EE232E PROJECT 1

Random Graphs and Random Walks

Devanshi Patel	504945601	devanshipatel@cs.ucla.edu
Ekta Malkan	504945210	emalkan@cs.ucla.edu
Pratiksha Kap	704944610	pratikshakap@cs.ucla.edu
Sneha Shankar	404946026	snehashankar@cs.ucla.edu

INTRODUCTION

Analysis of Graphs and Networks is an important study in the field of Computer science. Search Engines, E-commerce sites, Internet-Service Providers etc. store, process and forward huge amounts of data on daily basis. Data packets travel across multiple nodes in a network before reaching the destination from the source.

A user can take many routes from the source to the intended destination. Studying properties of random graphs can lead us to interesting insights in data flow across the network, as well gauge the impact of addition and deletion of nodes across the network.

In exploring graphs through Random Walks, a walker is placed at a randomly selected node across the network. The goal here is to perpetually visit all nodes of the network by traversing the edges of the graph. This Random Walk model has been the basis of search engine optimizations, and is used for crawling web, ranking pages and websites, performing network maintenance etc.

In this project we generate Random networks as well as study Random walks performed across those networks. We then study Preferential Attachment Models and observe their differences. The code has been developed using R kernel in Jupyter notebook.

1. Generating Random Networks

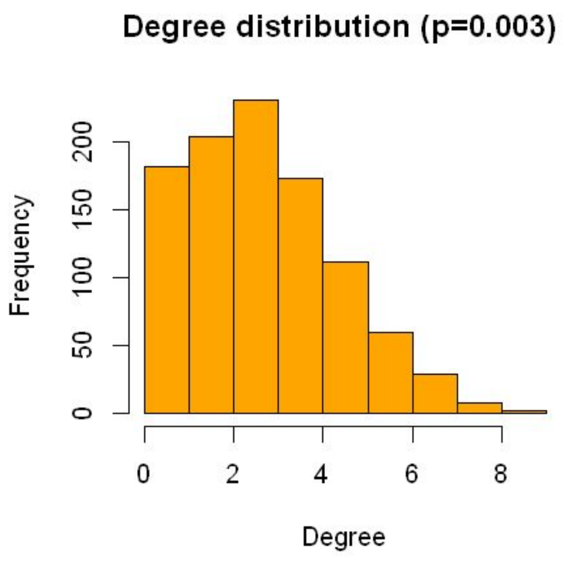
In this part , we generate Random Networks using two models, namely :

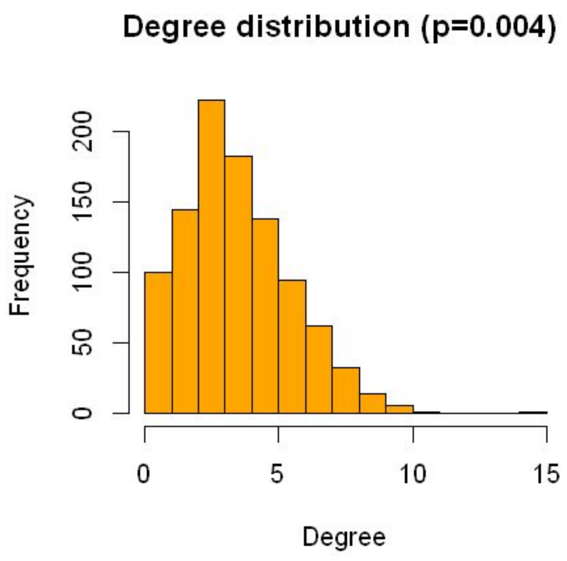
- Erdos-Renyi Model
- Preferential Attachment Model

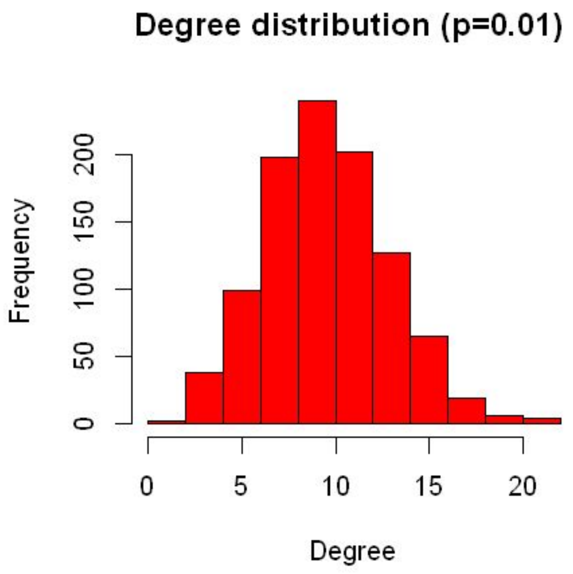
1.1 Create random networks using Erdös-Rényi (ER) model

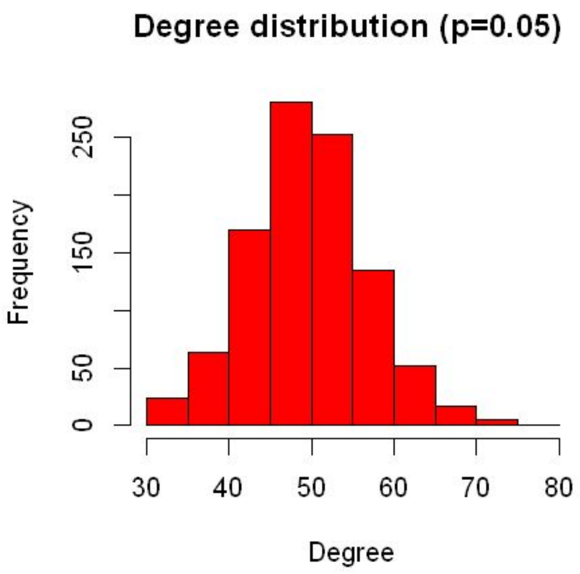
In Erdös-Rényi (ER) model, every new edge is created with the same probability. For a fixed number of “n” nodes, we can either state the probability “p” of an edge being present between any two nodes, or we can state “m”, which means m edges are chosen from the set of all possible edges.
In Graphs and Networks, the degree of a node is the number of edges that are connected to it. The degree distribution is simply a probability distribution of degree of all the nodes in the network.

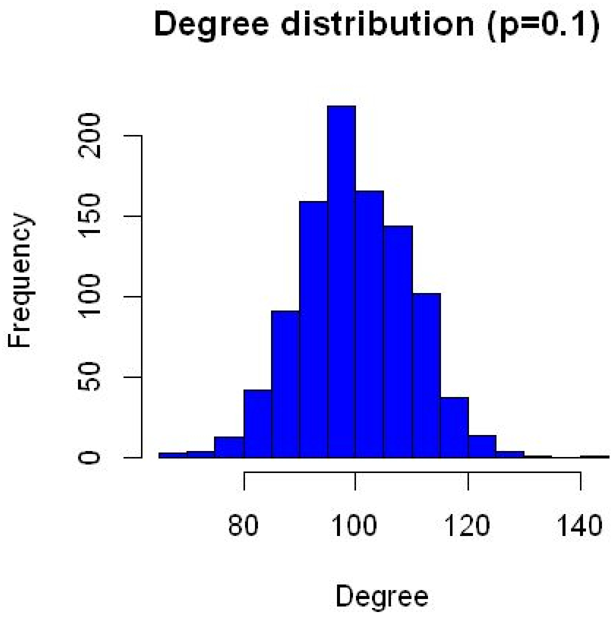
1.1(a) In this task, we chose different values of probability “p” in the Erdos Renyi Model for Graph generation and observed the degree distributions of the generated graphs.







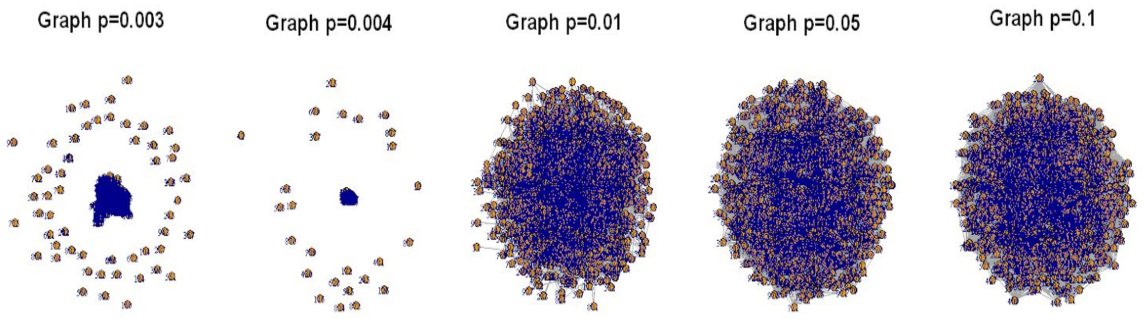




As we can see, the probability distributions of Erdos Renyi Graph models follow a "**Binomial Distribution**".

This is because for each pair of nodes there is choice of whether the edge will exist or no with a probability p. This is same as for every edge from the set of all possible edges, we toss a coin, if its heads we create an edge and if it is tails we do not create an edge. Each single toss of coin would follow a "Bernoulli Distribution", and if the coin is tossed multiple times, the sum of the outcomes (of individual Bernoulli distributions) would follow **Binomial Distribution**.

