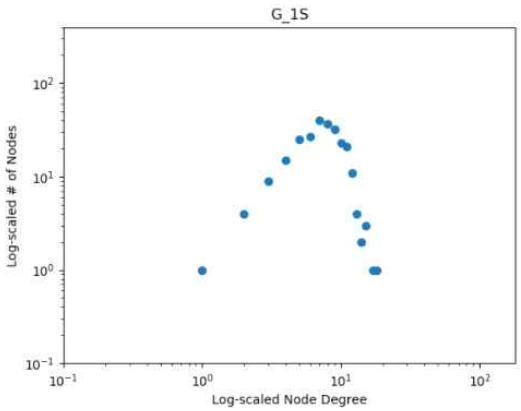
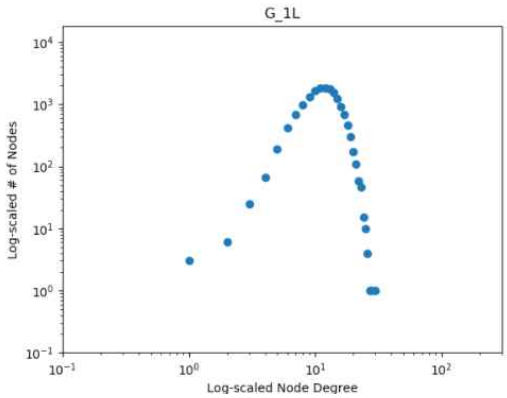
Graph Mining Assignment1

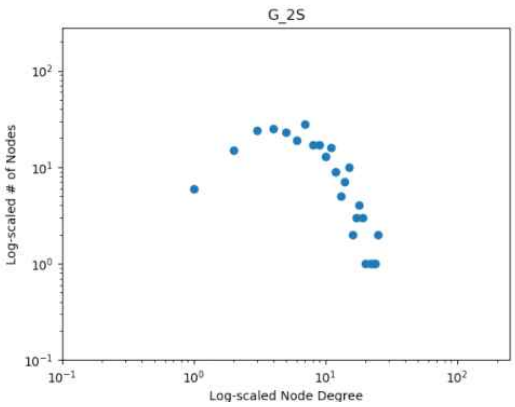
Name: Seungwoo, Ryu
ID: 20213207

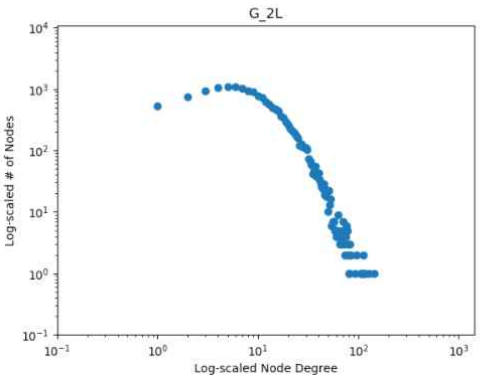
1.

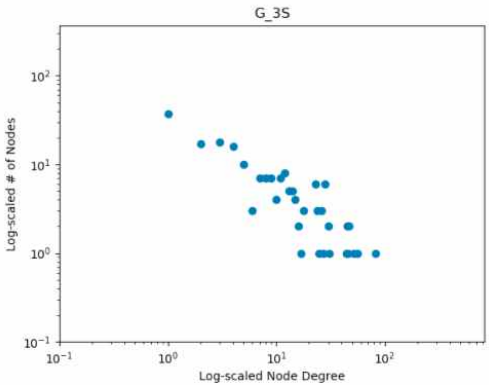
First of all, it is possible to distinguish 6 separate cases by the title name in the plot. For example, G_1L means that this graph is the case for (S1) with Large graphs. Figures below are the plots for **InDegree** distribution in a log-log scale gotten from **analyzeInDegrees** function.

