ASSIGNMENT 1

Question.

Create a random, unweighted, undirected, simple (no self-loop and parallel edge) connected graph G(V, E), where |V| = 1000. The edge (u, v) exists in the graph or not -- decide randomly, but the graph should be
connected, i.e., |E|>=999.

Use the following measures to understand the centrality (c_v) of a vertex(v):

(i) Normalized degree centrality of a node.
(ii) Closeness Centrality of a node.
(iii) Betweenness centrality of a node.
(iv) Eigen values and Eigen vectors.

- Compute c_v for all vertices in the graph using a combined formula made from (i), (ii), and (iii).
- Print the top 10 vertices in terms of the c_v scores.
- Also fine tune the combined formula (c_v).

Solution

Approach:

1. Creating a graph: We use a library networkX to make our desired graph (nodes = 1000 and edges >= 999).

2. Centrality measures: We know the definition of the following terms as,

Normalized degree centrality of a node: The ratio of the degree of the node with the maximum possible degree.

$$C_D(v) = \frac{\deg(v)}{N-1}$$

: The average shortest path lengths between the node and all other nodes in the graph.

$$C(v) = \sum_{u} \frac{d(v,u)}{n-1}$$

*$d(v, u)$ is the shortest path distance between node.

Betweenness centrality of a node: The ratio of the count of shortest paths that pass through the node and the total number of shortest paths in the graph. Initially, count for two nodes, s and t, and use the sum considering all s —> t shortest paths.

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

*σ_{st} is the total number of shortest paths between nodes s —> t.

*$\sigma_{st}(v)$ is the number of those shortest paths that pass through node v.

Eigen value and Eigen vector: For a given graph adjacency matrix A an eigenvalue λ and an associated eigenvector satisfy the equation:

$$Ax = \lambda x$$

Eigen vector Centrality: It assigns a centrality score x_i to each node *i* as,

$$x_i = \frac{1}{\lambda_1} \sum_{j} A_{ij} x_j$$

*x_i is the centrality score of node,

*A_{ij} is the adjacency matrix element,

*λ_1 is the largest eigenvalue of A,

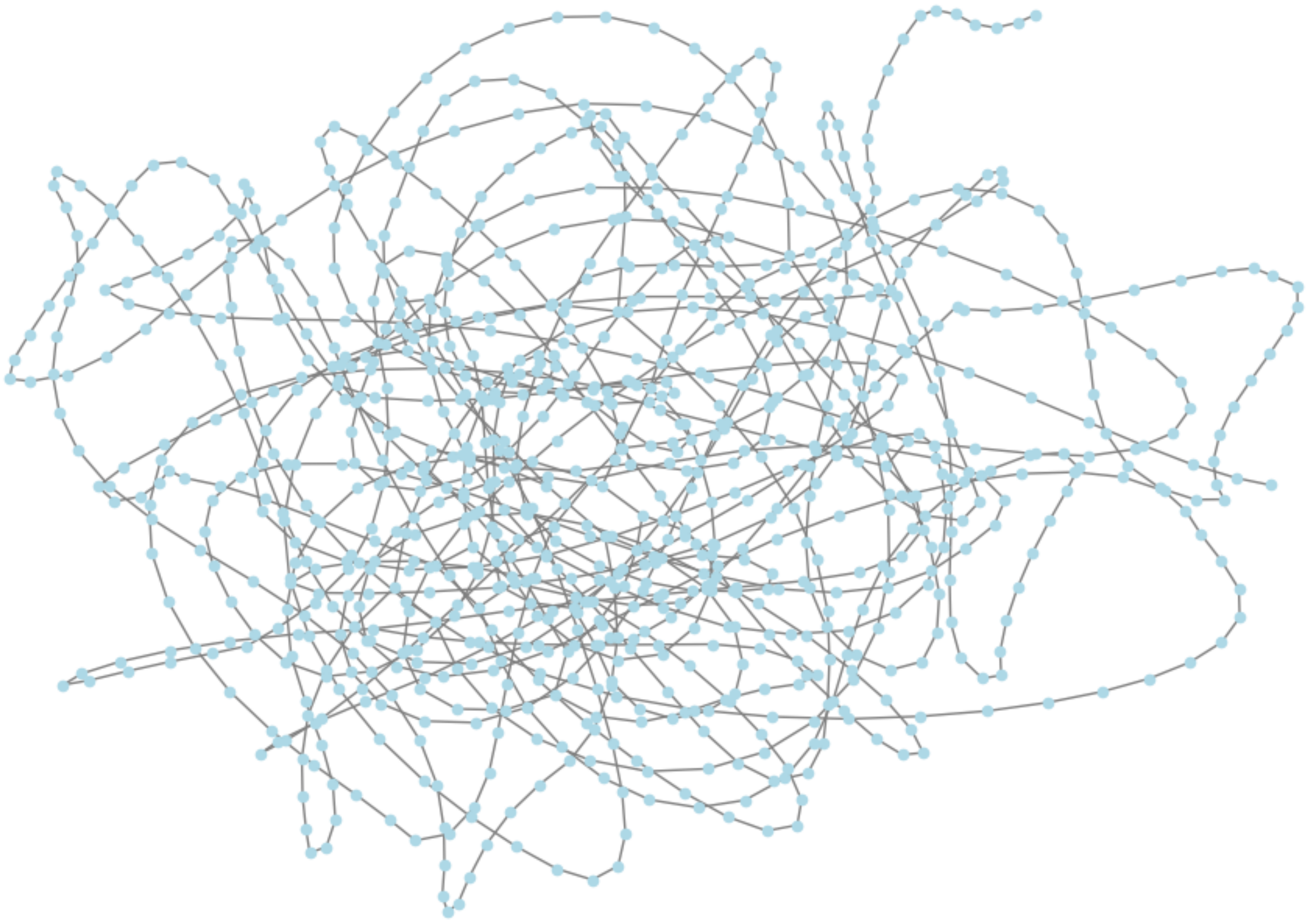
*x_j is the centrality score of neighboring nodes.

3. Fine-Tuning : A refinement process is applied to the top-ranking nodes, enhancing their scores based on second-level shortest path influence.