For further analysis , we will plot the graphs as follows:



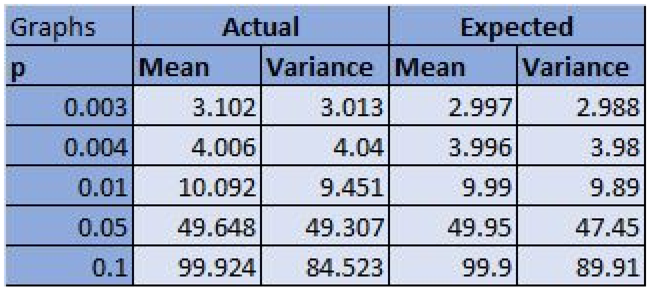
From the graphs above, we can clearly see how the graphs become denser as “p” increases.
We also calculated the expected mean and variance of degree distributions, and compared them with the actual values as follows:
For calculating the expected mean degree when self loops are not present, we will use the formula as :

Mean=(n-1)p

For calculating expected variance degree, ignoring self loops ,

Variance = np(1-p)

Hence, for n=1000 we get :



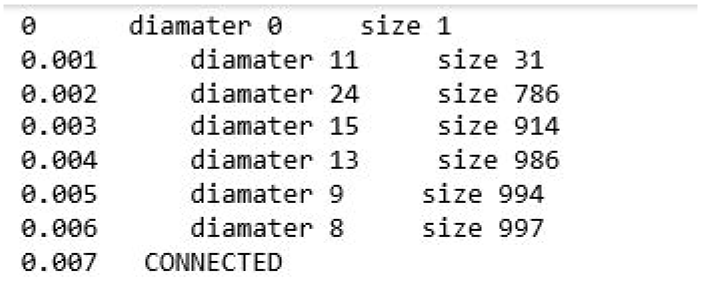
Graphs	Actual		Expected	
p	Mean	Variance	Mean	Variance
0.003	3.102	3.013	2.997	2.988
0.004	4.006	4.04	3.996	3.98
0.01	10.092	9.451	9.99	9.89
0.05	49.648	49.307	49.95	47.45
0.1	99.924	84.523	99.9	89.91

We can observe from the above results that the Actual and Expected values of Mean and Variance for Erdos Renyi Random Graphs are almost similar. As the value of p increases, the graph shifts towards higher degree.

1.1(b) For each p and n=1000, Are all random realizations of the ER network connected?
No, not all Random Graphs formed are connected.

Numerically estimate the probability that a generated network is connected. For one instance of the networks with that p, find the giant connected component (GCC) if not connected. What is the diameter of the GCC?

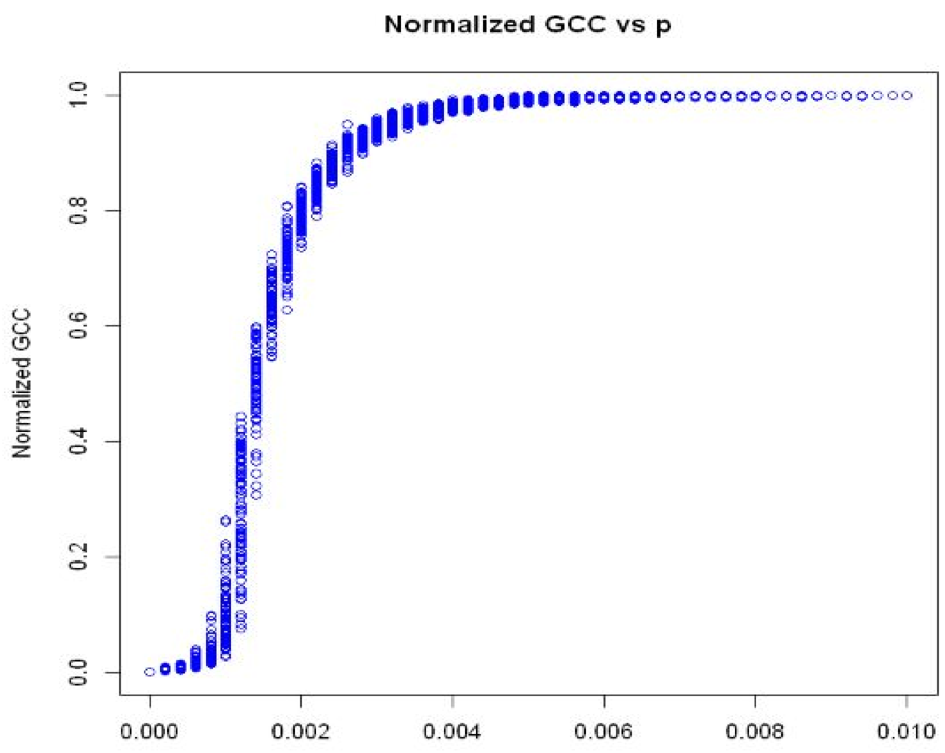
We find that when the value of p is less than a certain threshold, then the graphs are not connected. However for value of the probability at and above the threshold, the graphs are connected. To numerically estimate this threshold probability, we swept for p from 0 to 1 in intervals of 0.01 :



```
0      diamater 0     size 1
0.001      diamater 11     size 31
0.002      diamater 24     size 786
0.003      diamater 15     size 914
0.004      diamater 13     size 986
0.005      diamater 9     size 994
0.006      diamater 8     size 997
0.007   CONNECTED
```

As we can see from the results, the graphs become connected starting p=0.07. Also we can see that till p=0.01, the size of the GCC is extremely small (31), whereas it grows somewhere between 0.01 and 0.02 and at 0.02 the size is drastically increased to 786 and then goes to 914. This tells us that the size of the Giant Connected Component of the graph in Erdos Renyi Model is related to the probability p in a special way.
We follow the below tasks to further investigate into this relationship.

1.1(c) For n = 1000, sweep over values of p in this region and create 100 random networks for each p. Then scatter plot the normalized GCC sizes vs p.



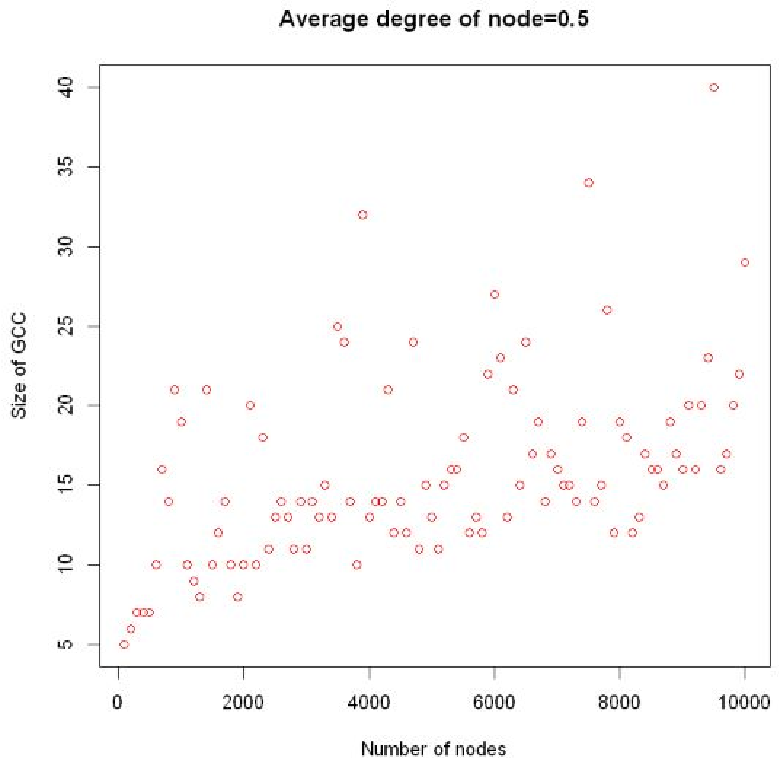
As per the graph above, for small p, there are very few edges on the graph. Almost all vertices are disconnected. The components are small (size about 31), with size of the GCC in O(log n), independent of p.

When we increase p ,then at p = 1/n i.e. when np=1, we see a phase transition happening. Thus at **p= 0.001**, we can see one giant connected component emerging.

From, p=0.001 to p=0.002, we can observe a drastic increase in the size of GCC. We can also observe, that at approximately **p=0.007**, the graph becomes fully connected.

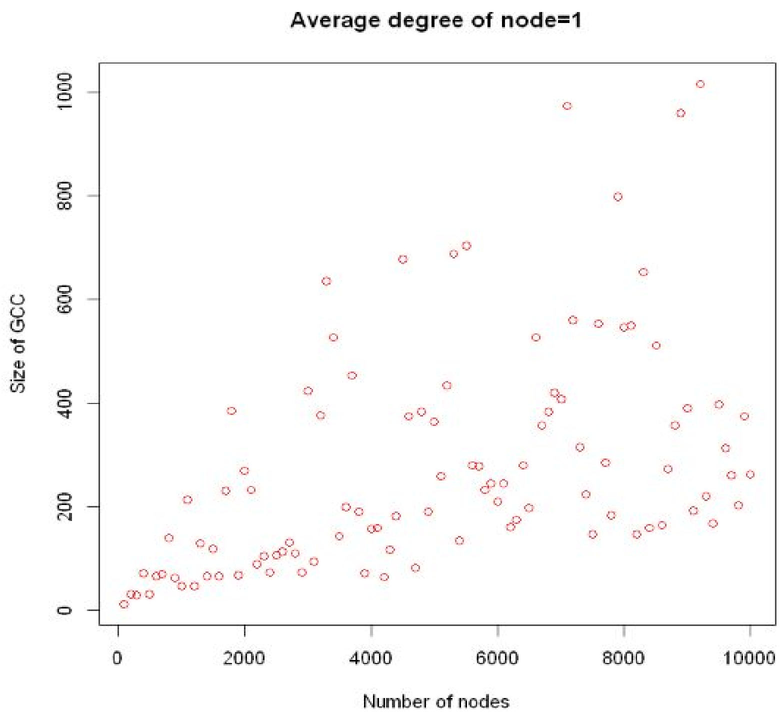
1.1(d)

i) Define the average degree of nodes c = n p = 0.5. Sweep over number of nodes, n, ranging from 100 to 10000. Plot the expected size of the GCC of ER networks with n nodes and edge-formation probabilities p = c=n, as a function of n.What trend is observed?