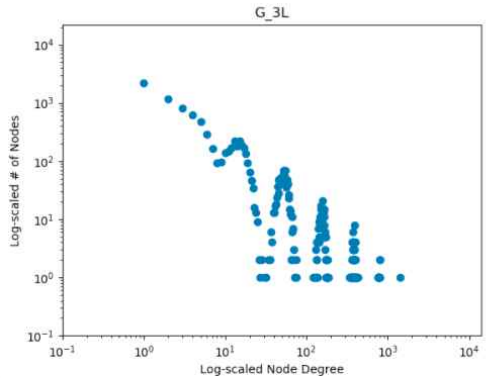




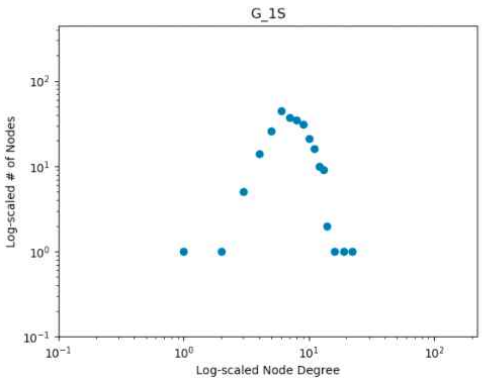


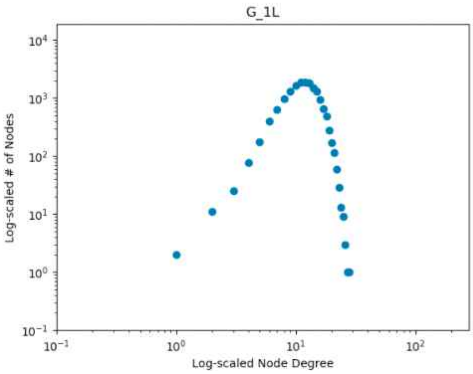


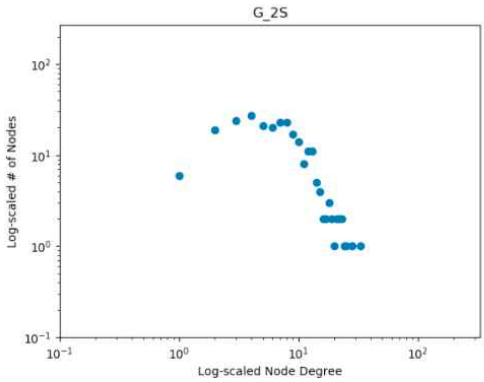


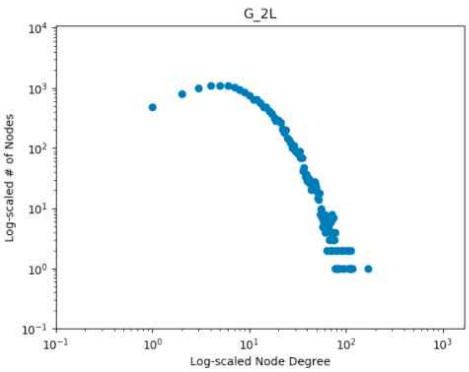


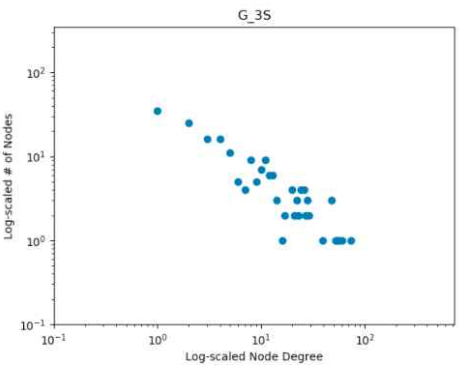
And then, the plots below are **OutDegree** distribution in a log-log scale gotten from **analyzieOutDegrees** function.

