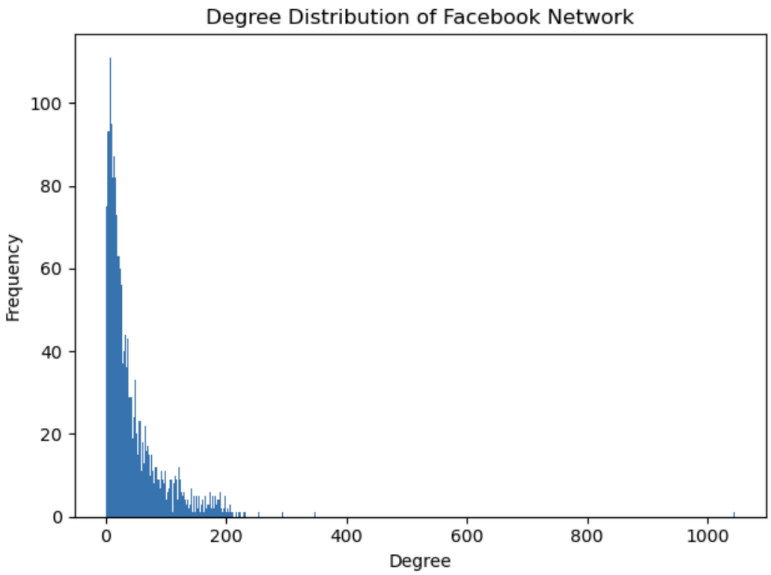
QUESTION 2: Find the diameter of the network. If the network is not connected, then find the diameter of the GCC.

```
Nodes: 4039 Edges: 88234
Computed diameter: 8
```

The diameter is 8.

QUESTION 3: Plot the degree distribution of the facebook network and report the average Degree.

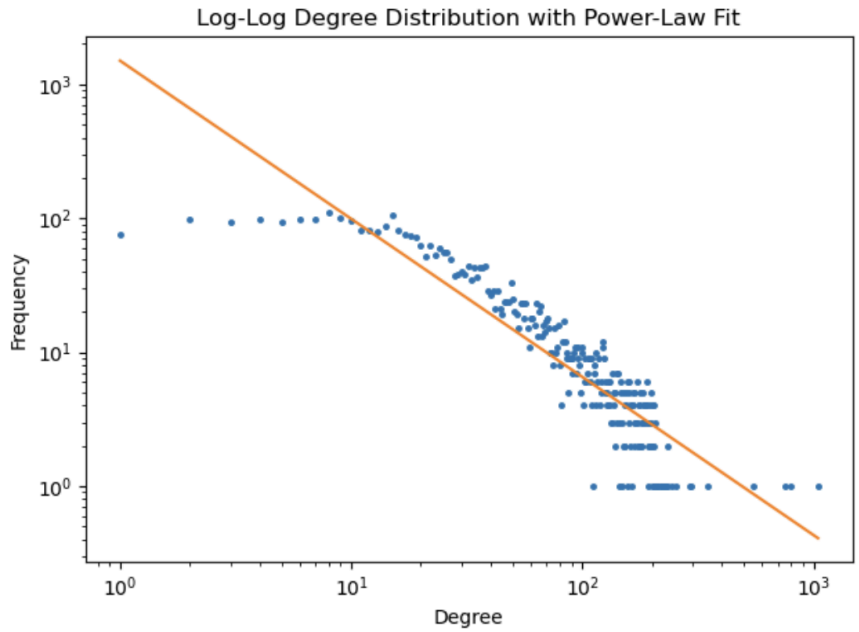
```
Average degree: 43.6910
```



The average degree is 43.6910.

QUESTION 4: Plot the degree distribution of Question 3 in a log-log scale. Try to fit a line to the plot and estimate the slope of the line.

```
Estimated slope of the fit line: -1.1802
```



2. Personalized network

QUESTION 5: Create a personalized network of the user whose ID is 1. How many nodes and edges does this personalized network have?

```
Personalized network nodes: 348
Personalized network edges: 2866
```

QUESTION 6: What is the diameter of the personalized network? Please state a trivial upper and lower bound for the diameter of the personalized network.

Diameter of the personalized network (user ID 1):

$h = 2$

Trivial bounds for the diameter:

Lower bound: 1 (any non-trivial connected graph with at least one edge has diameter ≥ 1)

Upper bound: 347 (in the worst case, a path graph on 348 nodes has diameter = 348 – 1 = 347)

QUESTION 7: In the context of the personalized network, what is the meaning of the diameter of the personalized network to be equal to the upper bound you derived in Question 6. What is the meaning of the diameter of the personalized network to be equal to the lower bound you derived in Question 6 (assuming there are more than 3 nodes in the personalized network)?

Diameter = upper bound (2): This means there is at least one pair of the user's friends (neighbors) who are not directly connected to each other. To go from one of those friends to the other, you must pass through the central user. In other words, not all of the user's friends are mutual friends.

Diameter = lower bound (1): This means every pair of nodes in the personalized network is directly adjacent. In particular, every friend of the user is also friends with every other friend. The ego plus its neighbors therefore form a clique of mutual friendships.

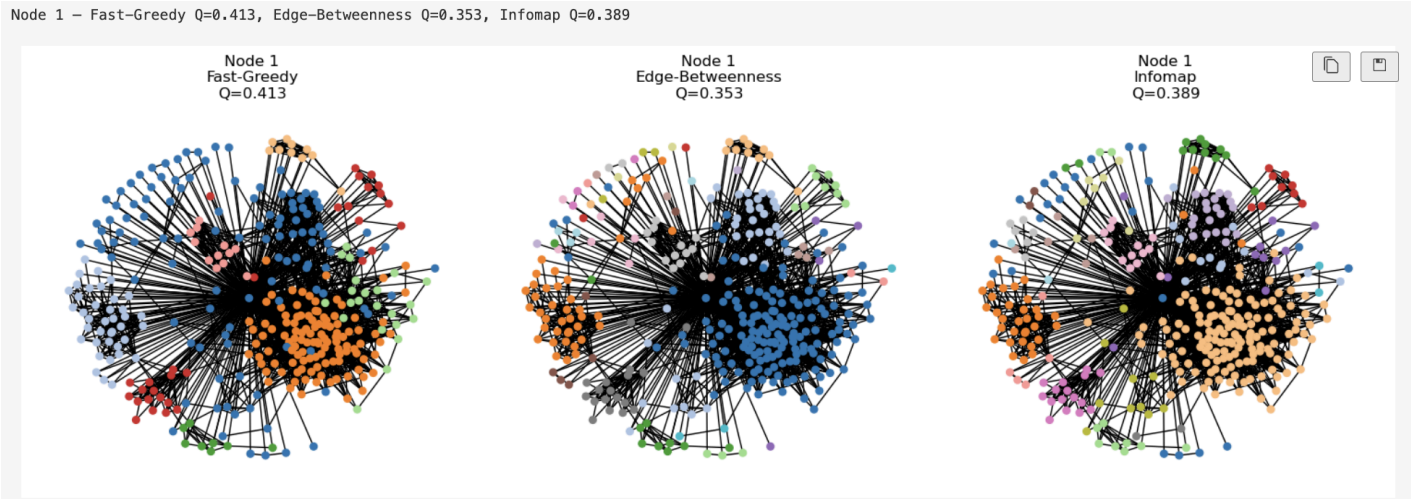
3. Core node's personalized network

QUESTION 8: How many core nodes are there in the Facebook network. What is the average degree of the core nodes?