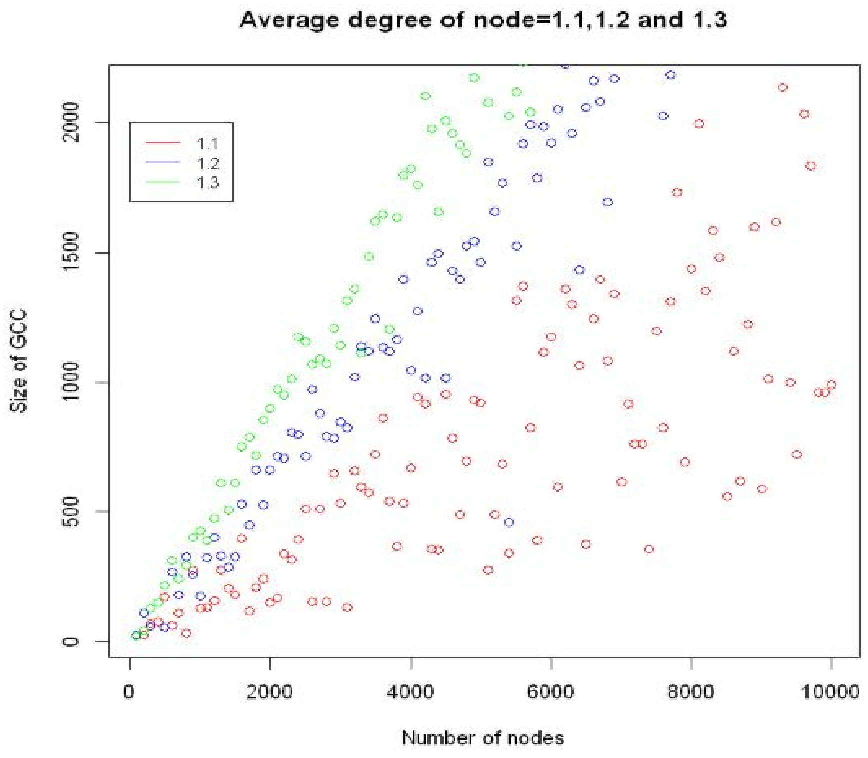


The above graph shows the size of GCC against the number of nodes where the average degree of node is 0.5. We can observe that as the value of n increases, the size of gcc increases.

ii) Repeat the same for c = 1.



iii) Repeat the same for values of c = 1:1; 1:2; 1:3, Plot these three values in a single plot.



From the above graph, we observe that as the value of c increases, the slope of the size of gcc against number of nodes increases, which means that GCC grows faster with a larger value of c. Consider for a graph with number of nodes=n,

GCC(1.1)<GCC(1.2)<GCC(1.3).

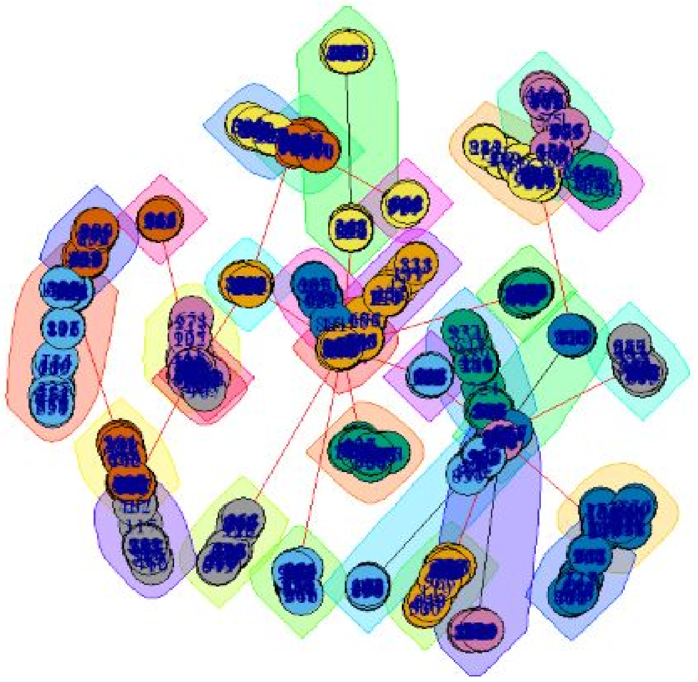
1.2 Create Random Networks using Preferential Attachment Model

Preferential attachment means that the more connected a node is, the more likely it is to receive new links. Nodes with higher degree have stronger ability to grab links added to the network. Intuitively, the preferential attachment can be understood if we think in terms of social networks connecting people.It demonstrates the “Rich gets Richer” phenomena. It shows how a network grows over time.

1.2. a)Create an undirected network with n = 1000 nodes, with preferential attachment model, where each new node attaches to m = 1 old nodes. Is such a network always connected?

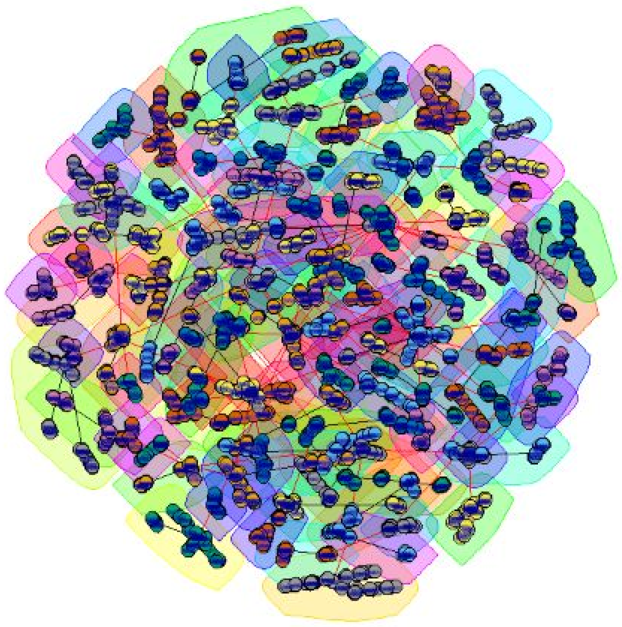
Yes, a preferential attachment network is always connected. During the formation of a preferential attachment model, any new node added, gets connected to one of the previous node(preference is given depending on the the connectivity of the node).
We created a preferential attachment network 100 times for n=1000, it was connected always.

1.2. b) Use fast greedy method to find the community structure. Measure modularity.
Modularity of a graph is used to measure the strength of division of a network into modules (also called groups, clusters or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules. Modularity is often used in optimization methods for detecting community structure in networks.



The above figure shows the community structure of a graph created with preferential attachment model with 1000 nodes. Fast greedy algorithm was used to find the community structure. **Modularity of this graph is 0.9326.**

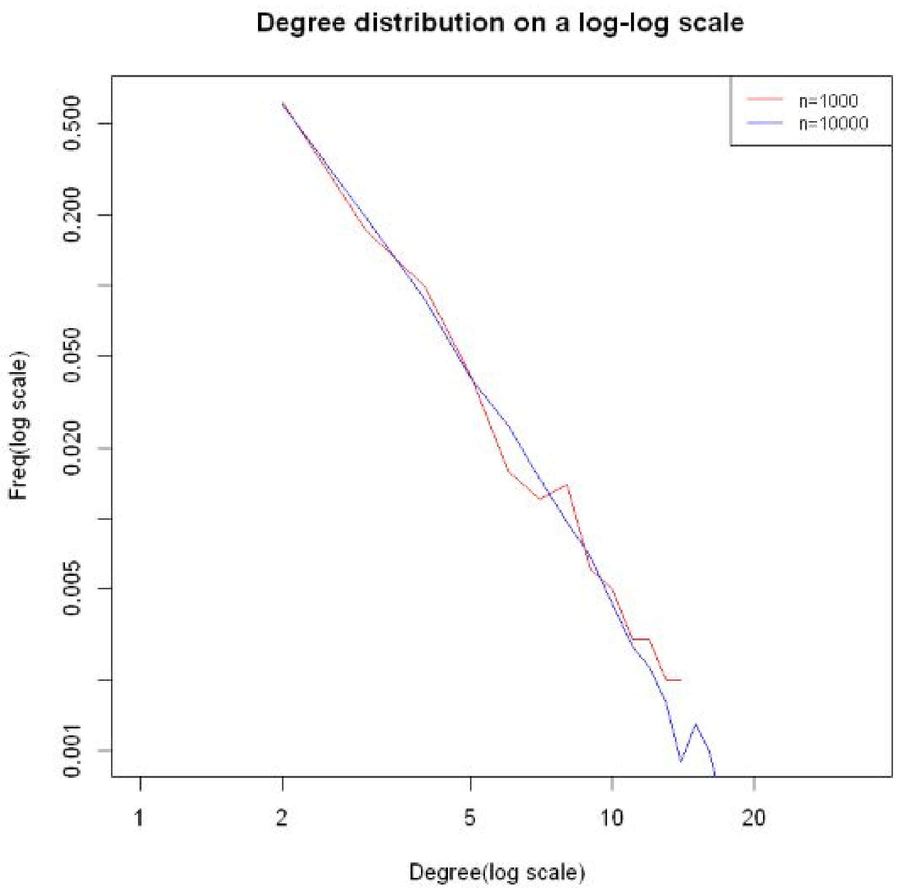
1.2. c)Try to generate a larger network with 10000 nodes using the same model. Compute modularity. How is it compared to the smaller network's modularity?



Modularity of this graph is 0.9784.

We observe here that as N (number of nodes) increases, modularity of the graph increases. Also, we see that in the graph of 1000 nodes, the communities are not very dense. However, for the graph with 10000 nodes, the communities are very dense which increases the modularity.

