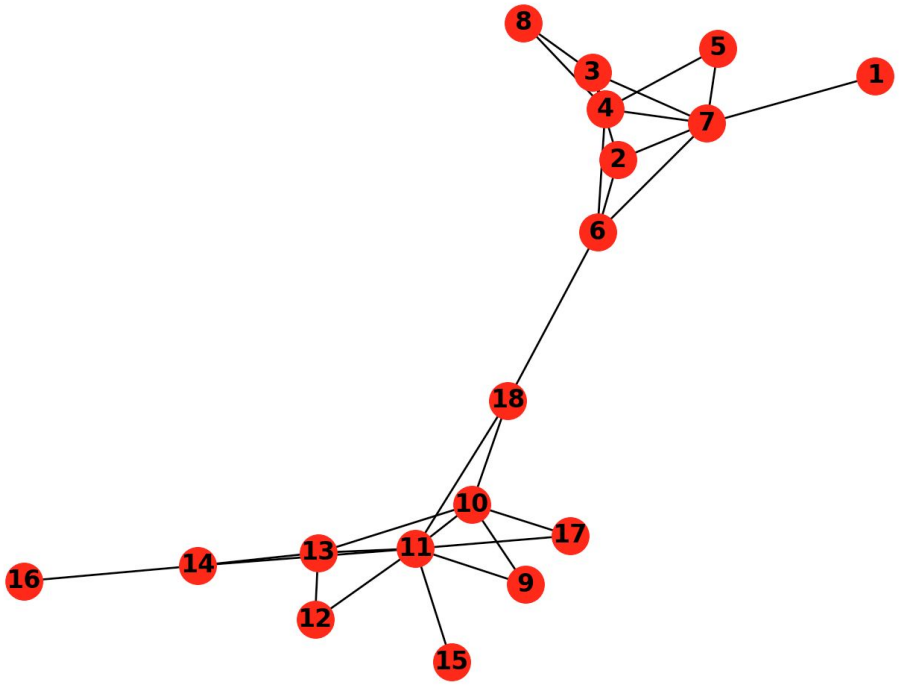
LA Assignment 2

Vishal Goel
Sr.No. - 15451

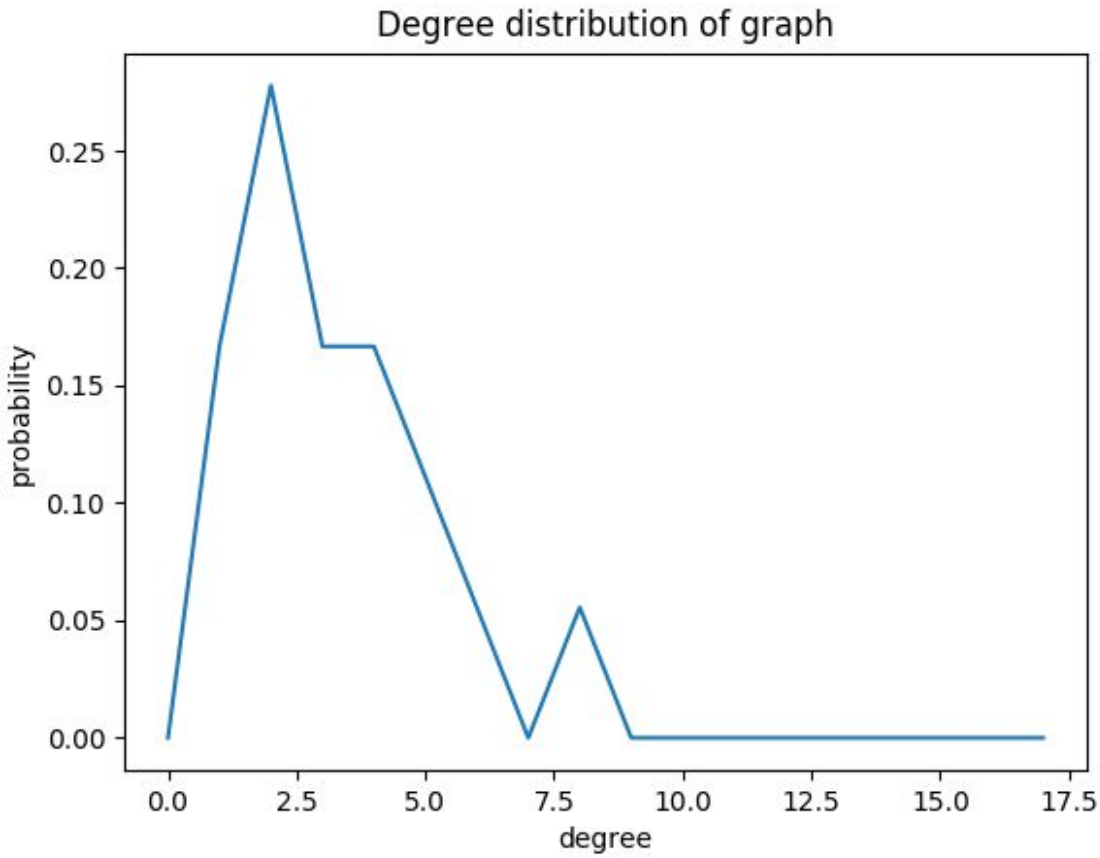
Python version 3

ANSWER 1:

The graph given as the input :



Task 1:

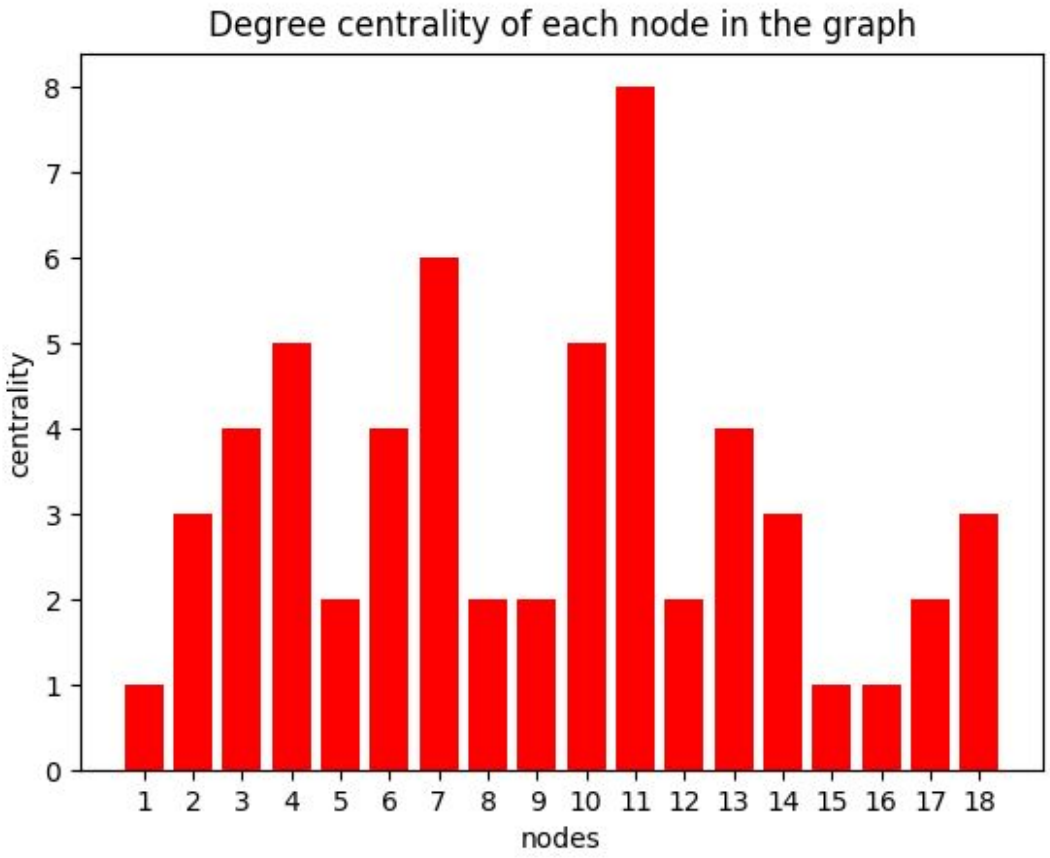


The degree distribution of the graph seems to follow Binomial distribution but since the graph has only a few vertices, it may become a Poisson distribution when graph becomes large and the probability of edges between any two vertices becomes less. This is because most of the edges are among different clusters (Houses) and not between the clusters. And thus, the probability of having edges among any two pair of vertices will reduce as graph gets large.