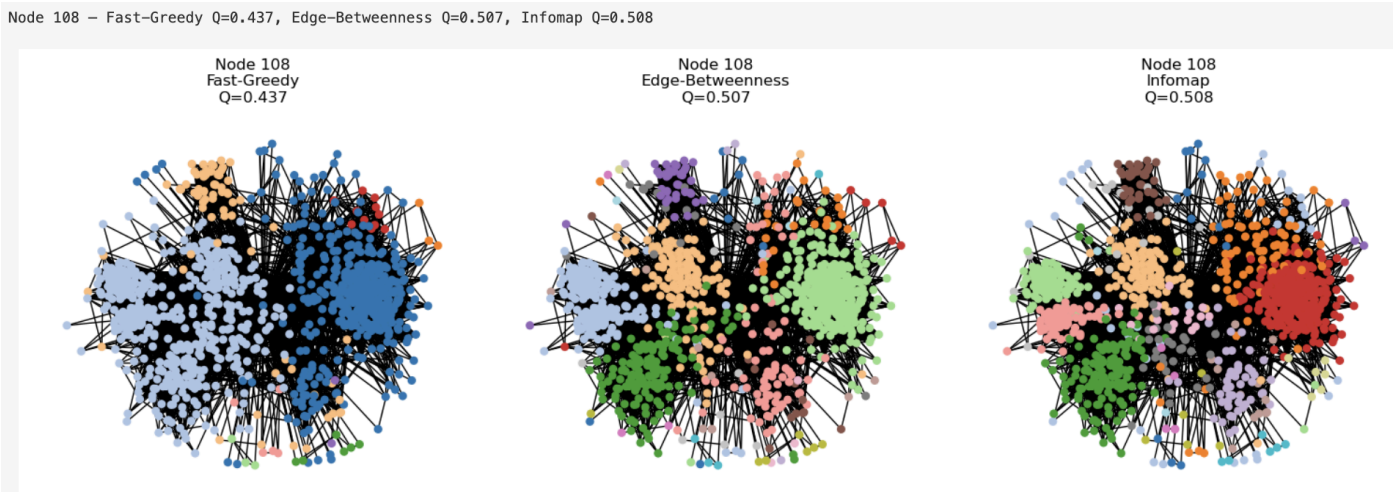
```
Number of core nodes (degree > 200): 40
Average degree of core nodes: 279.38
```

3.1. Community structure of core node's personalized network

QUESTION 9:

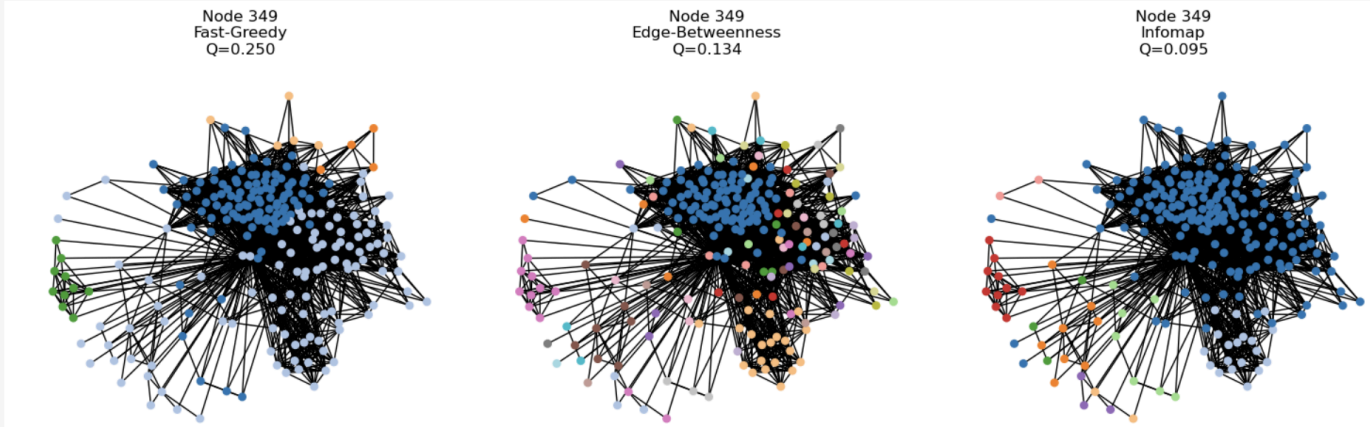


The above figure shows that for core node 1, Fast-Greedy yields the highest modularity (Q = 0.413), indicating the strongest community separation. Edge-Betweenness (Q = 0.353) and Infomap (Q = 0.389) detect smaller or more fragmented communities.



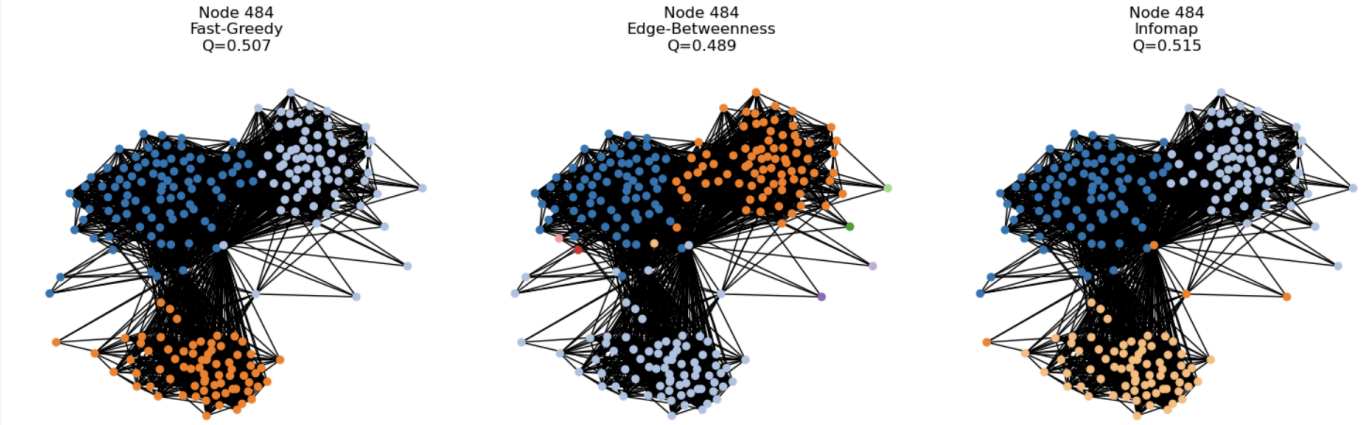
The above figure shows that for core node 108, Infomap yields the highest modularity (Q = 0.508), closely followed by Edge-Betweenness (Q = 0.507), while Fast-Greedy produces a lower score (Q = 0.437), indicating that Infomap and Edge-Betweenness detect tighter, more pronounced communities.

```
Node 349 — Fast-Greedy Q=0.250, Edge-Betweenness Q=0.134, Infomap Q=0.095
```



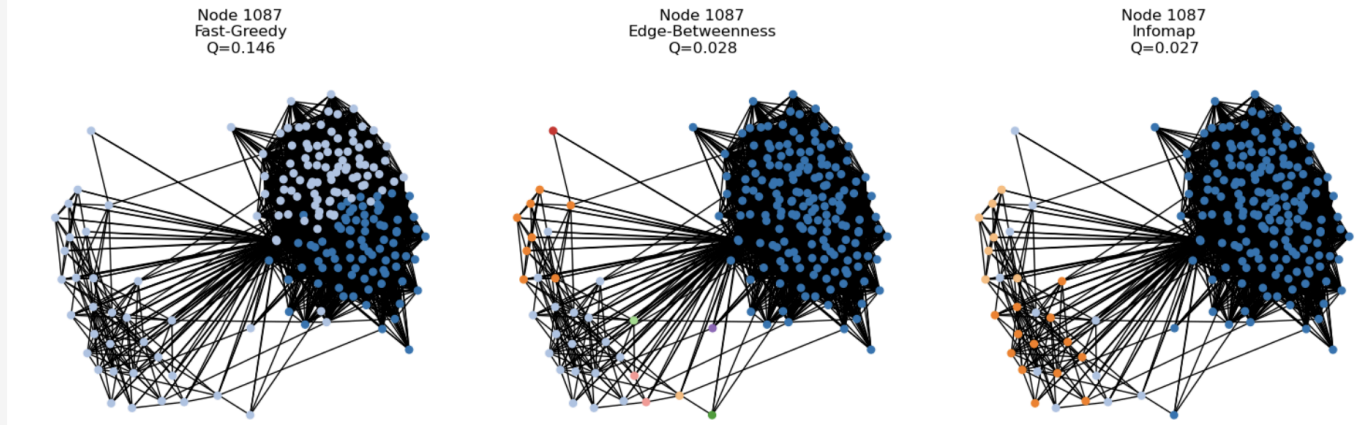
The above figure shows that for core node 349, Fast-Greedy produces the highest modularity (Q = 0.250), whereas Edge-Betweenness (Q = 0.134) and Infomap (Q = 0.095) yield substantially lower scores, suggesting a relatively weak and diffuse community structure overall.

```
Node 484 — Fast-Greedy Q=0.507, Edge-Betweenness Q=0.489, Infomap Q=0.515
```