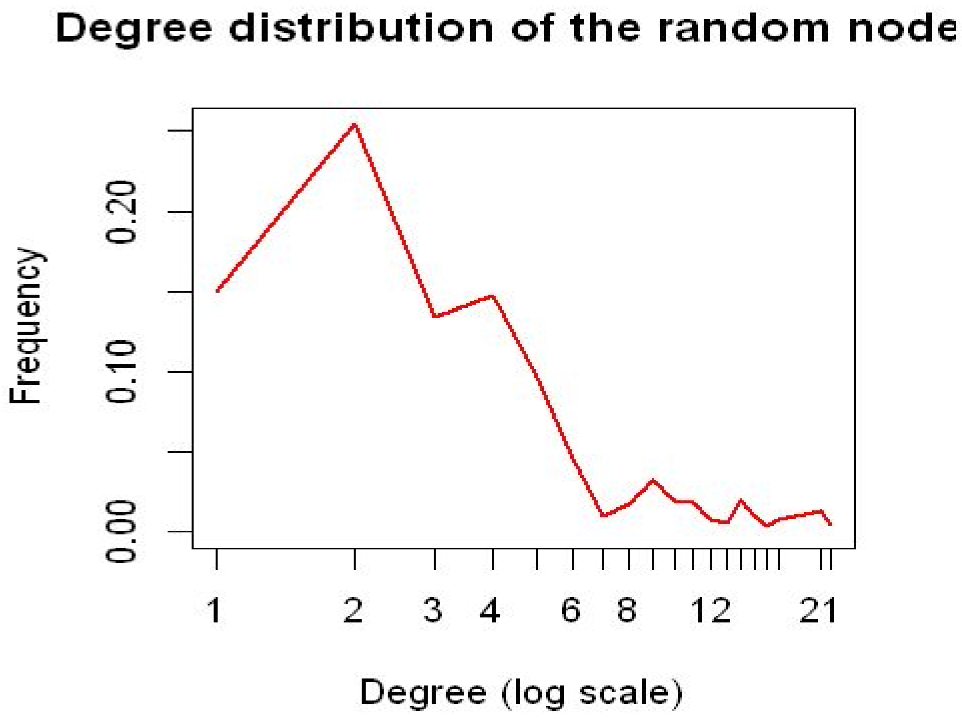
1.2. d)Plot the degree distribution in a log-log scale for both n = 1000; 10000, then estimate the slope of the plot.



Slope of both the plots above is approximately -0.0295.
Thus we see that irrespective of the number of nodes in the graph, the degree distribution of preferential attachment model (barabasi model) is the same.

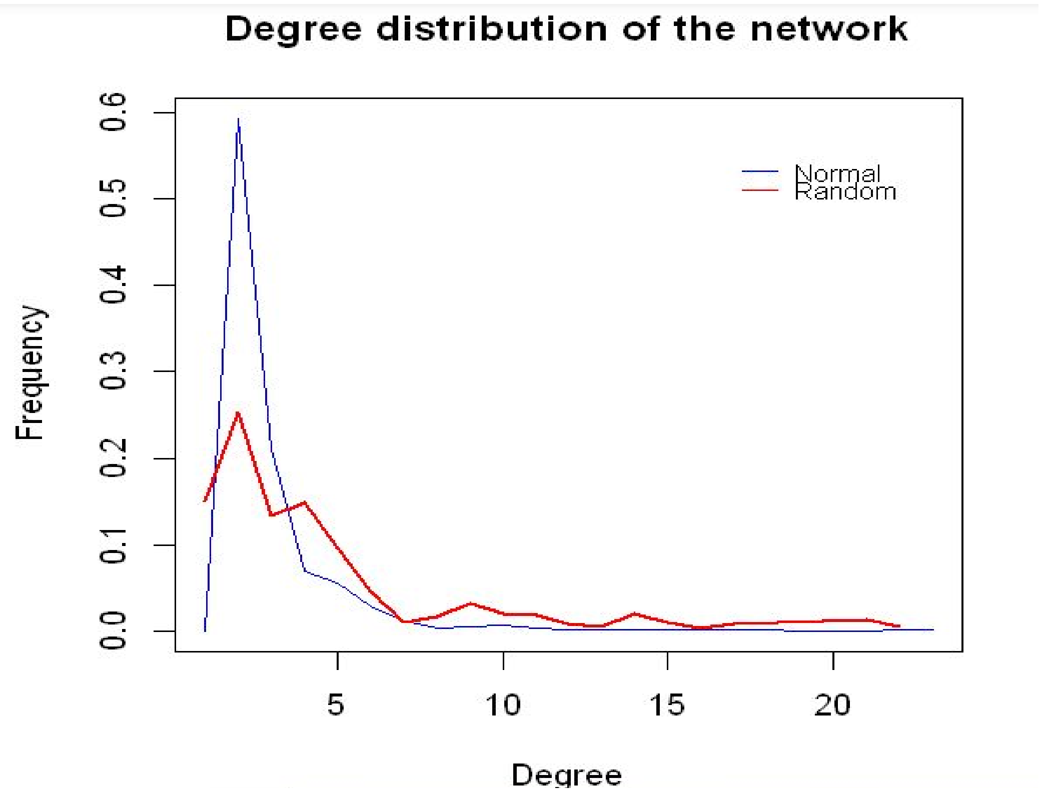
1.2. e)You can randomly pick a node i, and then randomly pick a neighbor j of that node. Plot the degree distribution of nodes j that are picked with this process, in the log-log scale. How does this differ from the node degree distribution?

Graph by randomly picking a node and finding its distribution on log scale: n=1000



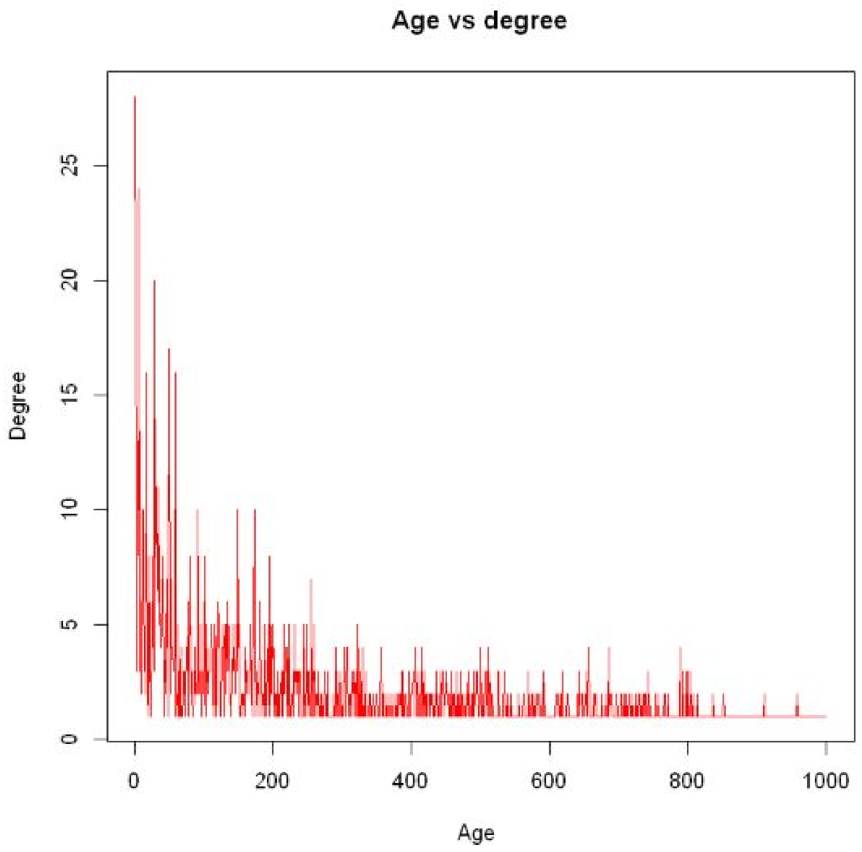
Slope of the graph above is: **-0.0125**

Comparison of this graph obtained by randomly picking up the node with the degree distribution of the entire graph.



As we can see from the graph above, the original graph of 1000 nodes has some nodes with a very high degree distribution. However, when we randomly pick up nodes from this graph, the chance of picking up those high degree nodes is less. Hence the peak of the graph of randomly selected nodes is far lesser than the original graph.

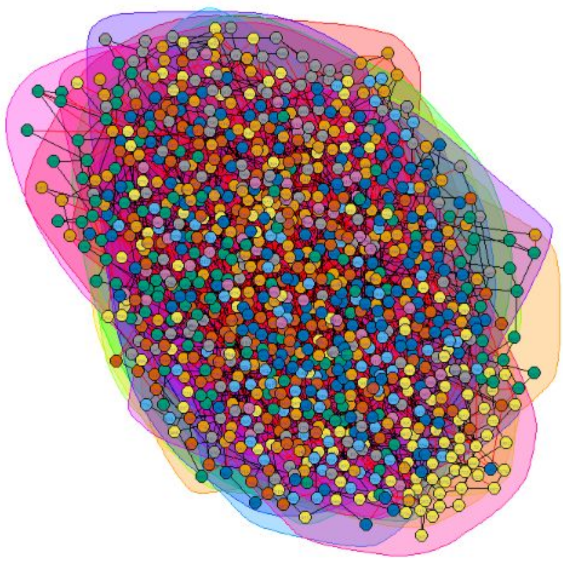
1.2. f) Estimate the expected degree of a node that is added at time step i for 1 to 1000. Show the relationship between the age of nodes and their expected degree through an appropriate plot.

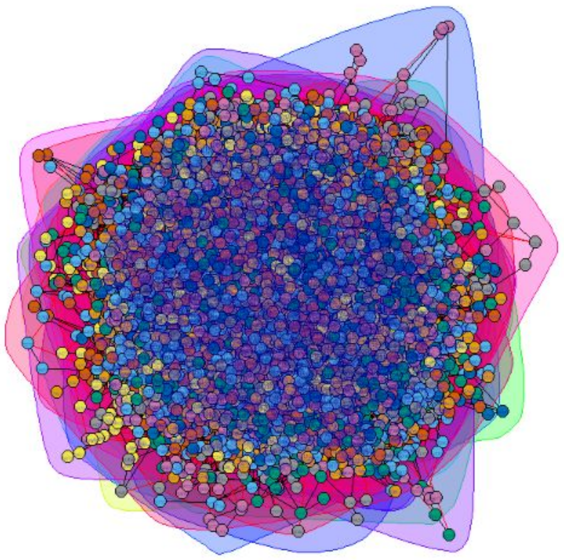


The above graph shows the relation of Age vs the Degree of a node. We can observe that the oldest nodes have higher degree compared to the newest nodes. This is because when the nodes are added to the graph , the node to which the new node will be added is decided on the degree of all the present nodes, which means that a node with higher degree will have a higher chance of getting attached the new node.This explains the above trend in the graph.In this graph and throughout the report, **the age 0 means the oldest node and age 1000 means the youngest node.**

1.2. g)Repeat the previous parts for m = 2; and m = 5. Why was modularity for m = 1 high?