Task 2 :

Node centrality measure picked : Degree centrality

The top two central nodes were found to be node 11 and node 7



Task 3 :

Edge centrality measure picked : Edge Betweenness (defined for an edge as the number of shortest paths between any pair of vertices in the graph that pass through that edge). The most central edge in the graph was found to be the edge that connects the nodes 6 and 18.

Task 4 :

The eigenvalues and eigenvectors of the Laplacian matrix were computed using the Faddeev Leverrier method to compute the coefficients of the matrix, followed by numpy to compute the roots (eigenvalues) of the characteristic equation and then followed by the Inverse Power Eigenvalue algorithm to finally compute the eigenvectors. The eigenvalues(given below) turned out to be real, as expected because the normalised Laplacian matrix is symmetric.

```
[ 1.73675441e+00  1.70229138e+00  1.60994361e+00  1.52870407e+00
  1.45962818e+00  1.39629626e+00  1.24802361e+00  1.19926841e+00
  1.00824005e+00  9.91641675e-01  8.27687874e-01  8.08429186e-01
  7.40138581e-01  6.84443980e-01  6.11293015e-01  4.07823929e-01
  3.93917822e-02 -2.60135108e-10]
```

Task 5:

The two smallest eigenvalues are

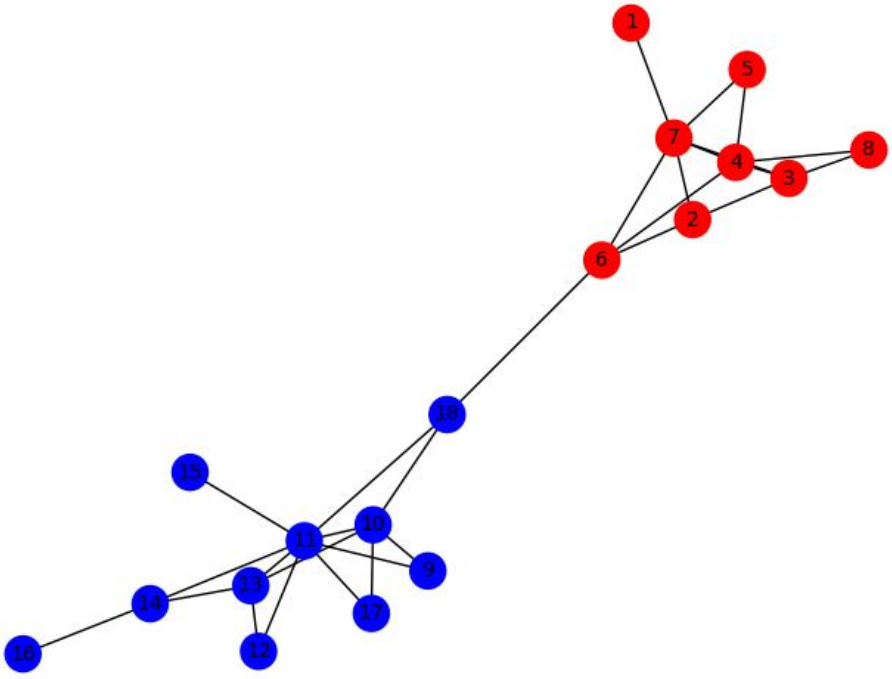
1. E1 = -2.6013510752008686e-10
2. E2 = 0.03939178217880512

Their corresponding eigenvectors are as follows:

1. EV1 = [0.13, 0.23, 0.26, 0.29, 0.19, 0.26, 0.32, 0.19, 0.19, 0.29, 0.37, 0.19, 0.26, 0.23, 0.13, 0.13, 0.19, 0.23]
2. EV2 = [0.15, 0.24, 0.3, 0.32, 0.21, 0.2, 0.35, 0.22, -0.18, -0.26, -0.35, -0.19, -0.27, -0.24, -0.13, -0.15, -0.18, -0.09]

Error in smallest eigenvalue (|E1-0|): 2.6013510752008686e-10
Since all values in eigenvector corresponding to the smallest eigenvalue should be equal, the error (calculated by taking average of the difference of each value from the mean) : 5.0%

Task 6 :



Using the eigenvector corresponding to the second smallest eigenvalue, the nodes of the graph were coloured red for positive valued nodes and blue for negative valued nodes. It was observed that each house got perfectly coloured with the same colour. Thus, this is a good technique to make two clusters. We can also try k-means clustering as a different technique to make the clusters.

