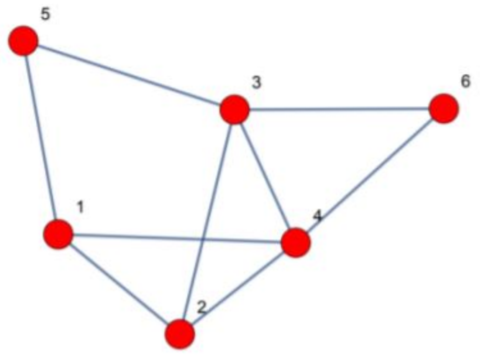
Networks Problem Set

CS166

December 5, 2020

1 Problem 1



Adjacency matrix

To find the adjacency matrix, we use each node in a sorted manner (1 to 6) as a column label, same as a row label. Then, we check each connection (edges) between the given nodes in row and column. For instance, the first item in the rows is *node 1*, and the first item in the column items is also *node 1*. It is the same node and there is no edge connection, thus the first item in the matrix is 0. However, *node 1* is connected to *node 2*, thus that connection is shown at the intersection between *node 1* and *node 2* in the matrix, as 1. Since this is an undirected graph, the connection could be noted as two since we assume that the edge is bidirectional, but in our case we use 1 to represent the edge that links two nodes and that is noted in two places in the matrix; position [1,2] and [2,1]. The corresponding adjacency matrix is:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Adjacency list

On the other hand, the adjacency list represents the same graph in another form; by taking the notes and linking the list of nodes they are connected to. For example, *node 2* is connected to *node 1, node 3,* and *node 4*. Thus, *node 2*'s linked list would contain the above mentioned node. The adjacency list for our example is:

$1 \rightarrow \{2, 4, 5\}$
$2 \rightarrow \{1, 3, 4\}$
$3 \rightarrow \{2, 4, 5, 6\}$
$4 \rightarrow \{1, 2, 3, 6\}$
$5 \rightarrow \{1, 3\}$
$6 \rightarrow \{3, 4\}$

Node Degree

The node degree represents the number of edges a node is connected to. In our case, the edges are assumed to be bidirectional, but we will count them as 1.
deg (1) = 3
deg (2) = 3
deg (3) = 4
deg (4) = 4
deg (5) = 2
deg (6) = 2

Path, Trail, Cycle, or other?

Based on the definition of the trail, we call *trail* a walk where vertices can be repeated but not the edges. None of the following examples is a trail. A *path* is a walk which has no repeated vertices or edges.In our case, the second example is a path. A *cycle* is a walk which is closed and it ends at the same vertex where it begins. The third walk exemplifies it.

$6 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 2 \rightarrow 1$: Other; we could simply call it a walk because it has repeated edges, thus repeated vertices, too. It could be a trail, only if we consider the bidirectionality of the edge.
$1 \rightarrow 4 \rightarrow 6 \rightarrow 3 \rightarrow 2$: Path; because neither vertices, nor edges are repeated.
$5 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5$: Cycle; because it starts and begins at node 5, making a closed walk.

Triangles

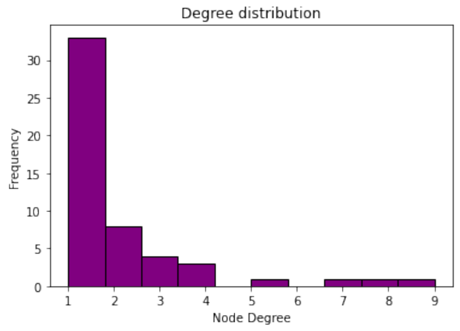
A connected subgraph of three-connected nodes represents a triangle. In our network, we see three such subgraphs; $\{1 \rightarrow 4 \rightarrow 2\}, \{2 \rightarrow 3 \rightarrow 4\}, \{3 \rightarrow 4 \rightarrow 6\}$.