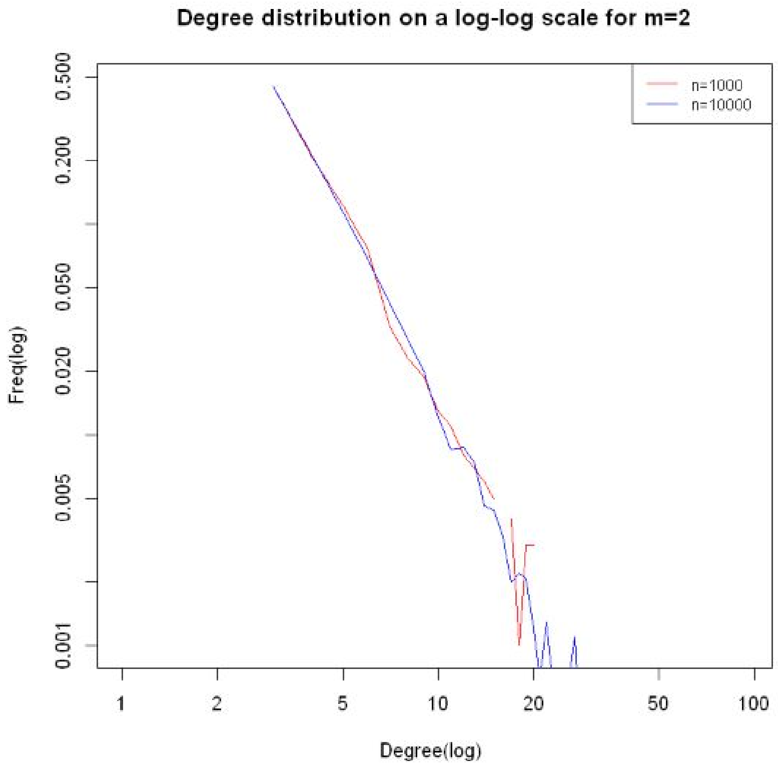
Modularity for n=1000 and m=2 is **0.5217459**
Modularity for n=10000 and m=2 is **0.529585**



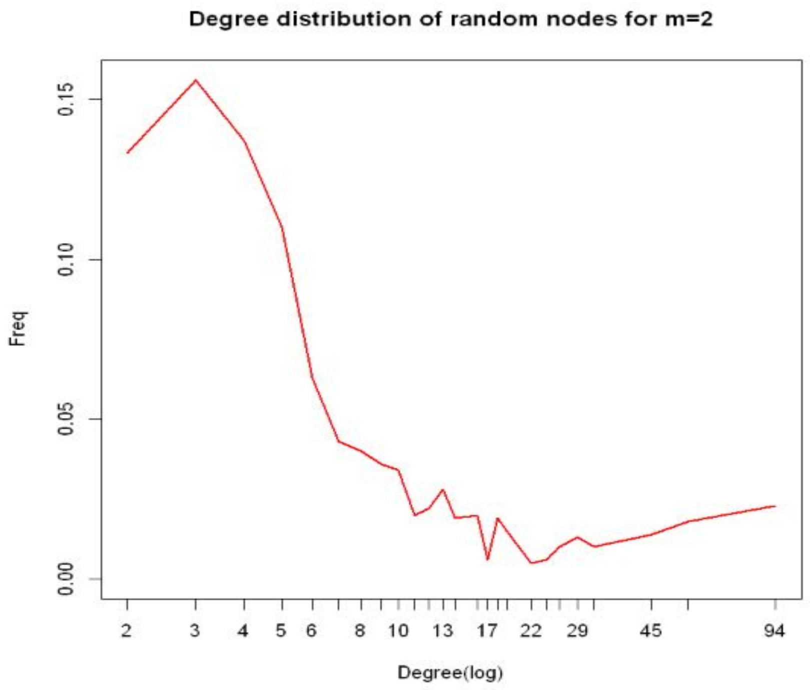


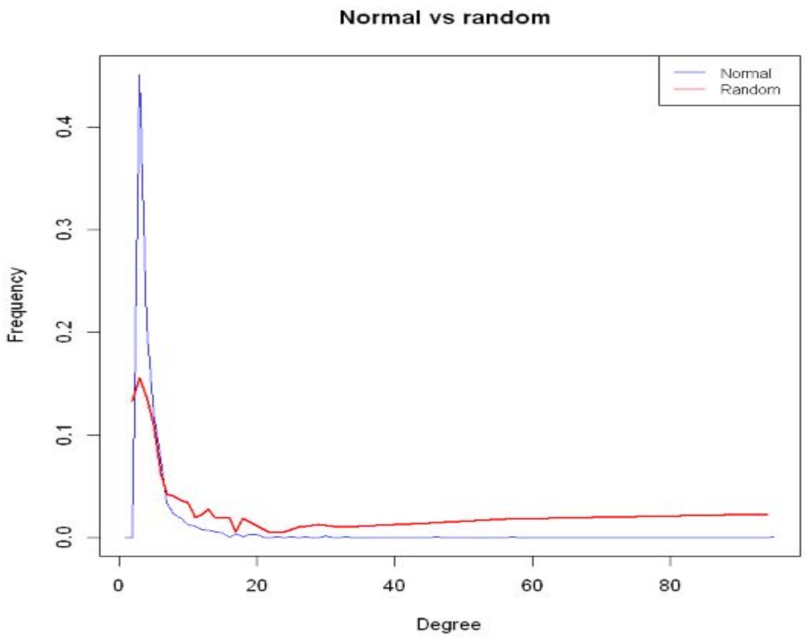
n=1000 n=10000

We can see that the communities within these graph are not very sparse which leads to less modularity.

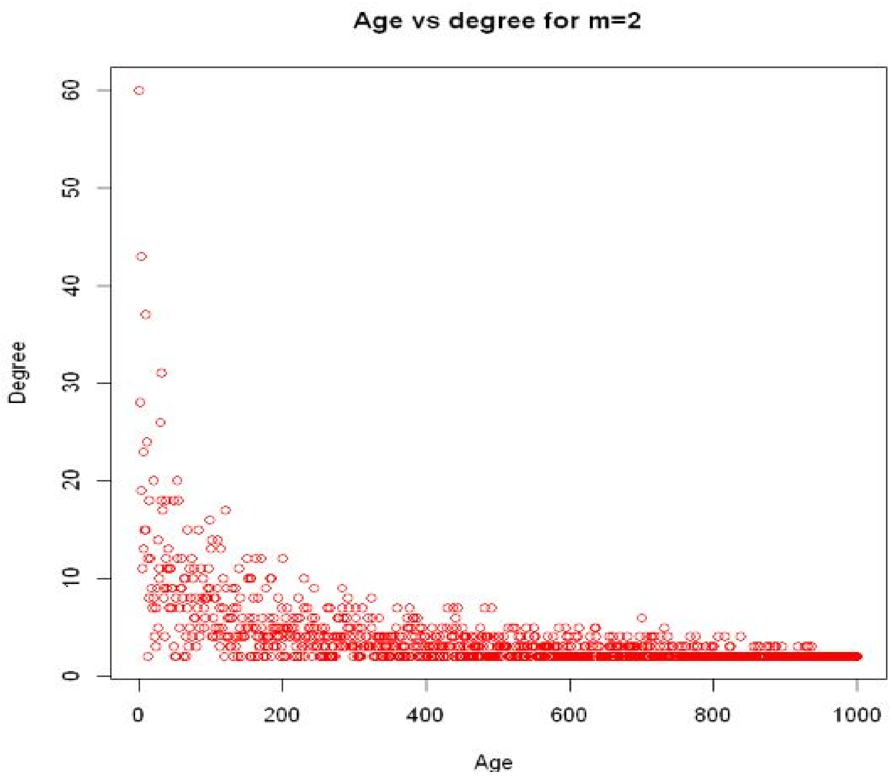


For m=2, the slope of degree distribution for n=1000 and 10000 is almost same approximately equal to -0.03.





The above graphs shows the degree distribution for normally generated graph and of randomly selected nodes. We can observe that the distribution for randomly selected nodes have a lower peak compared to the normal one.

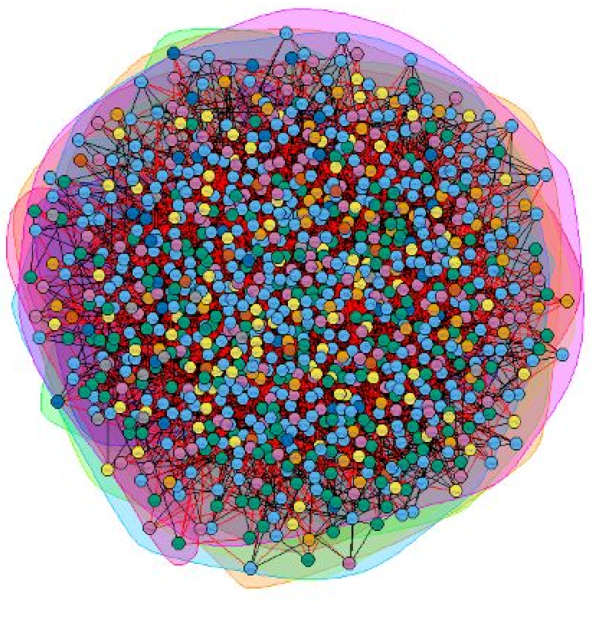


The oldest nodes have higher degrees compared to the youngest ones.