Task 7:

The variability of each cluster was calculated as the variance of the values got in the eigenvector corresponding to the second smallest eigenvector. The variance of the cluster with blue colour turned out to be more than that of the red colour and thus, the House of Algebrician should befriend the house with the blue colour.

Bonus 1:

The clusters formed by numpy are

1. [0, 1, 2, 3, 4, 5, 6, 7]
2. [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]

The clusters are exactly the same as we got!!

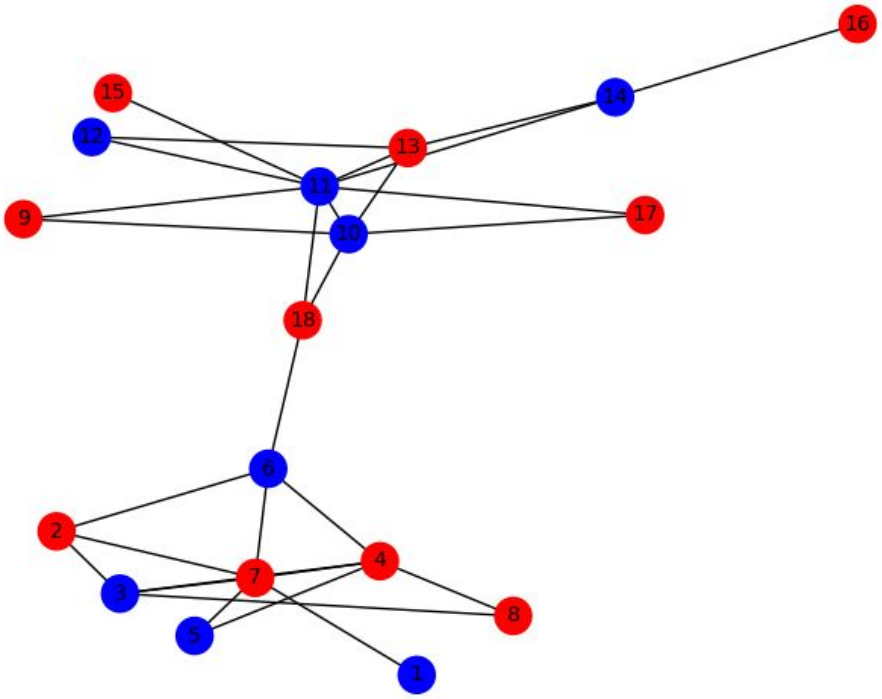
Bonus 2:

To make more than two clusters, we could use the following ideas:

1. After getting the two clusters, we could apply k-means clustering to make more clusters.
2. After getting the two clusters, we could pick the cluster with more variance and then limiting our graph to only this cluster, we could run our current algorithm to divide this into two clusters. We can repeat the algorithm on the three clusters and find more clusters similarly. The correctness of the idea is obviously questionable but it could be tried out.
3. We could also pick “central” edges in the graph to look for clusters. We can delete the edges with the highest edge centrality and the connected components obtained can be treated as clusters. But this approach is limited to getting random number of clusters instead of a fixed number of clusters.

Bonus 3:

The graph obtained after taking eigenvector corresponding to the second largest eigenvalue is:



Bonus 4:
The central edge in the graph (edge connecting nodes 6 and 18) turned out to be the one that connects the two clusters. And thus, it validates the fact that this edge is the most “important”.