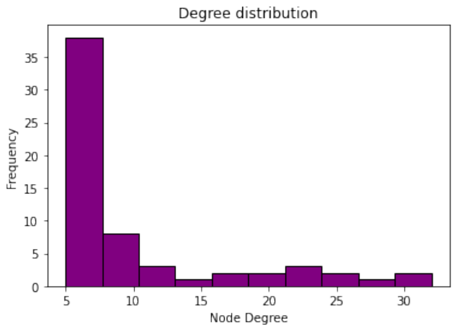
2 Problem 2

Barabási-Albert random graph algorithm generates random scale-free networks. Its algorithm has a preferential attachment function, and goes as follows:

1. Start with a given number of nodes and edges
2. Add one node at a time with probability of connecting to the most popular nodes.
3. Connect to less nodes than the total number of nodes

After running the algorithm 3000 times, with initial number of nodes m=2, and number of edges, n=1 (left), and m=12; n=5 (right), we get the following degree distributions.

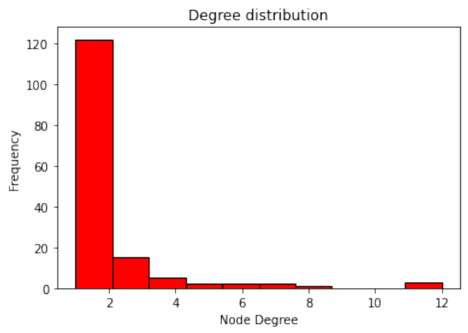


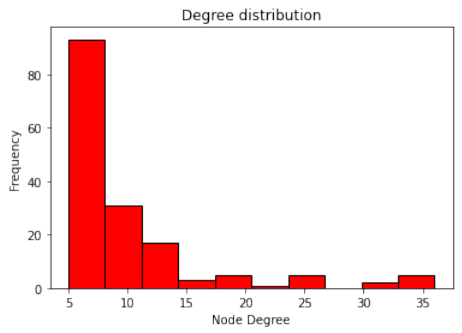


We could notice a trend between these two histograms. With the increase of number of nodes and edges, the range of the node degree expanded, but however it still follows the same distribution. This distribution is also known as power-law distribution.
We are required to modify the preferential attachment aspect of the algorithm, to a new one:

1. Start with a given number of nodes and edges
2. Pick a random number from 0 to 1.
   (a) If the random number is less than 0.5, then add a node according to preferential attachment
   (b) If the random number is bigger than 0.5, then attach the new node to a uniform-randomly selected node.
3. Connect to less nodes than the total number of nodes

Once we run the modified algorithm 3000 times, and with m=2, and n=1 (left), and m=12; n=5 (right) and, we get the following degree distributions:





A difference between the original and the modified algorithm stands at the node degree. Although the distribution remains the same, the node degree increases slightly when we run the modified algorithm. The connection between the nodes and edges is maintained, although there is 50% chance for the preferential attachment not to be applied. Such connection enables us to observe power-law distribution.

3 Problem 3

Highest Diameter Value

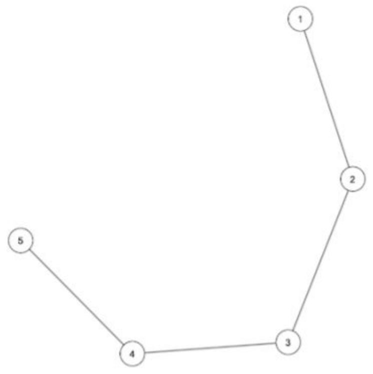
The diameter of the graph is the longest path out of all the existing ones. To see which graph has the longest path between two most distant nodes, we need to see how many edges connect the two furthest nodes for each graph.

Graph 1: The distance between any two nodes is 1 edges.

Graph 2: The distance between any two nodes in different clusters is 3 edges, because each of the nodes in any cluster are connected to one of the central nodes (1, 10, 20, 30) and these nodes are also connected to one another.

Graph 3: The distance between 1 to 5 is 4 edges.

Out of all the three graphs, Graph 3 has the biggest diameter.