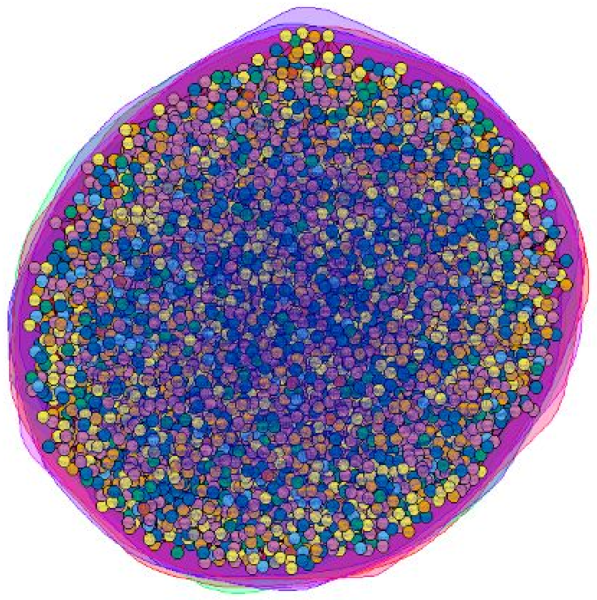
For m=5

Modularity for n=1000 and m=5 is **0.2704008**

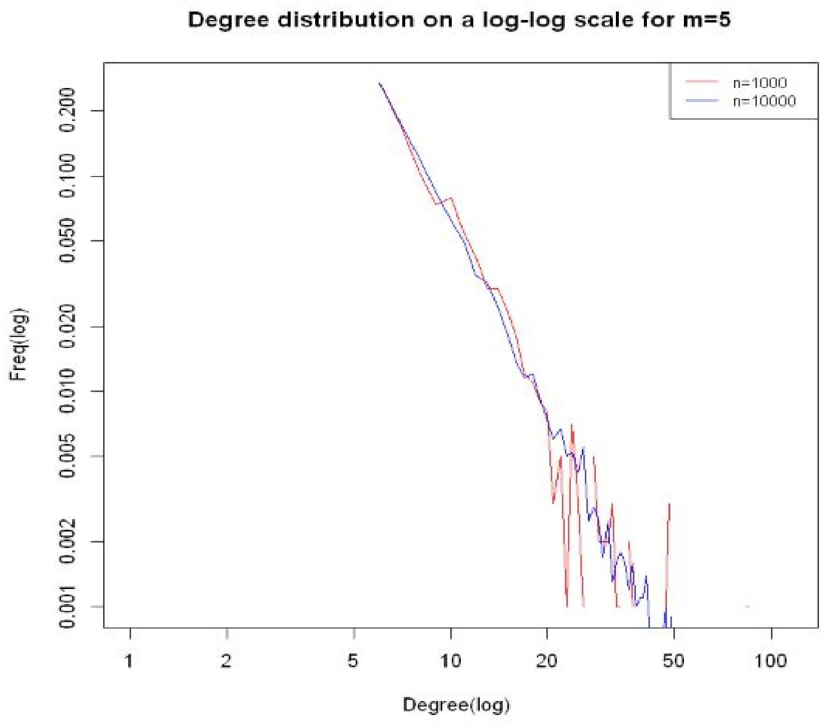
Modularity for n=10000 and m=5 is **0.2753281**



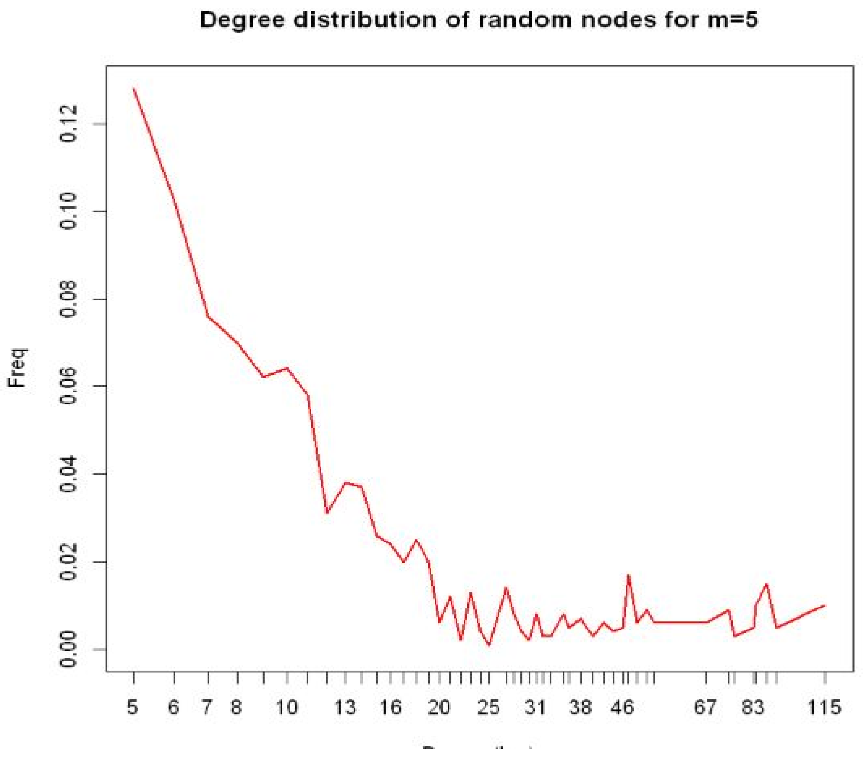
n=1000

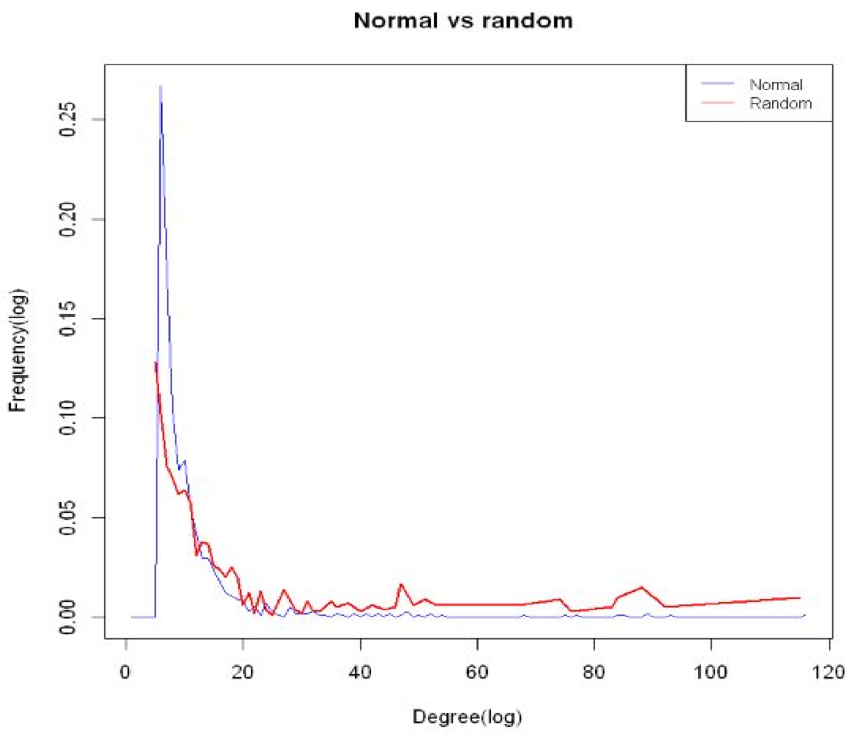


n=10000

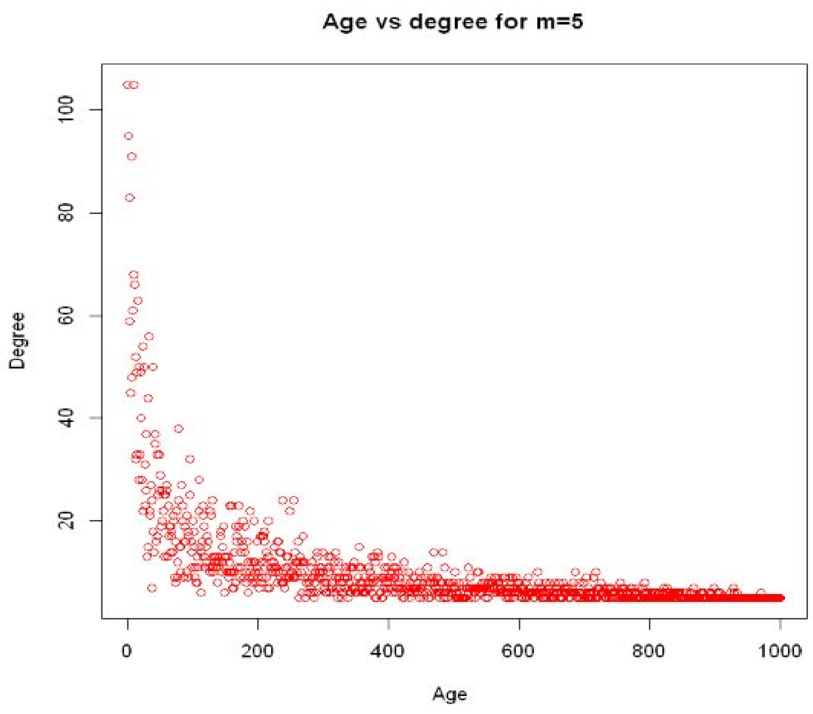


For m=5, the slope of degree distribution for n=1000 and 10000 is almost same approximately equal to -0.03.





The above graphs shows the degree distribution for normally generated graph and of randomly selected nodes. We can observe that the distribution for randomly selected nodes have a lower peak compared to the normal one.