The above figure shows that for core node 484, Infomap achieves the highest modularity (Q = 0.515), with Fast-Greedy close behind (Q = 0.507) and Edge-Betweenness slightly lower (Q = 0.489), indicating consistent identification of two well-separated communities.

```
Node 1087 — Fast-Greedy Q=0.146, Edge-Betweenness Q=0.028, Infomap Q=0.027
```



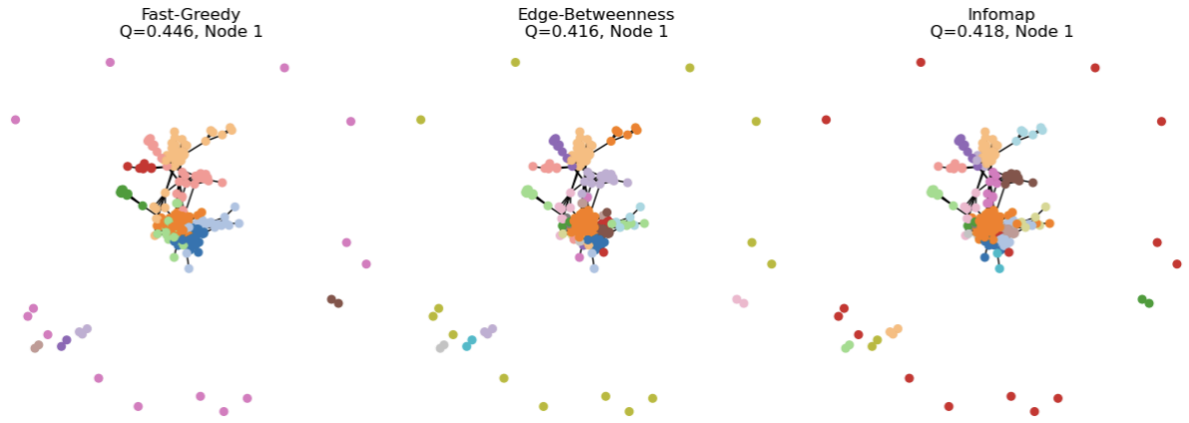
The above figure shows that for core node 1087, Fast-Greedy yields the strongest community separation (Q = 0.146), while both Edge-Betweenness (Q = 0.028) and Infomap (Q = 0.027) detect almost no modular structure, reflecting a densely interlinked network with few distinct groups.
The table below lists the modularity (Q) values for the five core nodes across the three community-detection algorithms for quick comparison.

Core Node ID	Fast-Greedy Q	Edge-Betweenness Q	Infomap Q
1	0.413	0.353	0.389
108	0.437	0.507	0.508
349	0.250	0.134	0.095
484	0.507	0.489	0.515
1087	0.146	0.028	0.027

3.2. Community structure with the core node removed

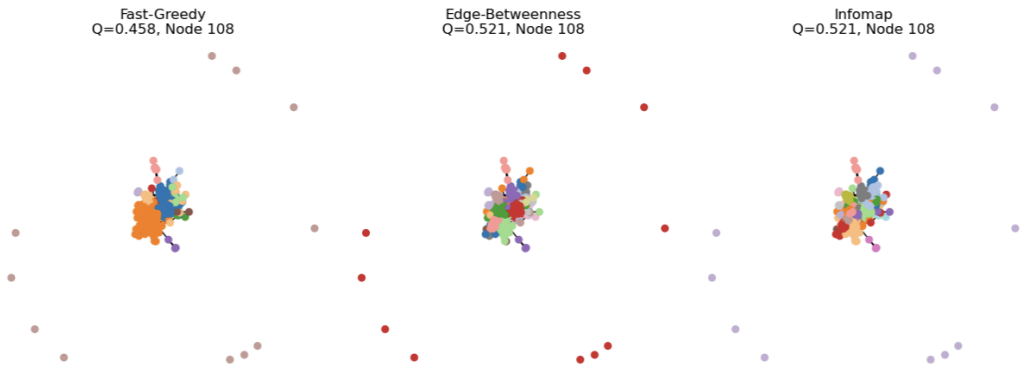
QUESTION 10:

```
Core node 1: Fast-Greedy Q_mod=0.446, Edge-Betweenness Q_mod=0.416, Infomap Q_mod=0.418
```



Removal of node 1 increases Q for all algorithms, showing that its many inter-community links had been obscuring finer clusters.

```
Core node 108: Fast-Greedy Q_mod=0.458, Edge-Betweenness Q_mod=0.521, Infomap Q_mod=0.521
```

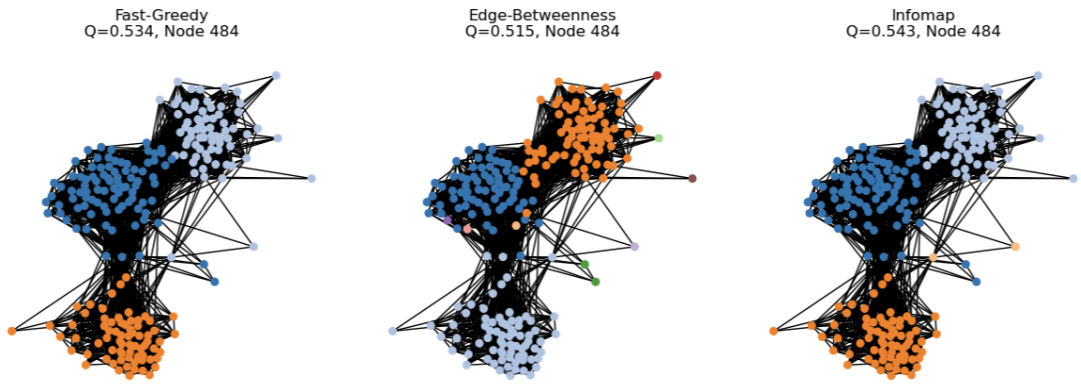


All three algorithms yield modest gains, indicating tighter, more pronounced communities once the hub is gone.

```
Core node 349: Fast-Greedy Q_mod=0.246, Edge-Betweenness Q_mod=0.151, Infomap Q_mod=0.245
```

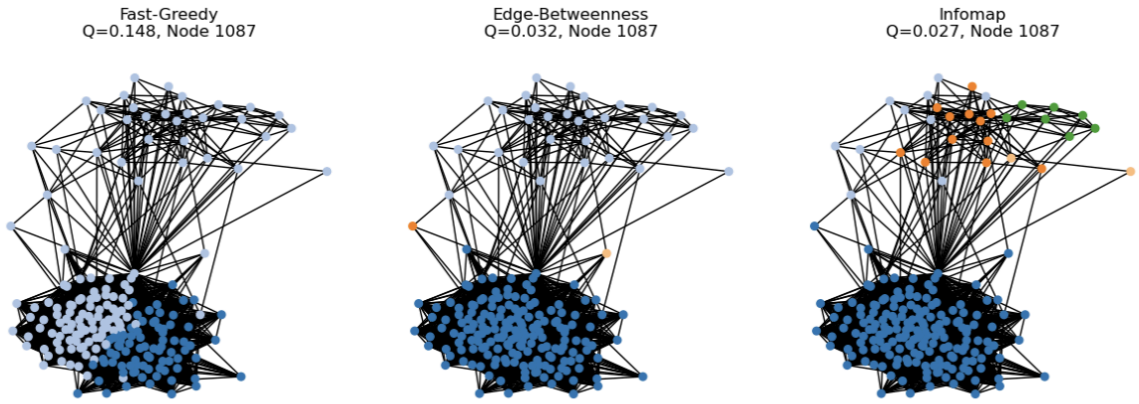
Fast-Greedy's slight drop suggests an already weak partition, whereas Infomap sees a large improvement, uncovering stronger groupings in the absence of the central connector.

```
Core node 484: Fast-Greedy Q_mod=0.534, Edge-Betweenness Q_mod=0.515, Infomap Q_mod=0.543
```



All algorithms show similar increases, reflecting clearer separation into two main clusters after node removal.

```
Core node 1087: Fast-Greedy Q_mod=0.148, Edge-Betweenness Q_mod=0.032, Infomap Q_mod=0.027
```



Minimal change across all methods indicates that this network's community structure was already diffuse and little affected by its highest-degree node.
Overall, except for Fast-Greedy on node 349 and Infomap on node 1087, removing the core node consistently increases modularity, confirming that high-degree hubs tend to bridge communities and mask the network's intrinsic modular structure.