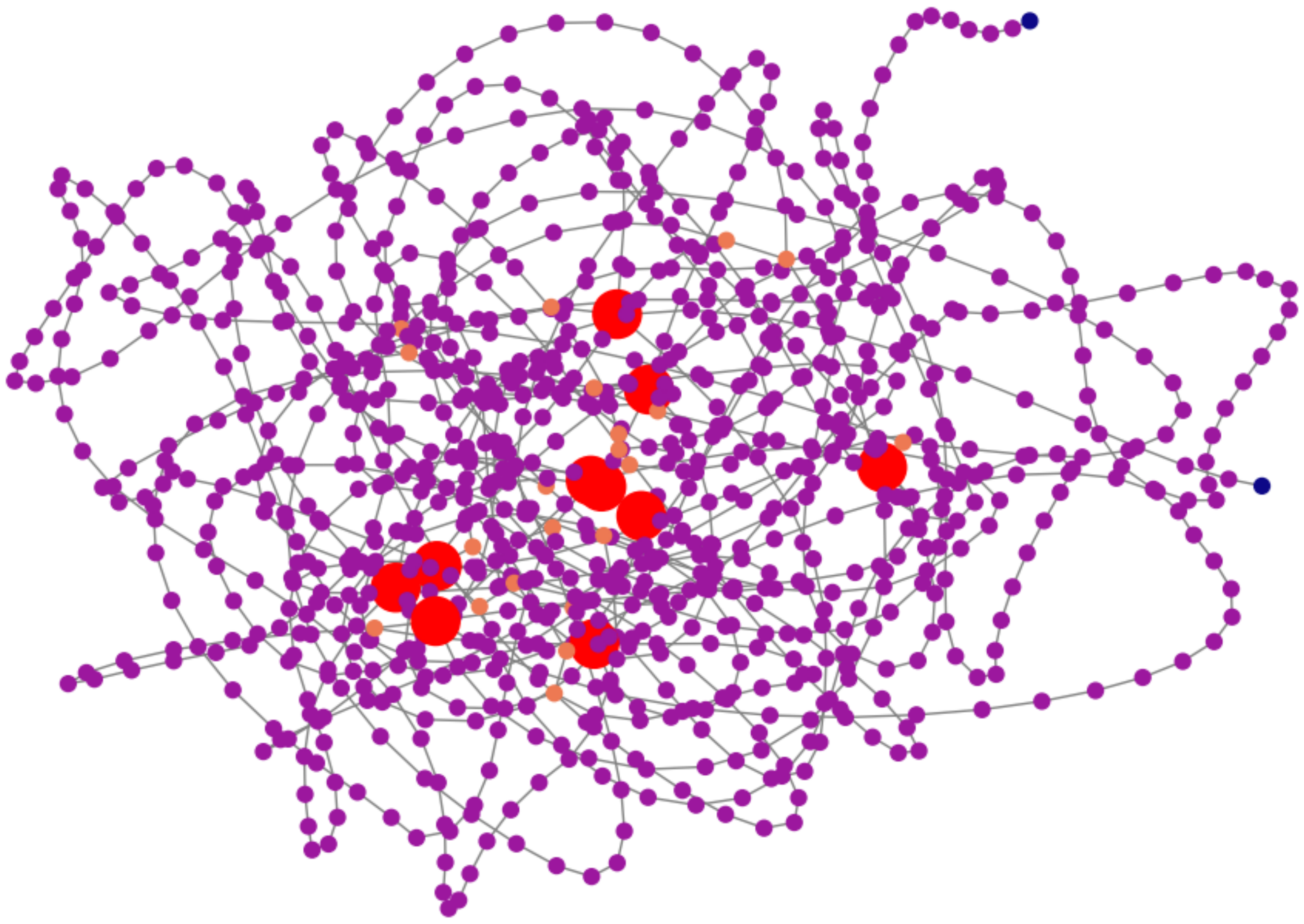
4. Printing the input and the output graph: We used the python library known as *matplotlib* to visualize the graphs.