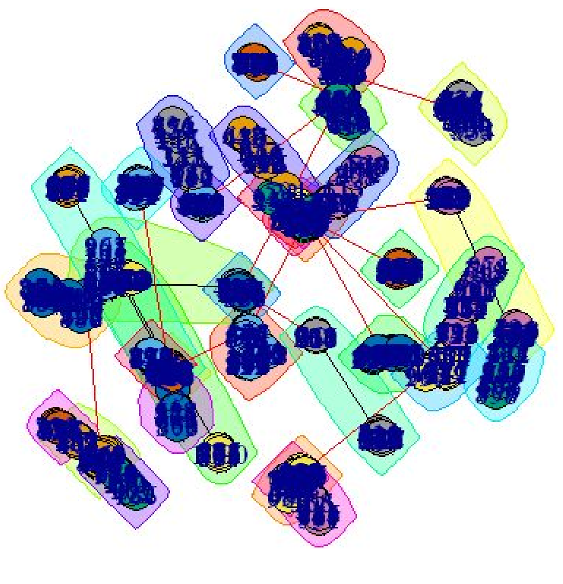


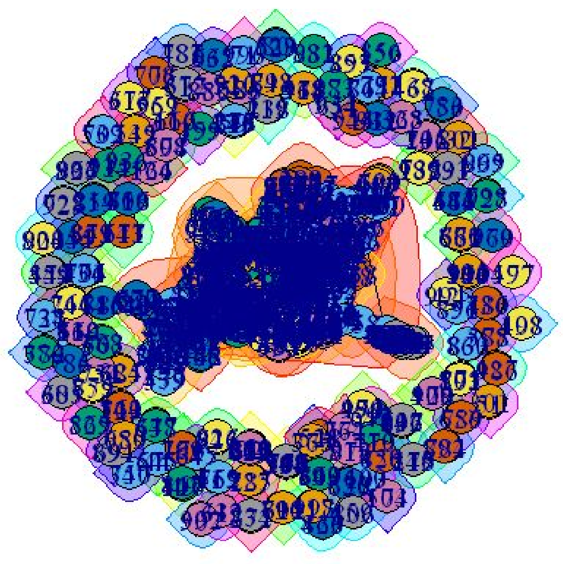
Again, the oldest nodes have higher degree compared to the youngest ones.

From the above graphs for m=1,2 and 5, following observations can be made,

1. For all preferential attachment models, older nodes have higher degrees
2. **Modularity is high for m=1. Modularity decreases as m increases.This is because for higher values of m, more edges are formed, which in turn decreases the sparseness of edges between the communities. Communities are not very well separated with more number of edges. Therefore, higher values of m leads to lower modularity.**
3. Degree of oldest node for m=1,2 and 5 is approximately 27, 60 and 110 respectively. The oldest node for m=5 has the highest degree. This is because for m=5, the set of possible edges increases.

1.2. h) Again, generate a preferential attachment network with n = 1000, m = 1. Take its degree sequence and create a new network with the same degree sequence, through stub-matching procedure. Plot both networks, mark communities on their plots, and measure their modularity. Compare the two procedures for creating random power-law networks.


Normal graph


Simulated Graph

Modularity of the normal graph = 0.9311
Modularity of the simulated graph = 0.8495

We can observe that the original graph has a higher modularity as compared to the graph simulated with a known degree distribution.

Communities in a graph are groups of densely interconnected nodes that are only sparsely connected with the rest of the network. We can observe that the normal graph has higher modularity as compared to the simulated graph. Also, from the graph we can see that the communities are very well separated for this graph.

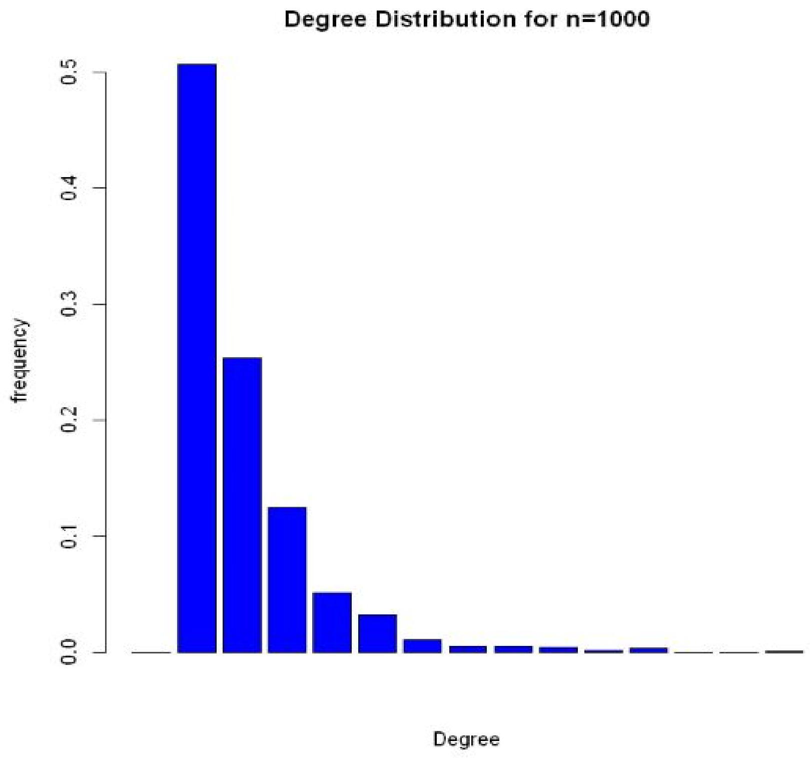
The simulated graph has a lower modularity, which is evident. The graphs depicts that the different communities of the simulated graph are extremely close to each other and not well separated.

1.3.Preferential Attachment Model that penalizes the age of a node

1.3. a)Each time a new vertex is added, it creates m links to old vertices and the probability that an old vertex is cited depends on its degree (preferential attachment) and age. In particular, the probability that a newly added vertex connects to an old vertex is proportional to:

$$P[i] \sim (ck_i^{\alpha} + a)(dl_i^{\beta} + b),$$

where ki is the degree of vertex i in the current time step, and li is the age of vertex i. Produce such an undirected network with 1000 nodes and parameters m = 1, α = 1; β =-1, and a = c = d = 1; b = 0. Plot the degree distribution. What is the power law exponent?

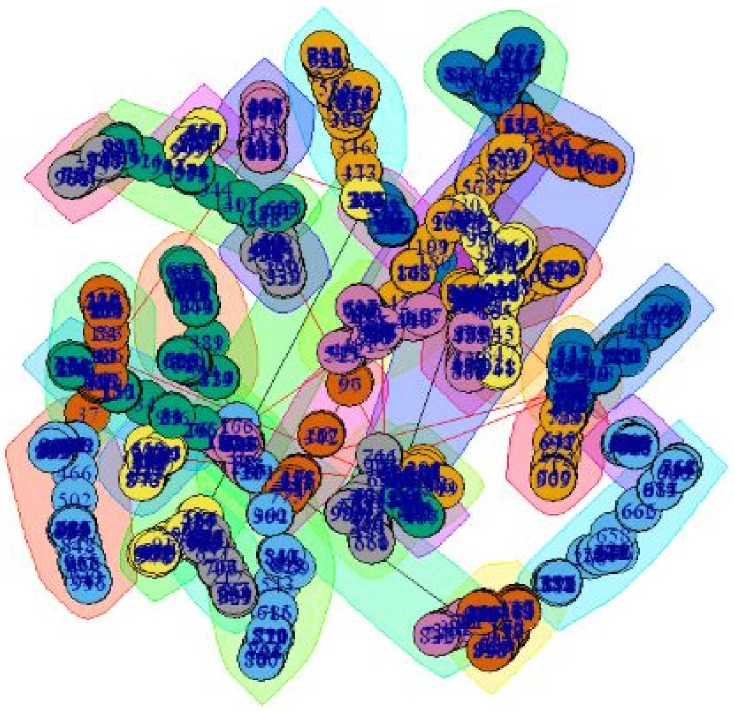


For the specifications given in the question, the degree distribution of the network is shown in the above figure. We can observe that the degree distribution follows the **power law distribution**.

We calculated the the exponent of this power law distribution and found it to be **4.176.**

1.3. b)Use fast greedy method to find the community structure. What is the modularity?

The community structure is given in the figure below. Modularity is **0.9347**.



2. Random Walk on Networks

In this part, we perform random walk on the following networks:

- Erdös-Rényi networks
- Preferential attachment networks

Further, we also use random walk to simulate pagerank as well as personalized pagerank.

2.1 Random Walk on Erdös-Rényi networks

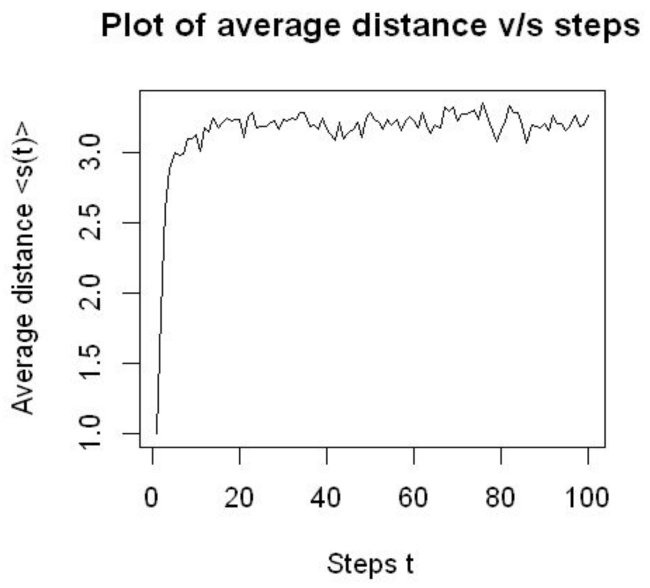
2.1 (a) Here, we created an undirected random network with 1000 nodes, and the probability p for drawing an edge between any pair of nodes is set to 0.01. We have used the erdos.renyi.game function for the same.