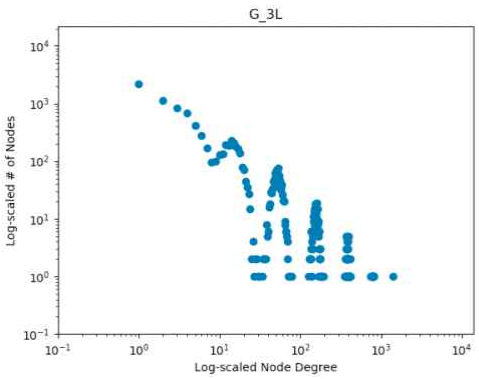






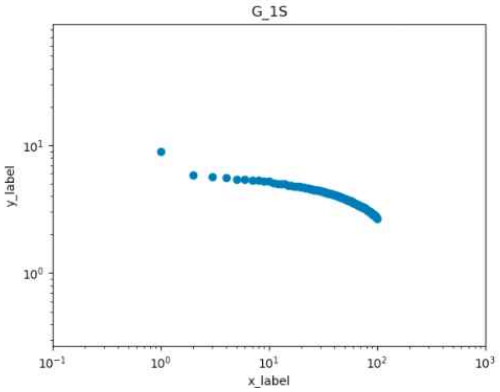


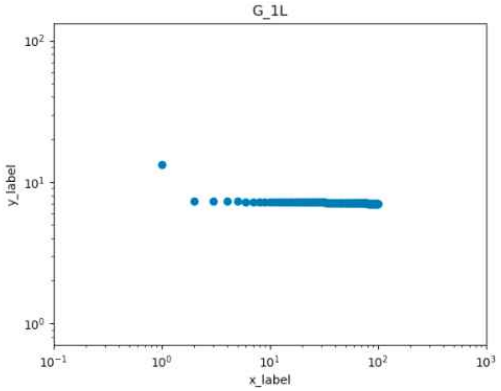


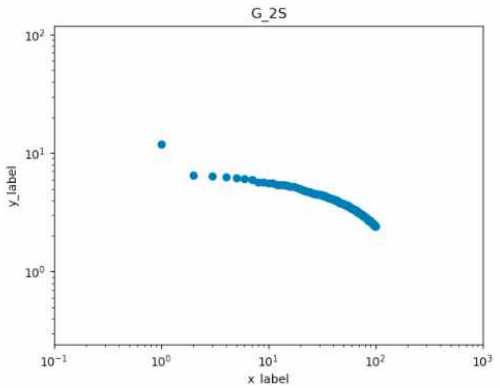


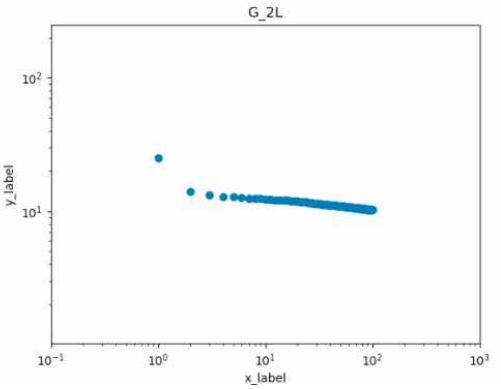
2.

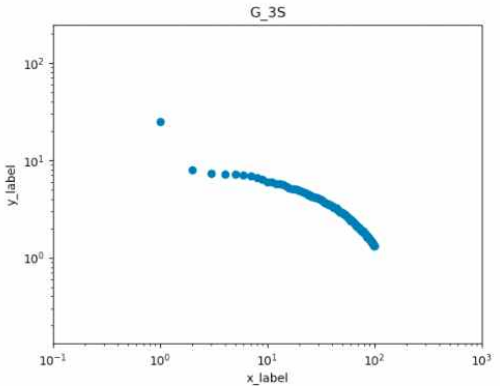
6 plots below are the plots in a log-log scale gotten from **getSingularValues** function.

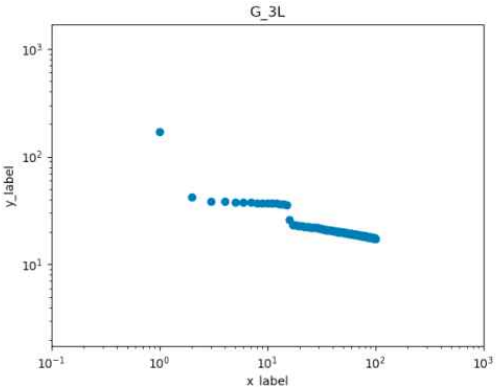












3.

[About Degree-Distribution]

For all cases, there's no remarkable difference between **Indegree-Distribution** and **Outdegree-Distribution** comparing each 6-pair. However, there's some noticeable difference according to the probability which is assigned to each block: A, B, C, and D.

First of all, when all 4 blocks are assigned with equal probabilitiy, i.e. 0.25, it showed 'bell-shaped' curve in a log-log plot. It means that this case cannot reflect on the property of real graph. It rather follows the behavior of random graph. However, ***as the probability of edges being assigned to A block increases (from 0.25, 0.4 to 0.7),*** it gradually resembles the behavior of real graphs which shows linear line with negative slope.

However, one exceptional case happens in G_3L. It shows somewhat bouncing trend rather than a linear line. It might be based on the sparsity of G_3L graph. For easy explanation, I am going to compare this case with G_3S. In case of G_3S, it takes 256 nodes with 2000 edges with probability 0.7, 0.1, 0.1, and 0.1 for each block. On the other hand, in case of G_3L, it takes 16384 nodes with 200000 edges with probability 0.7, 0.1, 0.1, 0.1.

Looking at the G_3S, the ratio of existing edges in adjacency matrix is about 3% (2000/(256*256)). On the other hand, the ratio of G_3L is about 0.07% (200000/(16384x16384). G_3L has much more possibilities to have various node degrees (16384 >> 256), but much sparser than G_3S. In summary, it is impossible for G_3L to distribute node degrees proportionally. There might be some node degrees which do not appear often.

[About Singular Values]

First of all, singular value decomposition can decompose a matrix into three components. In this case, the matrix object of SVD is 'adjacency matrix'. It might be helpful to save sparse matrix in a decomposed format rather than saving all the not-connected 0 stuffs. Saving in a decomposed form can save a lot of memories without much information loss.

As the probability for block A (p_A) increases (from 0.25, 0.4 to 0.7), singular value corresponding to rank 1 gradually increases. This increase is more remarkable in Large graphs (sparse matrix) which actually need efficient way to save connection information.

It means that any (sparse) matrix can be decomposed by SVD, and as the probability for block A increases, the eigenvector corresponding to rank1 actually plays an more important role at explaining the adjacency matrix. Even further, the importance of eigenvector axis corresponding to rank1 actually increases as p_A increases.

[Overall Interpretation]

A real graph can be created using various models. In this assignment, we imitates the real graph by using R-MAT assigning bigger probability to block A. An adjacency matrix of realistic graph (G_3S in this case) can be saved efficiently using Singular Value Decomposition, and there exists major eigenvector which includes much information about edge connection.