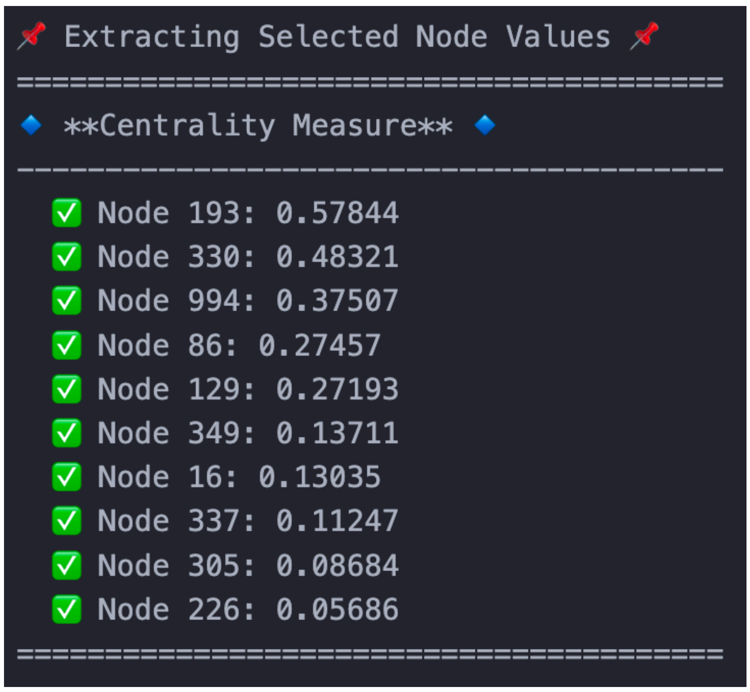
INPUT GRAPH

Nodes = 1000
Edges = 1015



OUTPUT GRAPH - I

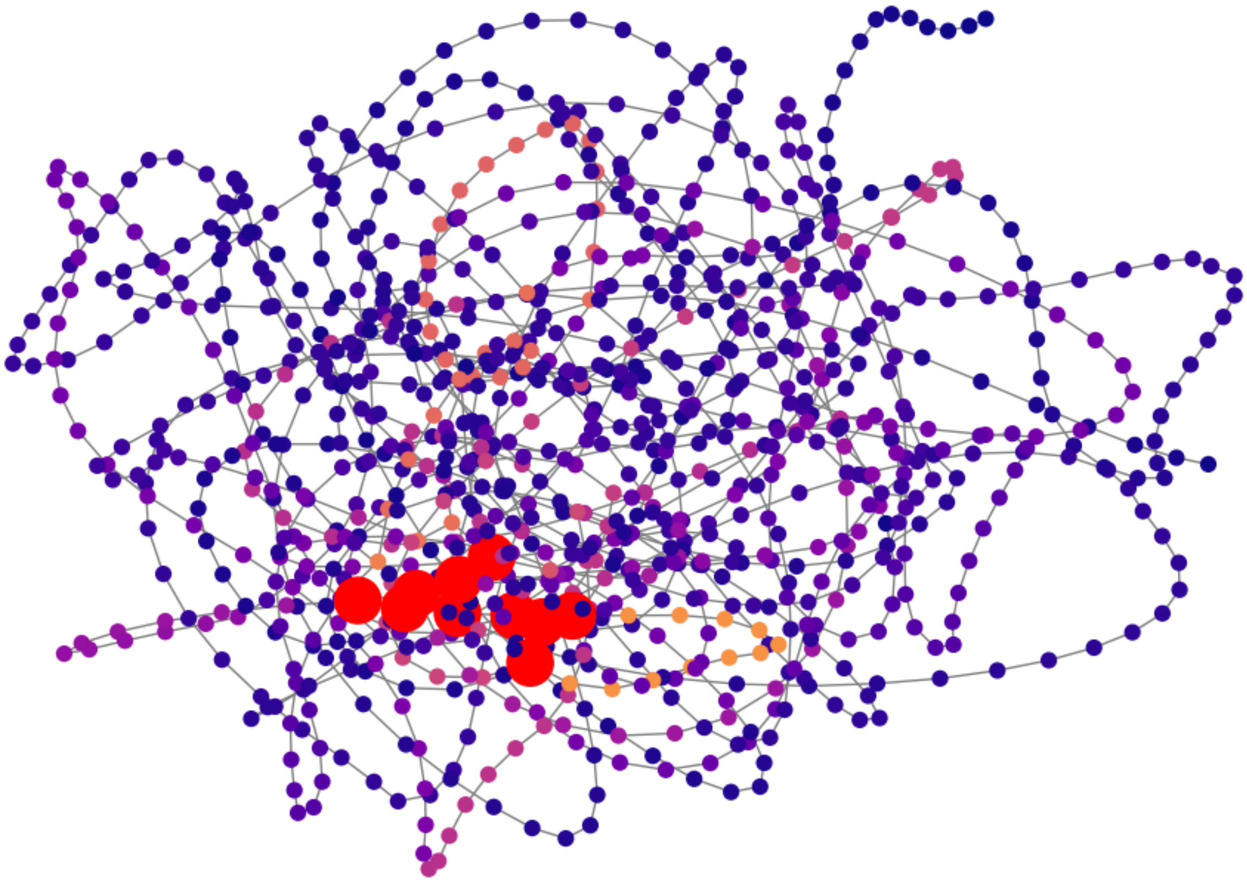


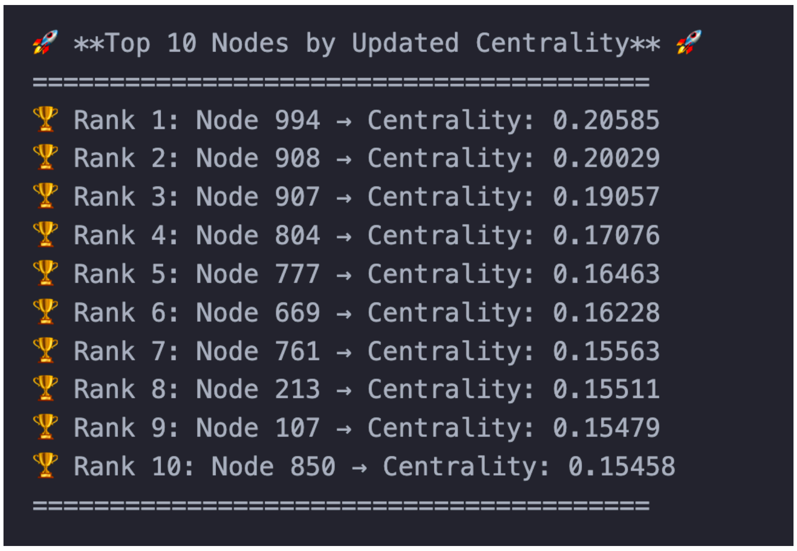


```
📌 Extracting Selected Node Values 📌
========================================
🔹 **Centrality Measure** 🔹
----------------------------------------
  ✅ Node 193: 0.57844
  ✅ Node 330: 0.48321
  ✅ Node 994: 0.37507
  ✅ Node 86: 0.27457
  ✅ Node 129: 0.27193
  ✅ Node 349: 0.13711
  ✅ Node 16: 0.13035
  ✅ Node 337: 0.11247
  ✅ Node 305: 0.08684
  ✅ Node 226: 0.05686
========================================
```

This graph highlights the nodes with the most centrality without any fine tuning or I have just used raw summation of the parameters {normalized(N), closeness(C), betweenness(B), eigenvector(E)}. value used = N+C+B+E.