Highest Density Value

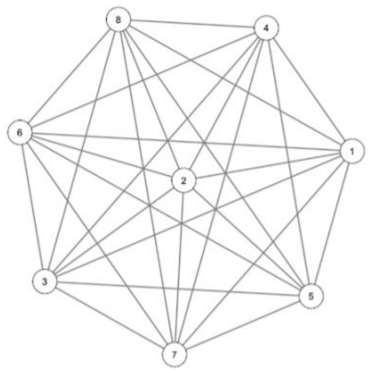
The density of a graph is measured by the number of existing edges over the number of all possible existing edges. The number of all the existing connections is found by the formula $a = \frac{n(n-1)}{2}$, where n represents the number of nodes, and a represents the number of all possible connections. To find the density, we simply divide the number of edges, e, over a; $d = \frac{e}{a}$.
In our case, for graph one, it is visible that each node connects with every other node. This could also be explained as *all of your friends are friends with one another.* In that case, our density should be 1. To prove this numerically, we can use the above formula.

$$n = 8; e = 28$$

$$a = \frac{8(8-1)}{2} = \frac{56}{2} = 28$$

$$d = \frac{e}{a} = \frac{28}{28} = 1$$

A density of 1, is also the maximum density that can be achieved, thus we could say that graph 1 has the highest density among three, or at least has equal density to another graph. However, we already know that it takes 4 edges to connect the first and the last node in graph 3, which makes its density less than 1, and we can observe that some nodes in graph 2 are not directly connected to one-another, also resulting in its density being less than 1. **Therefore, we could confidently say that Graph 1 would have the highest density.** Visually, we can see that in the first graph each node is connected to every node in the network.



Average clustering coefficient

Average clustering coefficient informs us, how likely it is that two connected nodes in the graph are part of a cluster. We can calculate it by finding the clustering coefficient (cc) for each node in the graph, and then finding the average. To do so, we need to know the degree of the node, d, and the connections(links) between its neighbours, c. The formula takes the shape of $cc = \frac{2 \cdot c}{d(d-1)}$.

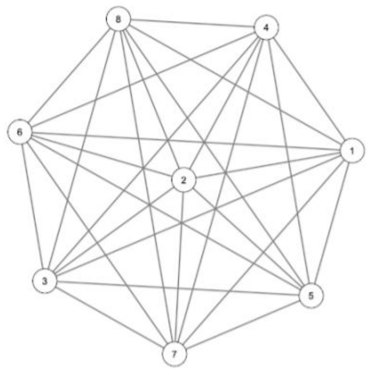
For **Graph 1**, if we find the clustering coefficient for one node, it will be representative of the whole graph, as each node has the same degree and their neighbours have the same number of connections. For instance, node 1, has a degree of 7, and each of its neighbours are linked with one another (28-7=21). That would result in $\frac{2 \cdot 21}{7(7-1)} = \frac{42}{42} = 1$. This also represents the average clustering coefficient.

For **Graph 2**, the results cannot be analyzed as easily analytically; as each node in the one of the three visible clusters has the same clustering coefficient, but nodes 1, 10, 20 and 30 have different coefficients. We can break this graph into the three visible clusters which appear to be fully connected. Thus, every node except 1, 10, 20, 30 has a coefficient of 1 (it is the same case as with Graph 1). We can observe that 1 and 30, and 10 and 20 have the same coefficients, so for the sake of simplicity we assume, that all of them have the same coefficients with one another.[1] Then, we calculate one of those, for instance, node 1:
The degree of 1 is 12 (with nodes 2-9, and node 10, 20, 30); and the links between their neighbours are $\frac{(9(9-1)}{2} + 3 = 39$.
Now, we can find the clustering coefficient for node 1: $cc = \frac{2 \cdot 39}{12(12-1)} = \frac{78}{132} = 0.59$.
The average clustering coefficient, is $\frac{(26 \cdot 1)+4 \cdot 0.59}{30} = \frac{28.36}{30} = 0.94$
These results make sense, as there are three clusters visually, thus there are at least 3-4 nodes that have to connect the clusters and have not as high of a clustering coefficient as the others (i.e 1,10,20 and 30), because not all of their neighbours are connected to one another.