3.3. Characteristic of nodes in the personalized network

QUESTION 11: Write an expression relating the Embeddedness between the core node and a non-core node to the degree of the non-core node in the personalized network of the core node.

For any non-core node u in the personalized network P(v) of core node v, its degree in P(v) counts (a) the one edge back to v, plus (b) one edge for each mutual friend. Hence if we let

$\deg_p(u)$ = degree of u in the personalized network P(v), and emb(u,v) = number of mutual friends of u and v,

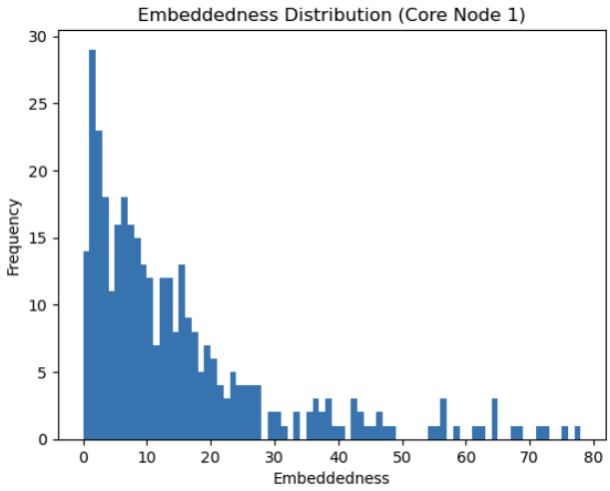
then

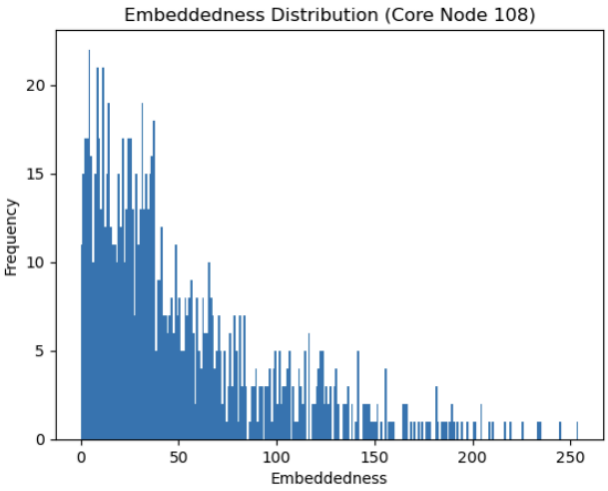
$\deg_p(u) = 1 + \text{emb}(u,v)$

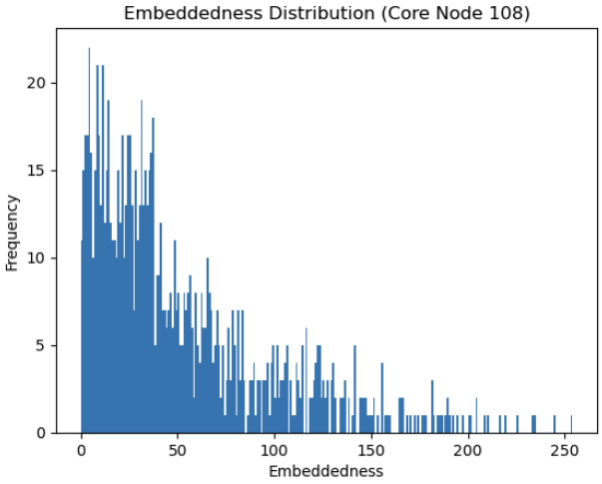
so equivalently

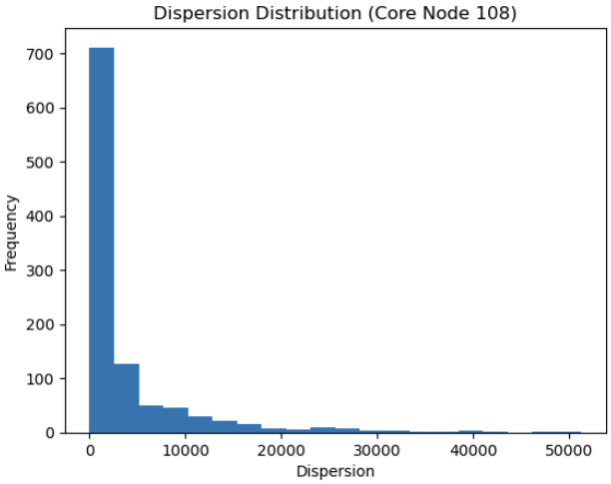
$\text{emb}(u,v) = \deg_p(u) - 1.$

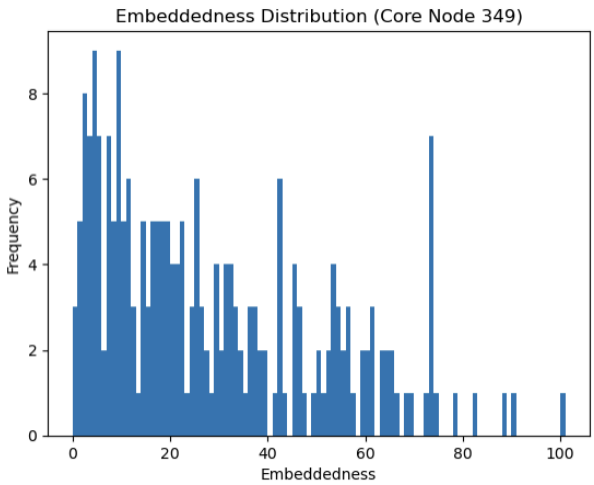
QUESTION 12:

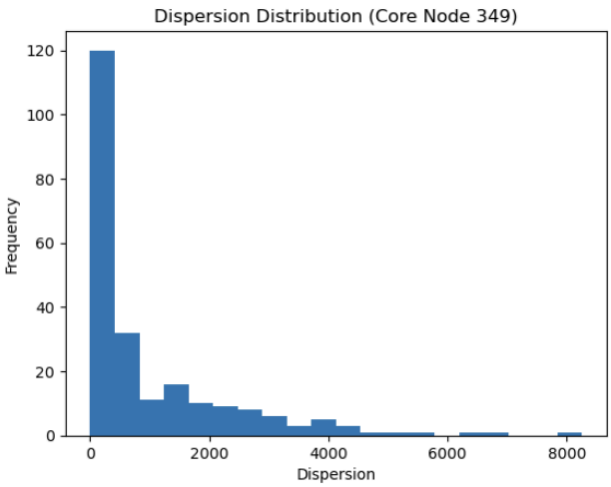

Embeddedness Distribution (Core Node 1)
Frequency
Embeddedness

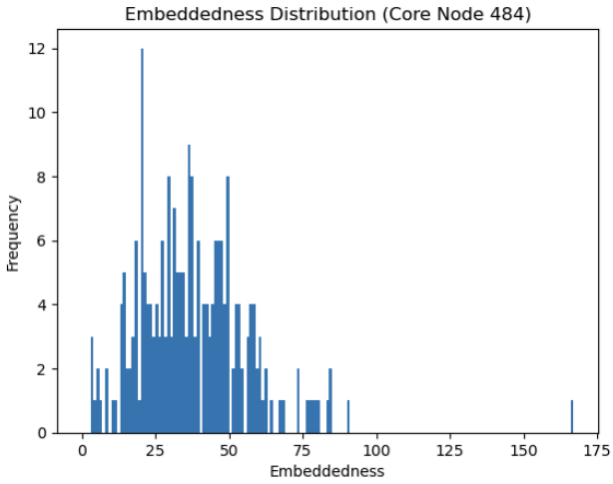

Embeddedness Distribution (Core Node 108)
Frequency
Embeddedness

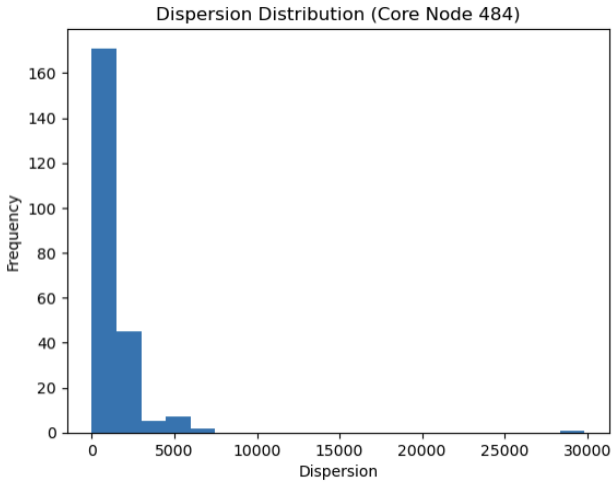

Embeddedness Distribution (Core Node 108)
Frequency
Embeddedness

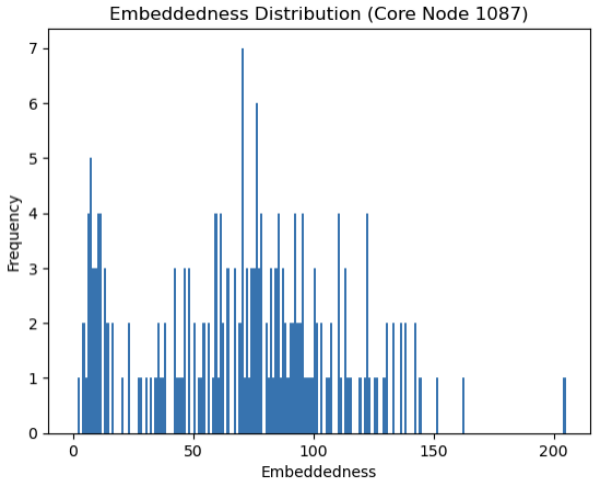

Dispersion Distribution (Core Node 108)
Frequency
Dispersion

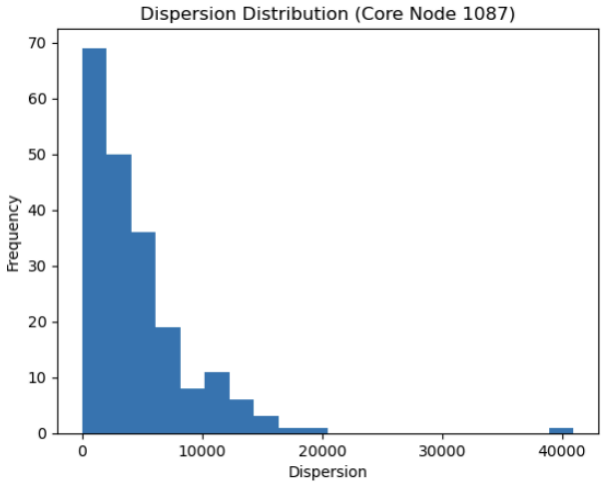

Embeddedness Distribution (Core Node 349)
Frequency
Embeddedness


Dispersion Distribution (Core Node 349)
Frequency
Dispersion


Embeddedness Distribution (Core Node 484)
Frequency
Embeddedness
0
2
4
6
8
10
12
0
25
50
75
100
125
150
175


Dispersion Distribution (Core Node 484)
Frequency
Dispersion
0
20
40
60
80
100
120
140
160
0
5000
10000
15000
20000
25000
30000

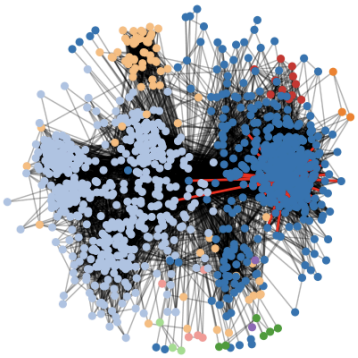

Embeddedness Distribution (Core Node 1087)
Frequency
Embeddedness
0
1
2
3
4
5
6
7
0
50
100
150
200


Dispersion Distribution (Core Node 1087)
Frequency
Dispersion
0
10
20
30
40
50
60
70
0
10000
20000
30000
40000

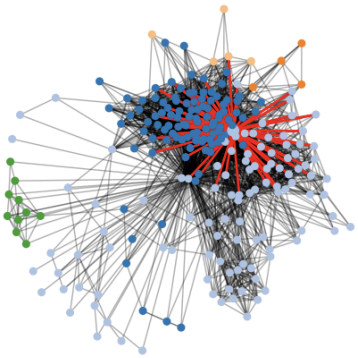
QUESTION 13:

Core Node 1, max dispersion

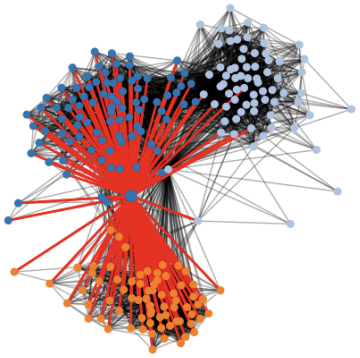
Core Node 108, max dispersion



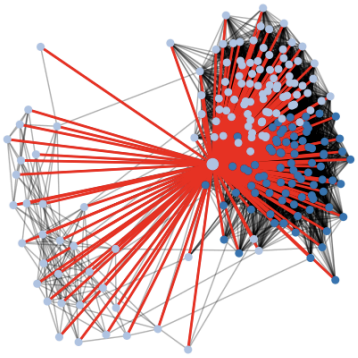
Core Node 349, max dispersion



Core Node 484, max dispersion

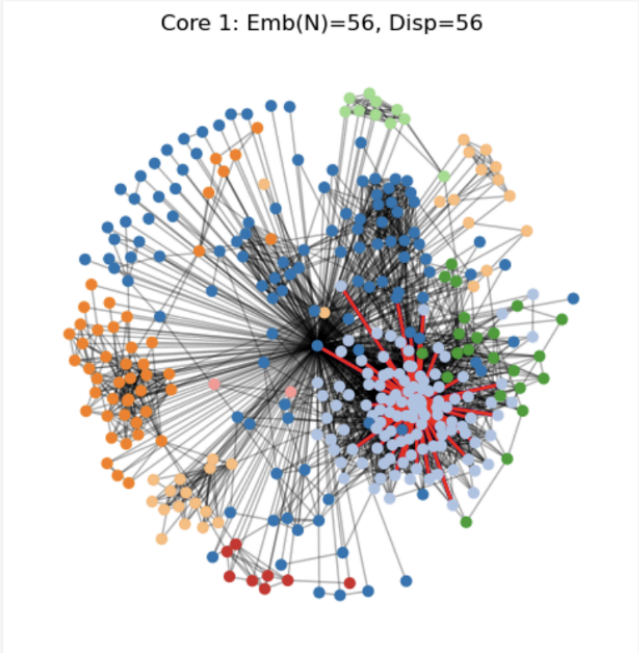


Core Node 1087, max dispersion

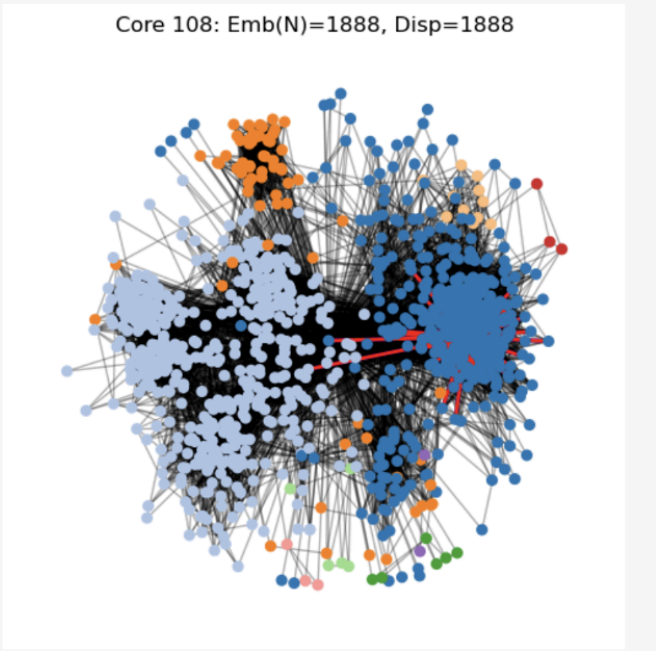


QUESTION 14:

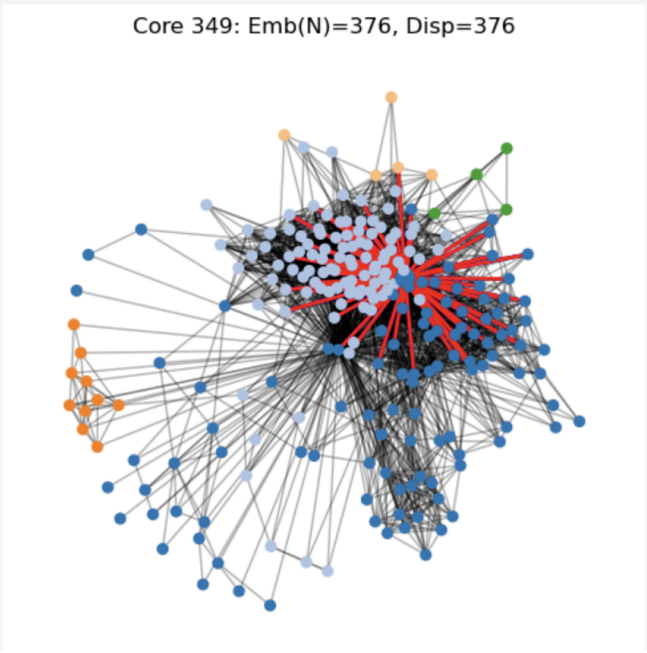
```
Core Node 1: max embeddedness node = 56, max dispersion node = 56
```



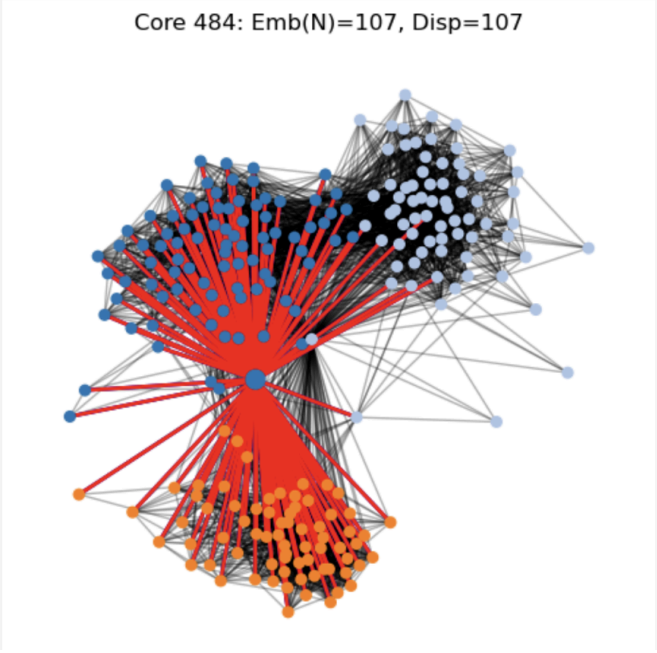

```
Core Node 108: max embeddedness node = 1888, max dispersion node = 1888
```



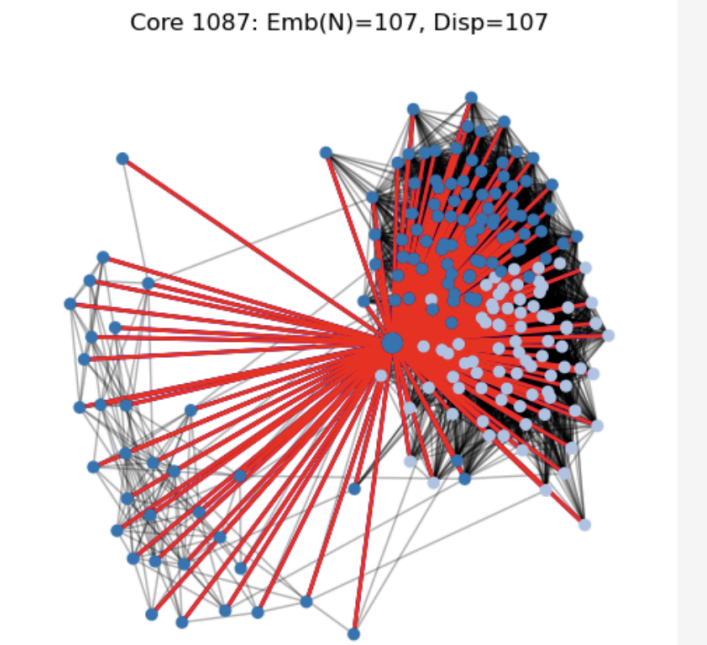

```
Core Node 349: max embeddedness node = 376, max dispersion node = 376
```




```
Core Node 484: max embeddedness node = 107, max dispersion node = 107
```



```
Core Node 1087: max embeddedness node = 107, max dispersion node = 107
```



QUESTION 15: Use the plots from Question 13 and 14 to explain the characteristics of a node revealed by each of this measure.

From our plots in Q13 (red highlights) and Q14 (blue highlights), we see two distinct roles:

Embeddedness

The node with the highest embeddedness sits deep inside one densely connected pocket of the core's friends.

It shares many mutual friends who are themselves all tightly linked—so removing it wouldn't break up that pocket.

Interpretation: high embeddedness ⇒ a "core clique" member, redundantly connected within one community.

Dispersion

The node with the highest dispersion reaches out into multiple, otherwise separate friend-clusters.

Its incident edges (red) span across different colored communities, so without it those clusters would drift apart.

Interpretation: high dispersion ⇒ a "bridge" or "broker," holding together disparate parts of the core's network.

When they coincide

Occasionally the same node tops both measures—meaning it's both deeply embedded in one clique and links out to others.

Such a node combines the redundancy of a core member with the reach of a bridge.

In short, embeddedness flags who is most locally connected, while dispersion flags who is most globally connective across communities.

QUESTION 16:

```
|Nr| = 11
```

QUESTION 17:

```
|Nr| = 11
Average accuracy:
 Common Neighbors: 0.8274
 Jaccard:          0.7885
 Adamic-Adar:      0.8196

Best algorithm: Common Neighbors (accuracy 0.8274)
```

2. Google+ network

1. Community structure of personal networks

QUESTION 18:

There are 57 personal networks for users with more than 2 circles.

QUESTION 19:

We selected the following 3 user IDs:

109327480479767108490, 115625564993990145546 and 101373961279443806744.

For each user, we plotted both the in-degree and out-degree distributions:

109327480479767108490:

- In-degree distribution: Most nodes have small in-degree values, concentrated around 0-20.
- Out-degree distribution: Few nodes have extremely large out-degree values (long tail).

115625564993990145546:

- In-degree and out-degree distributions show a broader spread, but similarly, most degrees are low.