OUTPUT GRAPH - II

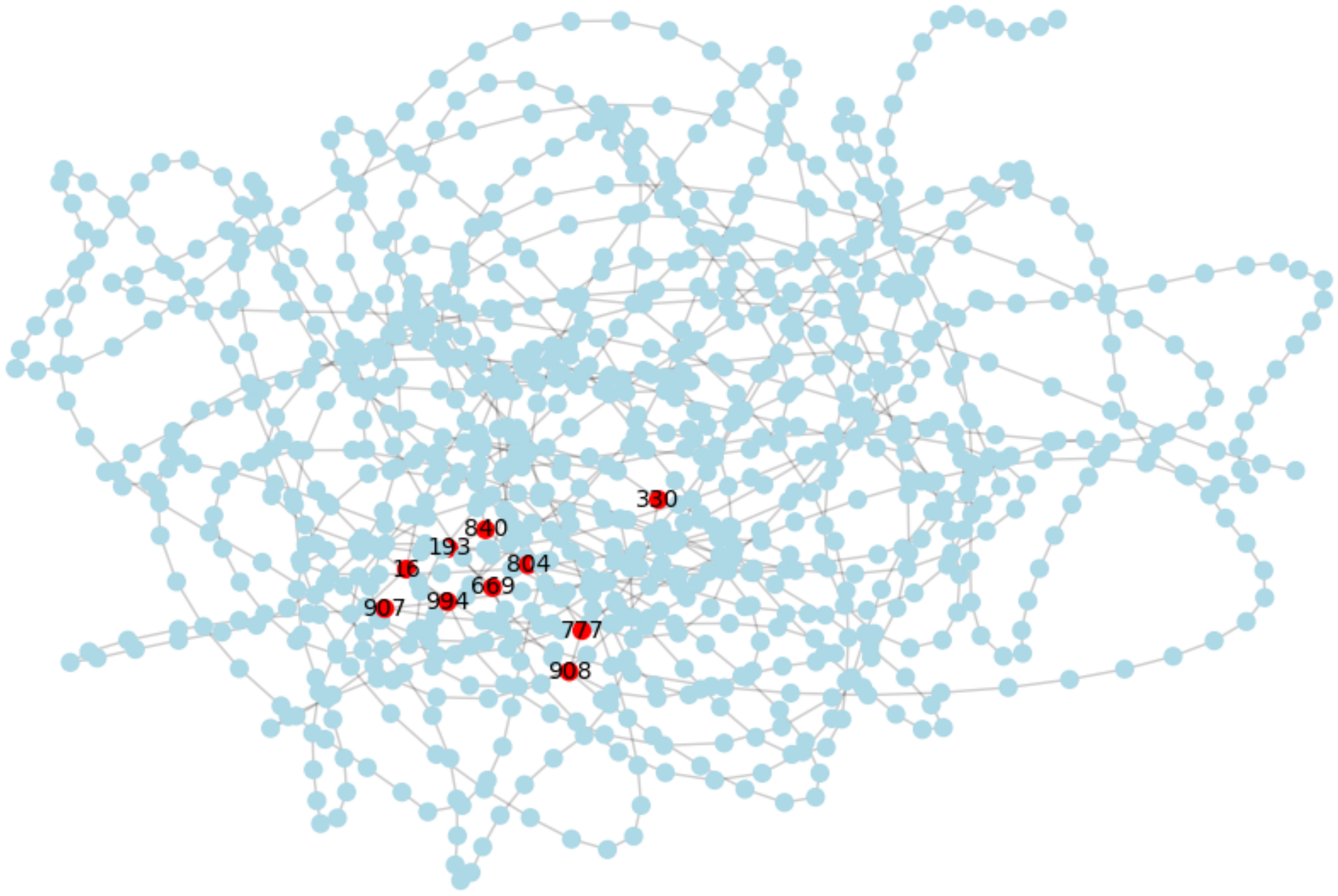


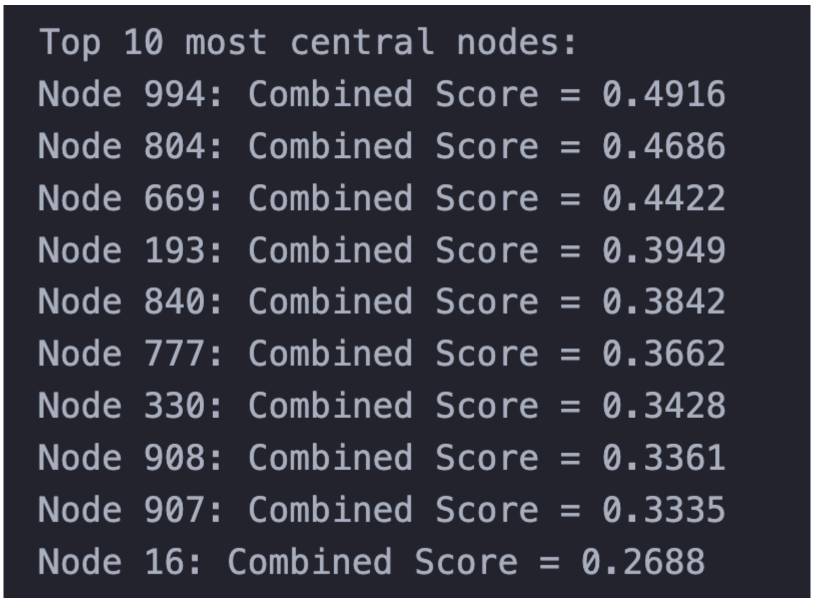


```
🚀 **Top 10 Nodes by Updated Centrality** 🚀
========================================
🏆 Rank 1: Node 994 → Centrality: 0.20585
🏆 Rank 2: Node 908 → Centrality: 0.20029
🏆 Rank 3: Node 907 → Centrality: 0.19057
🏆 Rank 4: Node 804 → Centrality: 0.17076
🏆 Rank 5: Node 777 → Centrality: 0.16463
🏆 Rank 6: Node 669 → Centrality: 0.16228
🏆 Rank 7: Node 761 → Centrality: 0.15563
🏆 Rank 8: Node 213 → Centrality: 0.15511
🏆 Rank 9: Node 107 → Centrality: 0.15479
🏆 Rank 10: Node 850 → Centrality: 0.15458
========================================
```