For **Graph 3**, the clustering coefficient is 0, as there is only one neighbour, thus the number of links between neighbours is 0. The whole formula takes the shape of $\frac{0}{d(d-1)}$, which is equal to 0. Visually such results make sense as there are no clusters at all.
Thus, Graph 1 has the highest clustering coefficient. For instance, if we pick any two nodes from Graph 1, we will find that they are part of a cluster, but in Graph 2, if we picked 10 and 20, then we'd come to the conclusion that

[1] All these coefficients will be under 1, and approximately the same. In addition, without this assumption, the calculation of the average clustering coefficient becomes very complicated.

they are not part of the same cluster.



4 Problem 4

As of now we calculate the PageRank of a page as a sum of the PageRanks of all nodes that are linked with the page with incoming edges, over the number of outgoing edges. Moreover, it has integrated the damping factor which is the probability that "the random surfer" will continue clicking on links. A Silly PageRank would only consider the number of incoming edges to a node, as its PageRank value. In other words, we can say that the actual PageRank contains more information about both the quality and popularity of the page, but a Silly PageRank would rather consider only its "popularity". Let's analyze the strengths and weaknesses of such an algorithm.

Strengths of Silly PageRank:

- From the webpage owner perspective, it would be much easier to create a higher rank for the page.
- It would be easier to calculate the PageRank of a page.

Weaknesses of Silly PageRank:

- The goal of getting the most relevant information at the top would be defeated, by getting the most popular information at the top. For instance, consider a bad article which a lot of websites refer to, but which contains no hyperlinks to other sources. That would still appear first, although it is not evidence-based or relevant (worth the time reading).
- Generally, this algorithm has such very limited information to measure the relevance of the article.

- From the user perspective, the most relevant pages would be those who gamed the algorithm, which results mostly in dissatisfaction.

Reducing the algorithm to only considering incoming edges means a higher chance of gaming the system, as well as less likelihood of providing the most relevant information first. This could further be translated into less interested users, and less profit for the search engine company.

To see how much better the actual PageRank algorithm does, while keeping in mind that its goal is to provide the most relevant information to the user, allow the most relevant pages to have the highest rank, and prevent web page owners to game the system, we could conduct a SWOT Analysis[2] of the existing algorithm:

Strengths of PageRank:

- The current algorithm fulfills its goal with quite high accuracy.
- The search of billions of existing pages is expected to take a lot of computational power, yet this algorithm has found an efficient way to reduce the auxiliary memory complexity, by only considering the nodes that are linked to the current page. It reduces the search of billion pages to only the ones that matter the most locally.
- The algorithm can help us for predictive purposes with high accuracy.
- Not every part of the actual algorithm is accessible, and that hidden secret ingredient makes Google less prone to spamming.

Weaknesses of PageRank:

- The algorithm is fairly public, thus creating more space for spammers to create web pages that would increase the rank of other web pages. This could end up in less relevant/credible information being spread.
- It only accounts for content-relevance, and not time-relevance.

Opportunities of PageRank:

- Add more parameters to measure the credibility, besides relevance. One of the biggest demands in the world is to get reliable information in the Internet world, and by adding a factor that counts for credibility would address that need. At the same time, it is a social responsibility to provide reliable information, if your product is the source of information.
- Account for time-relevance, not only content-relevance. As much as we know about the algorithm, new, more relevant, and updated pages will not be shown at the top of the search if they have a lower rank than old pages. The users seek for the latest updates, thus, such a feature in the algorithm could increase the user satisfaction.

[2]**strategize**: Applied SWOT Analysis to analyze the PageRank algorithm.

Threats of PageRank:

- Google is not the only search-engine, which means that it has to provide the most updated, relevant and important information better than its competitors (i.e. Bing).
- Spammers are always looking to cheat the PageRank algorithm and get their pages to appear as most relevant. Especially with a lot of PageRank formula being easily accessible, this gives more information to spammers to counterattack the purpose of the algorithm.

In summary, PageRank algorithms optimizes on maximizing relevance and minimizing spamming, better than Silly PageRank, because it accounts for more factors. In order to improve it, Google could add more factors based on time-relevance and credibility.

5 Code

https://gist.github.com/almagashi/1d2586d05efa3ae4359873d67d7c9161