Answer 2:

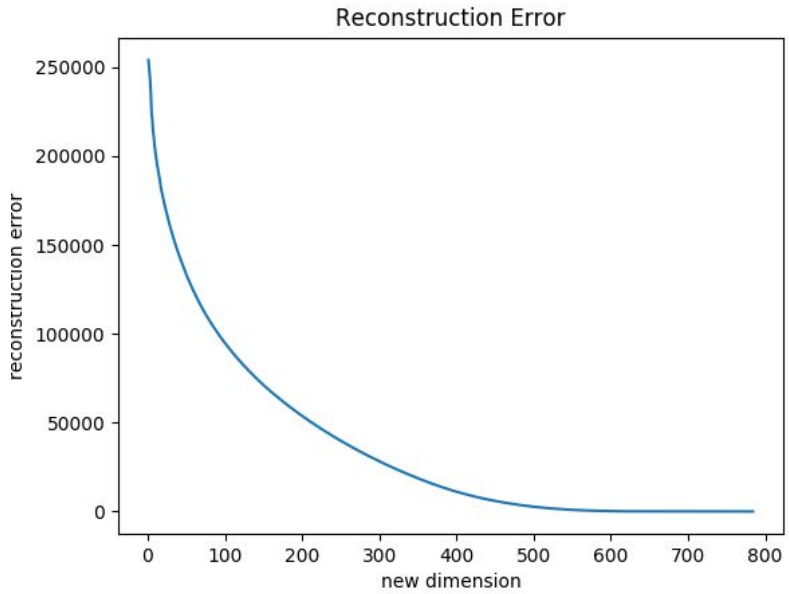
Task 1:
The mean and the covariance of the given matrix were found and stored as mean_vector.txt and covariance_matrix.txt

Task 2:
The eigenvalues and the eigenvectors of the matrix were obtained. Only one eigenvalue 0 was found to be repeating with multiplicity = 116. Also, all eigenvectors were found to be orthogonal as checked by taking dot product of every pair of eigenvectors that turned out to be approximately 0. Eigenvalues and eigenvectors have been stored in eigenvalues.txt and eigenvectors.txt

Task 3:
Gram-Schmidt Orthogonalisation implemented.

Task 4:
Dimensionality Reduction code implemented.

Task 5:

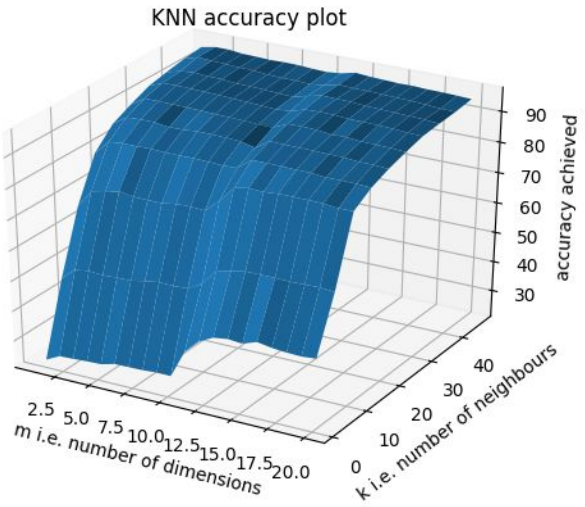


The reconstruction error plot was calculated by taking absolute distance between the original matrix and the reduced matrix (with padded 0s for the rest of the dimensions) in the same coordinate system. The plot validates the fact that the error should decrease as the number of dimensions increase.

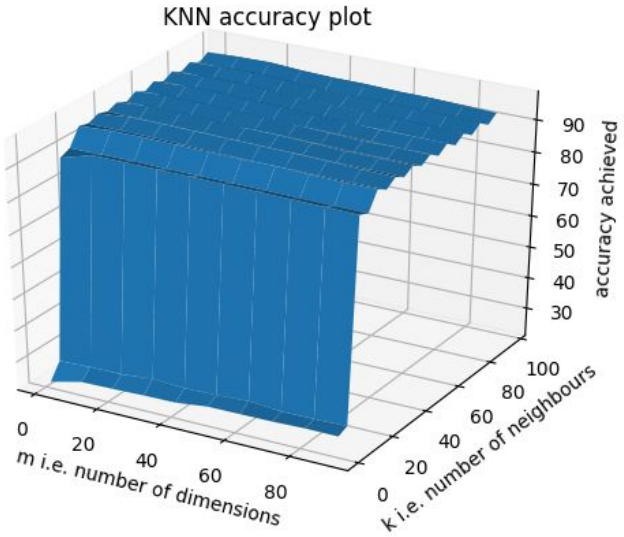
Task 6:
m = number of dimensions
k = parameter in k-means clustering

The following plots were plotted, at different granularities w.r.t. M.