This graph I have used PCA (Principle component analysis without the weights normalization method to calculate the four weights, and then found out the centrality index and rank then accordingly. Weights here are as [0.29514461, 0.29210217, 0.16291511, 0.24983812].(Closeness, Betweenness, Eigenvector, Normalized)

OUTPUT GRAPH - III





```
Top 10 most central nodes:
Node 994: Combined Score = 0.4916
Node 804: Combined Score = 0.4686
Node 669: Combined Score = 0.4422
Node 193: Combined Score = 0.3949
Node 840: Combined Score = 0.3842
Node 777: Combined Score = 0.3662
Node 330: Combined Score = 0.3428
Node 908: Combined Score = 0.3361
Node 907: Combined Score = 0.3335
Node 16: Combined Score = 0.2688
```

In this graph I have used the same PCA algorithm but with normalized weights this time to find the centrality score and then used it to sort the top 10 nodes accordingly.

Weights here are as [0.289, 0.311, 0.161, 0.238](Closeness, Betweenness, Eigenvector, Normalized)

What is PCA ?

PCA is a statistical technique used for dimensionality
reduction and identifying patterns in high-dimensional data. It transforms correlated variables into a set of uncorrelated principal components (PCs) that capture the maximum variance in the data. This simplifies analysis, visualization, and modeling while retaining essential information.

Key Concepts in PCA

Principal Components (PCs): Orthogonal (uncorrelated) directions where the data varies the most.

First PC: Captures the most variance.

Second PC: Captures the next most variance (orthogonal to the first), and so on.

Eigenvalues and Eigenvectors: Eigenvectors define the direction of PCs. Eigenvalues indicate the variance explained by each PC.

Covariance Matrix:
Measures how variables vary together. PCA uses this to compute PCs.

Why PCA in our case of fine tuning ?

1. Handles Multicollinearity :-
Centrality measures (e.g. Closeness, Betweenness) are often correlated (e.g., nodes with high degree might also have high eigenvector centrality).

PCA addresses this: It creates uncorrelated principal components (PCs) by transforming the original variables into orthogonal axes.

Benefit: Avoids "double-counting" redundant information when combining metrics.

2. Data-Driven Weighting :-
Instead of arbitrary weights (e.g. equal weights in Method 1), PCA calculates weights based on variance in the data:

PCA identifies the linear combination of centrality metrics that explains the most variance in the dataset.

Weights (w, x, y, z) are derived from the eigenvectors of the covariance matrix, reflecting each metric's contribution to the overall variance.

Benefit: Weights are statistically meaningful and adapt to the specific graph structure.

3. Scale Sensitivity Mitigation :-
Centrality metrics often have different scales (e.g., Betweenness values can span orders of magnitude, while Degree is smaller).

Method 2 (PCA without normalization): Favors metrics with larger variances (e.g. Betweenness dominates).

Method 3 (PCA with normalization): Scales metrics to unit variance (measure of the amount of spread of data around the mean value), ensuring equal influence during PCA.

Benefit: Normalization in Method 3 ensures no single metric disproportionately drives the result.

4. <u>Reduces Arbitrary Assumptions:-</u>
Method 1 assumes equal importance (w=x=y=z=1), which is rarely true in real-world graphs.