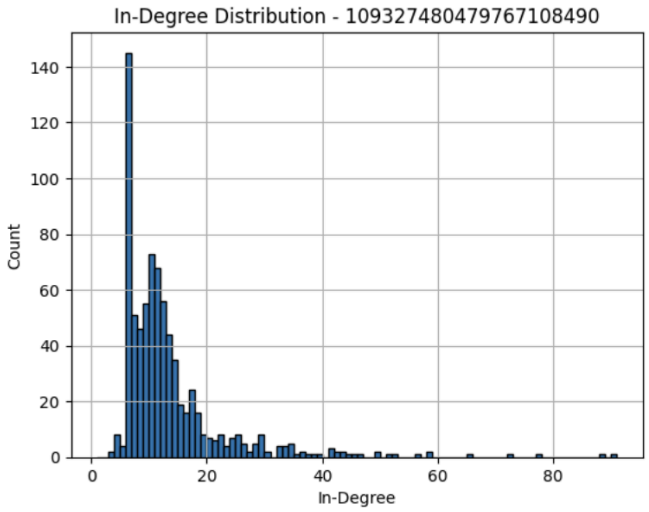
101373961279443806744:

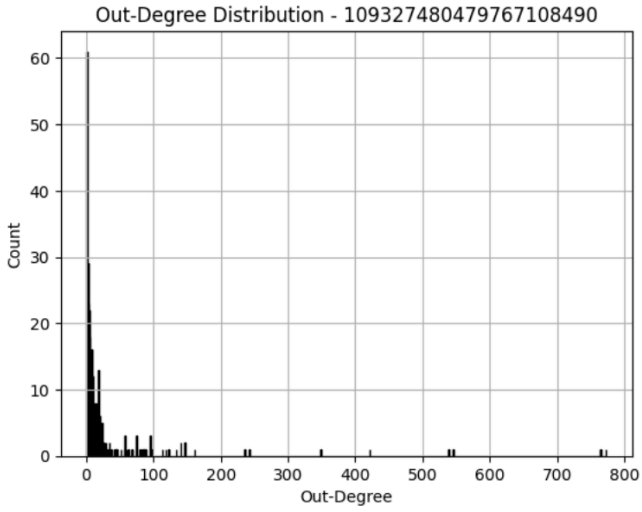
- In-degree and out-degree distributions are even wider, with some degrees reaching into the hundreds or thousands.

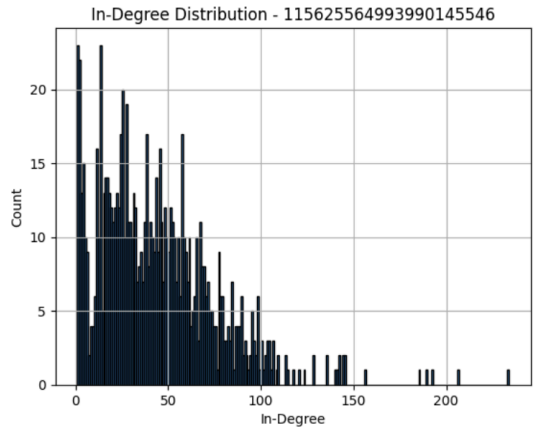
Statistical Summary:

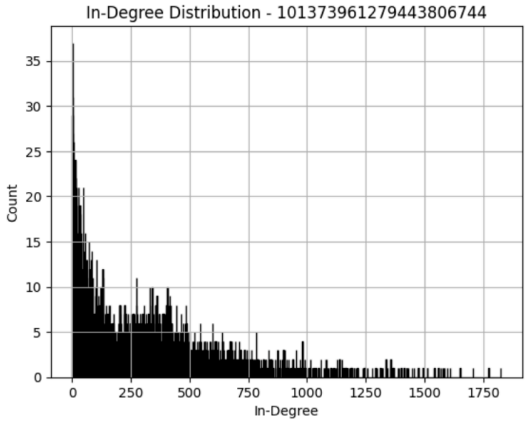
Node ID	Mean In-degree	Variance In-degree	Mean Out-degree	Variance Out-degree
1093274804797671084 90	13.080	95.745	13.080	3842.082
1156255649939901455 46	42.687	1018.555	42.687	8512.612
1013739612794438067 44	297.196	86385.462	297.196	162944.821

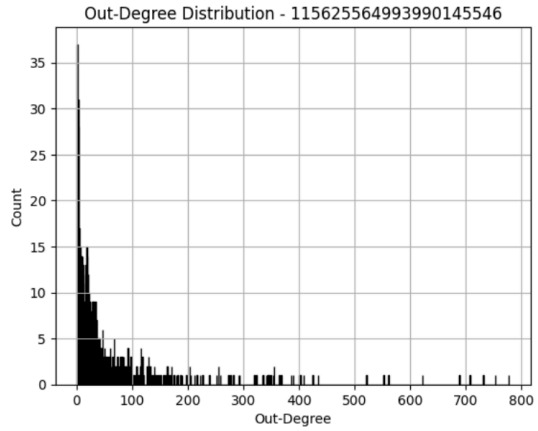
Observation: In all three networks, the out-degree variance is much higher than the in-degree variance, meaning some users follow (or are followed by) many more others.

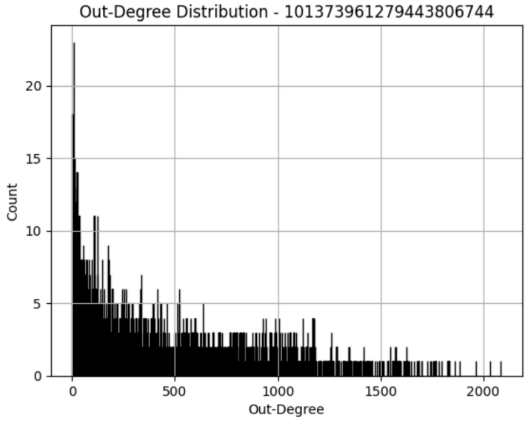








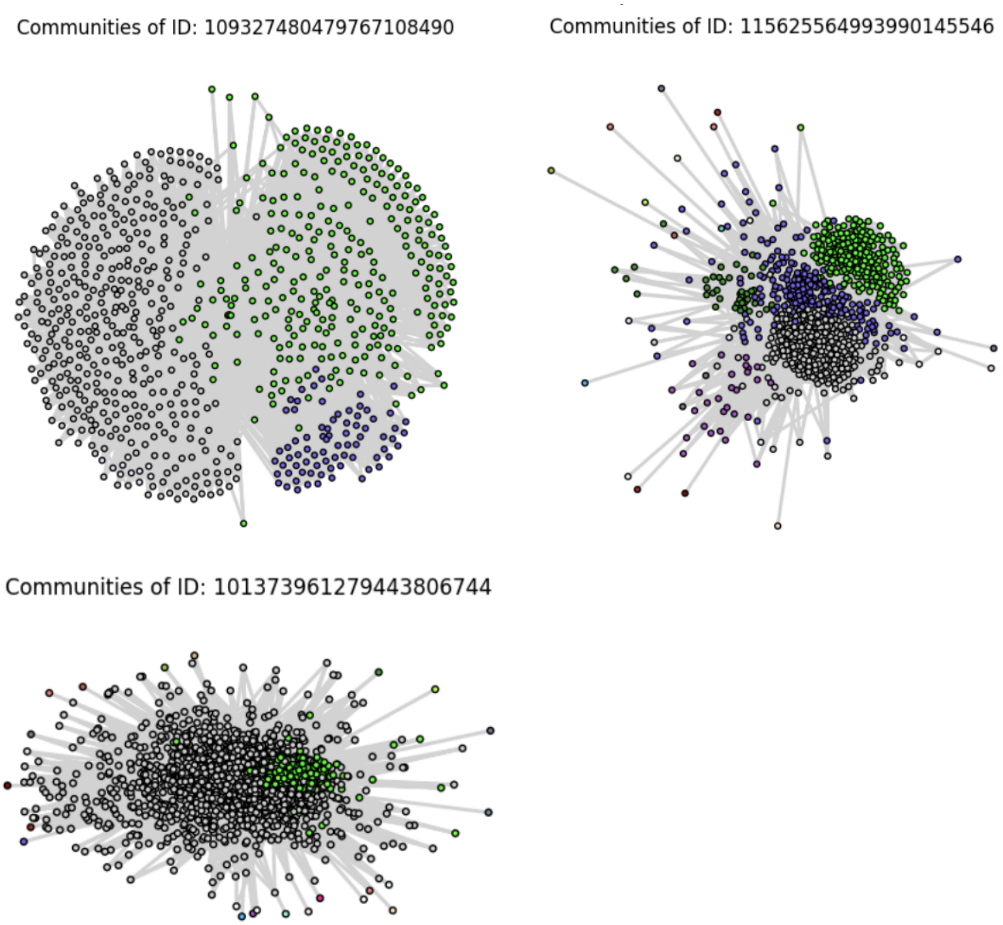




QUESTION 20:

We extracted community structures using the Walktrap community detection algorithm.