1. m < 20 : As the value of m and k increases, the accuracy also increases. There is a sudden increase in accuracy around m=10 as well for smaller values of k.



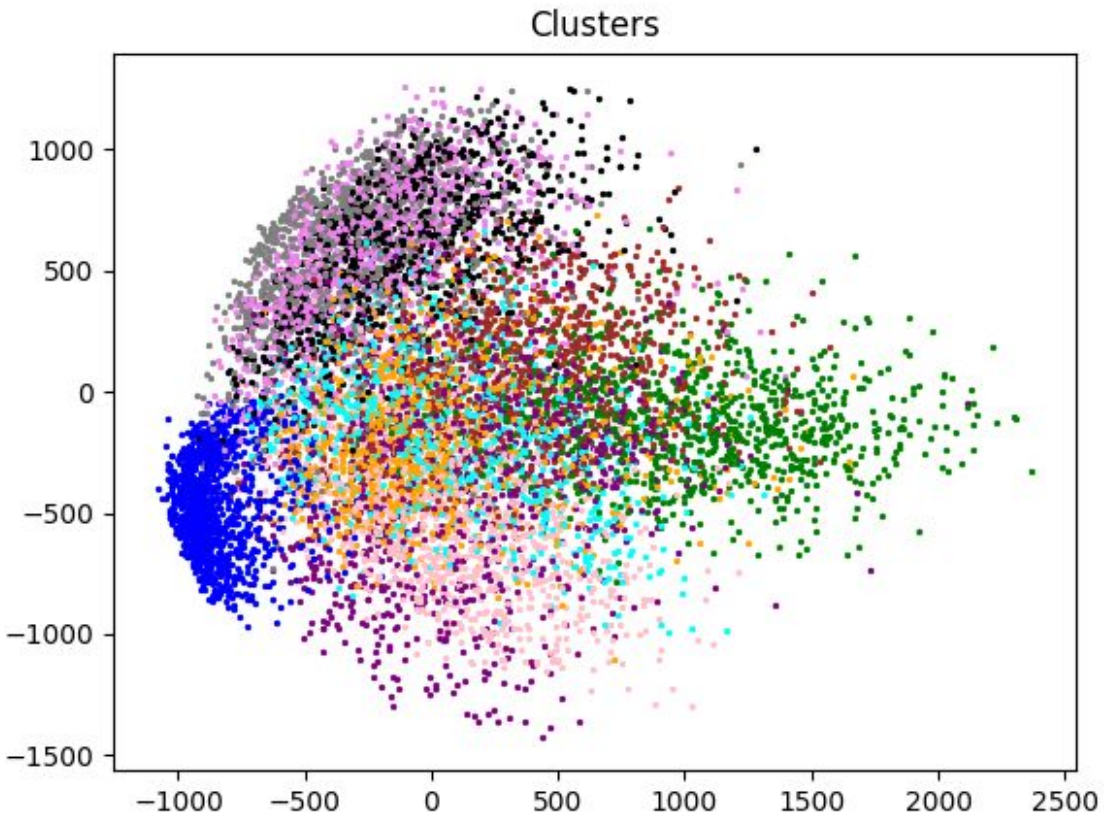
2. m < 80 : For a given dimension, it is observed that as k increases initially from around 5-6, the accuracy starts at peak and then decreases steadily and then there are gradual ups and downs in the accuracy. But the peak is achieved somewhere between 0 to 10 for k.



Finally, it was observed that the peak accuracy was achieved at m=51 and k=6

Bonus 1:

The data was plotted on a 2D surface and it was observed that instances belonging to the same label class clustered together.



Bonus 2:
The accuracy achieved on the validation set using my implementation = 97.1%
The accuracy achieved on the validation set using sklearn KNN = 96.9%

The accuracy achieved on the test set using my implementation = 93.8%
The accuracy achieved on the validation set using sklearn KNN = 92.5%