PCA (Methods 2 and 3) eliminates this bias by letting the data determine the weights.

<u>Benefit</u>: More objective and adaptable to different graph types (e.g., social networks vs. infrastructure networks).

5. <u>Captures Dominant Patterns:-</u>
PCA focuses on the principal component that explains the most variance in the data (usually the first PC).

This component represents the *latent structure* shared across centrality metrics (e.g., a node's overall importance).

<u>Benefit</u>: Provides a holistic centrality score that balances multiple perspectives (degree, betweenness, etc.).

Comparison between our three approaches:

Parameter	***Linear Addition***	***PCA (Without Norm)***	***PCA (With Norm)***
Weighting Approach	Equal weights (implicit: w, x, y, z=1)	PCA-derived weights (data-driven, no scaling)	PCA-derived weights (data-driven, scaled features)
Normalization	None	None	Applied at each step
Scale Sensitivity	Highly sensitive (raw scales distort results)	Sensitive (PCA on raw data favors high-variance metrics)	Reduced sensitivity (normalization balances scales)
Multi-collinearity Handling	None (assumes independence)	Partial (PCA reduces correlated features)	Improved (PCA + normalization handles correlations)
Bias	Favors metrics with larger raw values (e.g. Betweenness)	Favors high-variance metrics (unscaled data)	Balances variance (normalization mitigates bias)
Statistical Rigor	Low (arbitrary summation	High (data-driven, but assumes raw scales are meaningful)	High (data-driven, adjusts for scale differences)
Weights (C, B, E, N)	All weights are equal as 1	0.295,0.292,0.162,0.249	0.289,0.311,0.161,0.238

OBSERVATION

We observed change in the nodes in all three cases
CASE I Linear Addition, we saw the nodes {129, 994, 193, 226, 330, 16, 305, 337, 86, 349} with values as 0.57844, 0.48321, 0.37507 and so on.
CASE II PCA, we saw the nodes {994,908,907,804,777,669,761,213,107,850} with values as 0.20585, 0.20029, 0.19057 and so on.
CASE III PCA with norm, we saw the nodes {994,804,669,193,840,777,330,908,90716} with values 0.4916,0.4686,0.4422 and so on.

The observed changes in node rankings across the three cases highlight the impact of different weighting methods. Case I (Linear Addition) shows nodes like 129 and 226, but its equal weighting likely biases results due to scale differences. Case II (PCA) introduces nodes like 908 and 907, reflecting data-driven weights, but without normalization, high-variance metrics like Betweenness may dominate. Case III (PCA with normalization) balances scales, resulting in a mix of nodes (e.g. 994, 804, 193) and more equitable weights (e.g. 0.4916, 0.4686). Normalization improves fairness, ensuring no single metric skews results. Further refinement could involve validating weights with domain knowledge or testing on diverse graphs.

CONCLUSION

- Method Sensitivity: Node rankings vary significantly across methods, underscoring the impact of weighting strategies (linear vs. PCA-driven) on centrality outcomes.
- Normalization Matters: Case III (PCA with normalization) balances metric scales, yielding hybrid rankings (e.g. 994, 193) and higher scores (0.4916 vs. 0.20585 in Case II), reducing bias from raw metric magnitudes.
- Bias in Linear Addition: Case I's equal weights likely prioritize nodes with inherently larger metric values (e.g. Betweenness), skewing results toward scale rather than structural importance.
- PCA Trade-offs: Case II (PCA without normalization) highlights high-variance metrics but risks overemphasizing scale-sensitive features (e.g. node 908), while Case III mitigates this through scaling.
- Interpretability vs. Rigor: Linear addition (Case I) is intuitive but statistically weak, whereas PCA methods (II/III) offer data-driven insights at the cost of complexity.
- Future Improvements: Hybrid approaches (e.g., domain-informed PCA or entropy-weighted normalization) could refine robustness, ensuring rankings align with network-specific importance while retaining statistical rigor.