Node ID	Number of Communities	Modularity
109327480479767108490	4	0.2798
115625564993990145546	25	0.3231
101373961279443806744	29	0.1951



Observation: All three networks show reasonably good modularity values (> 0.19), indicating that the Walktrap algorithm successfully detected meaningful communities. Networks with more communities (like the third one) tend to have slightly lower modularity.

QUESTION 21:

Homogeneity (h): Measures whether each community contains only nodes from a single user-defined circle. A high homogeneity means nodes within a community are strongly aligned to one original circle.

Completeness (c): Measures whether all nodes from a circle are assigned to the same community. A high completeness indicates that the circles are well preserved in the detected communities.

Mathematically, High **h** implies communities are "pure". High **c** implies circles are "fully captured" by communities.

Both metrics range from 0 to 1 under ideal conditions (but negative values may happen if entropy calculations yield unexpected results).

QUESTION 22:

Node ID	Homogeneity (h)	Completeness (c)	V-measure
109327480479767108490	0.85188512	0.32987391	0.47558710
115625564993990145546	0.44228461	-3.87467181	0.00000000
101373961279443806744	0.00390402	-1.52347843	0.00000000

Interpretation: For user 109327480479767108490, homogeneity is high (~0.85), meaning communities are relatively pure. However, completeness is moderate (~0.33), meaning some circles are fragmented. For users 115625564993990145546 and 101373961279443806744, completeness values are negative, and V-measure is zero. This happens because the community detection poorly captures the original circles, leading to invalid entropy ratios. Negative values suggest serious mismatch between circles and communities. Negative completeness arises when the detected communities do not align with user circles at all. Specifically, the conditional entropy becomes larger than the base entropy, causing c = 1 - (H(K|C)/H(K)) to fall below zero.

3. Cora dataset

QUESTION 23: Idea 1

- In this part, we load the Cora dataset using the Planetoid interface from PyTorch Geometric, applying feature normalization and self-loop augmentation. We implement a two-layer GCN with 16 hidden units per layer and a dropout rate of 0.5 after the first convolution. The model is trained for 200 epochs using the Adam optimizer (learning rate = 0.01, weight_decay = 5×10^{-4}), and we select the checkpoint with the highest validation accuracy for final test-set evaluation.
- Hyperparameter Choices

Hyperparameter	**Value**	**Rationale**
# of layers	2	Standard two-layer GCN suffices for Cora classification
Hidden channels	16	Balances expressivity and overfitting risk
Learning rate	0.01	Stable convergence on small graph

Weight decay	5e-4	Prevents overfitting
Dropout	0.5	Regularizes hidden representations
Epochs	200	Ensures convergence;

```
Epoch 020, Loss: 1.7128, Train: 0.9429, Val: 0.6900, Test: 0.7310
Epoch 040, Loss: 1.3322, Train: 0.9500, Val: 0.7540, Test: 0.7840
Epoch 060, Loss: 0.8966, Train: 0.9714, Val: 0.7780, Test: 0.8000
Epoch 080, Loss: 0.6493, Train: 0.9857, Val: 0.7940, Test: 0.8160
Epoch 100, Loss: 0.5250, Train: 0.9857, Val: 0.8020, Test: 0.8250
Epoch 120, Loss: 0.4725, Train: 0.9929, Val: 0.7880, Test: 0.8190
Epoch 140, Loss: 0.3870, Train: 0.9929, Val: 0.7900, Test: 0.8180
Epoch 160, Loss: 0.3783, Train: 1.0000, Val: 0.7940, Test: 0.8150
Epoch 180, Loss: 0.3224, Train: 1.0000, Val: 0.7920, Test: 0.8160
Epoch 200, Loss: 0.3007, Train: 1.0000, Val: 0.7980, Test: 0.8130
```

- A two-layer GCN with the chosen hyperparameters achieves ~82.5% test accuracy on Cora.

QUESTION 24: Idea 2

In this part, we load the Cora citation network via PyTorch Geometric's Planetoid interface with feature normalization and self-loops. We then generate 64-dimensional node embeddings using Node2Vec (walk_length=20, context_size=10, walks_per_node=10, p=1, q=1) trained for 50 epochs with SparseAdam (lr=0.01). Finally, we extract the 1433-dim bag-of-words text features and form three datasets—Node2Vec-only, Text-only, and their concatenation—and train a 100-tree RandomForest classifier on each for comparison.
Below is the test accuracies of the three feature sets under the initial RandomForest model

```
Node2Vec-only: 0.6530
Text-only:     0.5660
Combined:      0.7160
```

Node2Vec-only outperforms Text-only because structural embeddings capture topic homophily in the citation graph. Papers in the same field often cite each other, which yields higher accuracy than using text features alone.
The Combined feature set outperforms both individual sets because it merges structural and semantic information, providing complementary signals that lead to the best performance.
We further tuned classifiers on the combined feature set:

- **SVM (linear kernel)**
  - Grid over C ∈ {0.01, 0.1, 1, 10, 100}
  - Best C = 0.1 → 5-fold CV = 72.14%, Test = 70.90%

```
Best SVM C: 0.1
CV Accuracy: 0.7214285714285714

Test Accuracy: 0.709
```

- **RandomForest**

- Grid over n_estimators ∈ {50, 100, 200, 500}, max_depth ∈ {None, 10, 20, 50}
- Best: n_estimators = 500, max_depth = none → 5-fold CV = 77.86%, Test = 74.50%

```
Best RF params: {'max_depth': None, 'n_estimators': 500}
CV Accuracy: 0.7785714285714285
Test Accuracy (RF): 0.745
```

- Which one outperforms?
  - Node2Vec-only (65.3%) outperforms Text-only (56.6%), and the Combined (71.6%) outperforms both individual feature sets.
- Why is this the case?
  - Node2Vec captures the citation graph's homophily (papers of the same topic tend to cite each other), yielding stronger structural signals than raw text alone. Text features provide semantic content but miss relational hints. Combined features leverage both structure and semantics, giving the highest accuracy.
- Best classification accuracy: 74.50% test accuracy, achieved by training a classifier (RandomForest) on the concatenated Node2Vec + text feature vectors.

QUESTION 25: Idea 3

p	**Test Accuracy**	**Macro-F1**
0.0	74.99 %	71.51 %
0.1	74.69 %	74.06 %
0.2	73.21 %	72.66 %

In this experiment, we first extracted the giant connected component of the Cora citation graph, yielding 2 485 nodes. Each node carries a 1433-dimensional bag-of-words feature vector, which we used to define softmax-based transition probabilities among its neighbors. For each of the seven classes, we designated its 20 labeled papers as teleport "seeds." From each seed, we ran 1 000 random walks of length 100: at each step, with probability p the walker teleported back to a random seed of that class; otherwise it moved to a neighboring node according to the text-based transition probabilities.

We evaluated three teleportation probabilities (p = 0.0, 0.1, and 0.2) by counting, for every unlabeled node, the total visits it received from walks originating in each class and assigning it the class with the highest visit count. The resulting test accuracies were 74.99 % (macro-F1 = 71.51 %) for p = 0.0, 74.69 % (macro-F1 = 74.06 %) for p = 0.1, and 73.21 % (macro-F1 = 72.66 %) for p = 0.2.

The case p = 0.0 yielded the highest raw accuracy by strictly following the graph structure, but its class balance lagged behind. Introducing a small teleport probability (p = 0.1) only slightly lowered accuracy (–0.30 %) while boosting macro-F1 by 2.55 points, striking the best balance between overall correctness and even performance across all classes. A larger teleport probability (p = 0.2) over-diluted the structural signal, degrading both metrics.

Given the assignment's emphasis on both accuracy and F1, we choose p = 0.1, achieving a test accuracy of 74.69 % and macro-averaged F1 of 74.06 %.