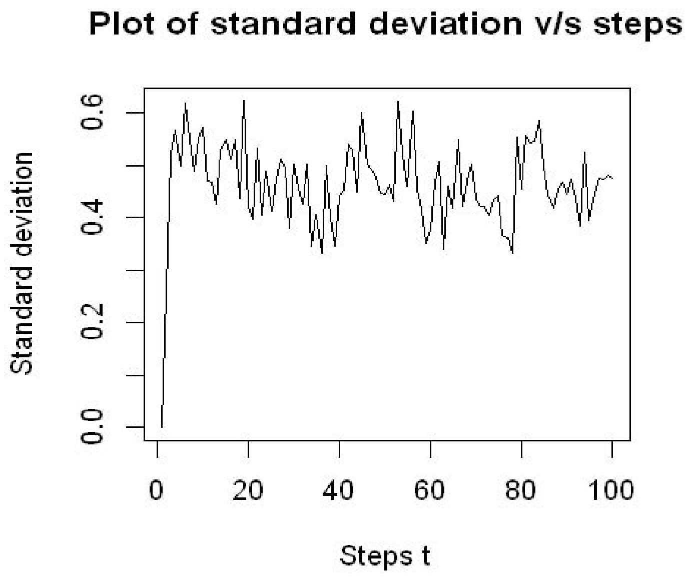
2.1 (b) In this part, we select a random node to start from and then the random walker begins a walk through the graph from this node. We then measure the distance of the walker at each time step from this starting node for time steps ranging from 1 to 100. We repeat this procedure several times with different and random starting nodes to find the average distance, <s(t)> which is the shortest path length at each time step.

We further measure the standard deviation of this distance which is given by the following formula:

$$\sigma^2(t) = <(s(t) - <s(t)>)^2>$$

To observe the change in behavior of average distance and standard deviation with the number of steps, we plot both of these quantities as shown below.



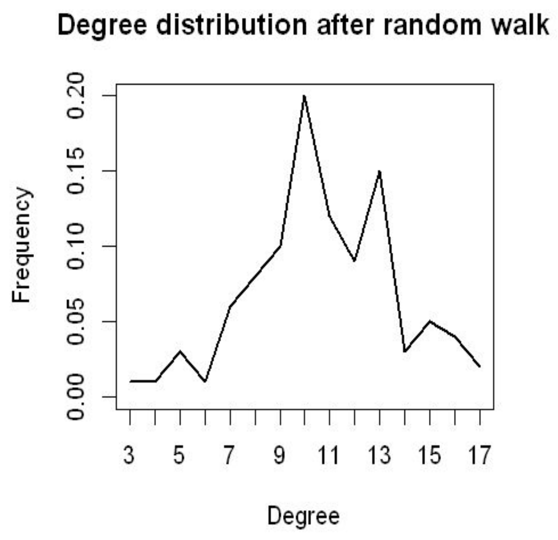


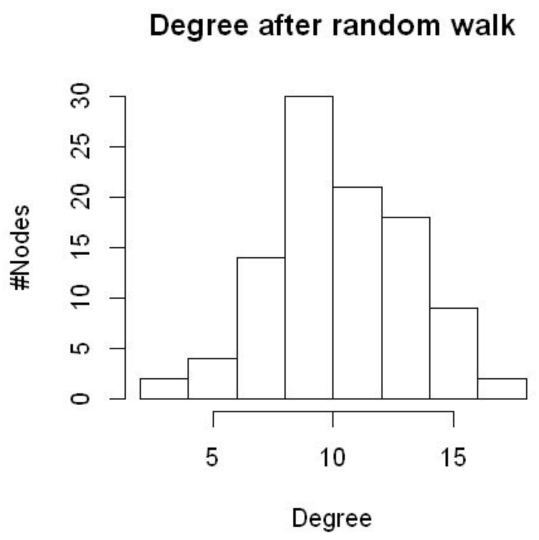
Observations:

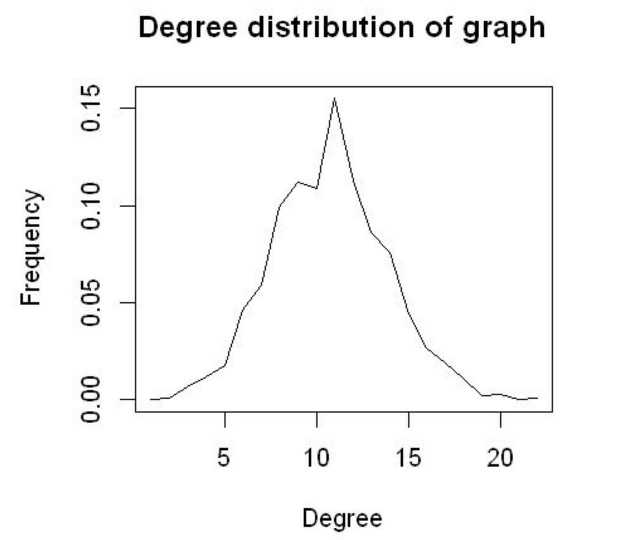
As the random walker starts from the initial node and walks through the graph, the average distance of the node visited at each time step t also increases till 10 steps. After that, the average distance converges to 3.5.

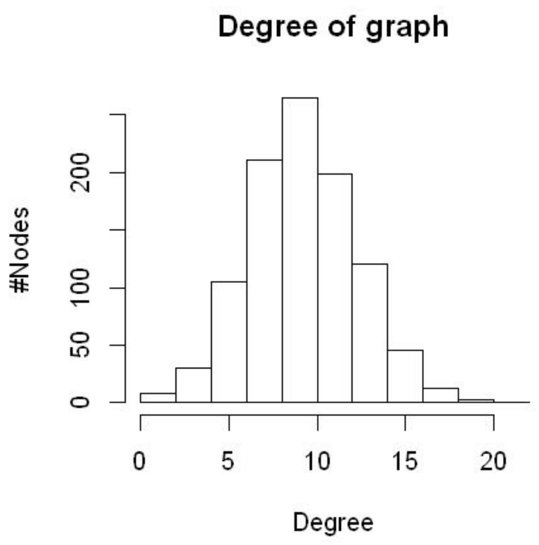
As seen from the plot above, the standard deviation is small and in the range of 0.3 to 0.6. It converges to 0.5. The range of standard deviation indicates that shortest path length at each time step is not far from the average distance.

2.1 (c) In this part, we measure the degree distribution of the nodes that were reached at the end of random walk. And then we compare it with the degree distribution of the graph.









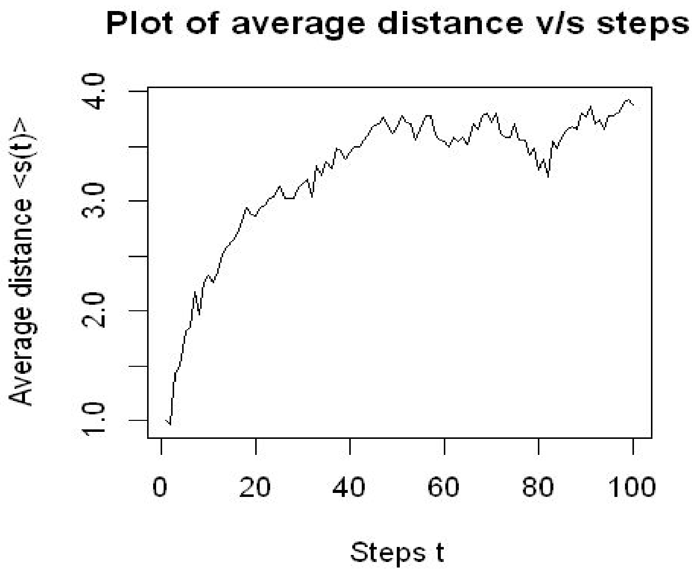
Observations:

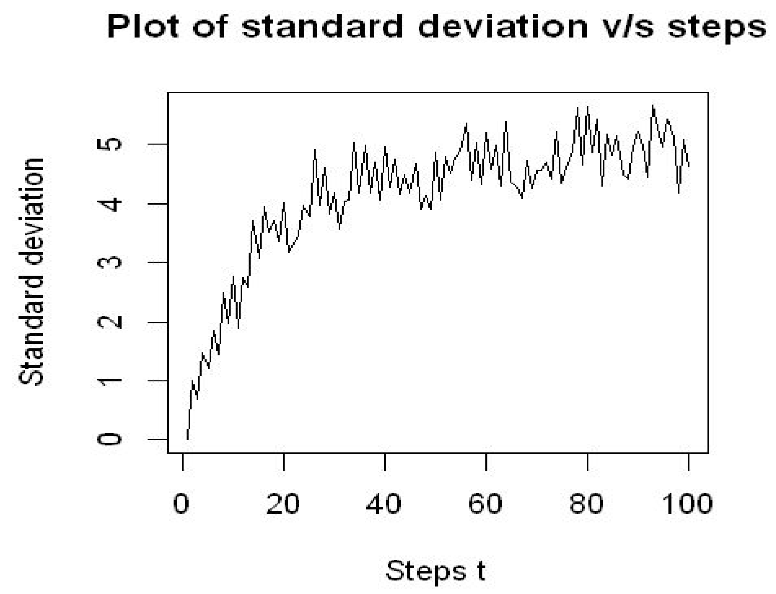
From the plots of degree distribution of graph and the distribution at the end of the random walk, we can say that the overall distribution is quite similar and resembles to normal distribution. It is just smoother in case of the entire graph. We can also say that majority of the nodes have degree of 10 (or around 10). There are very few nodes which have extremely high or low degrees.

2.1 (d) We now repeat the above calculations for 100 and 10000 nodes in the random network.

For 100 nodes:
As seen from the plots below, the random network generated with 100 nodes and 0.01 probability is a disconnected graph. Hence, we first extract giant connected component(GCC) and then perform the random walk on the graph.

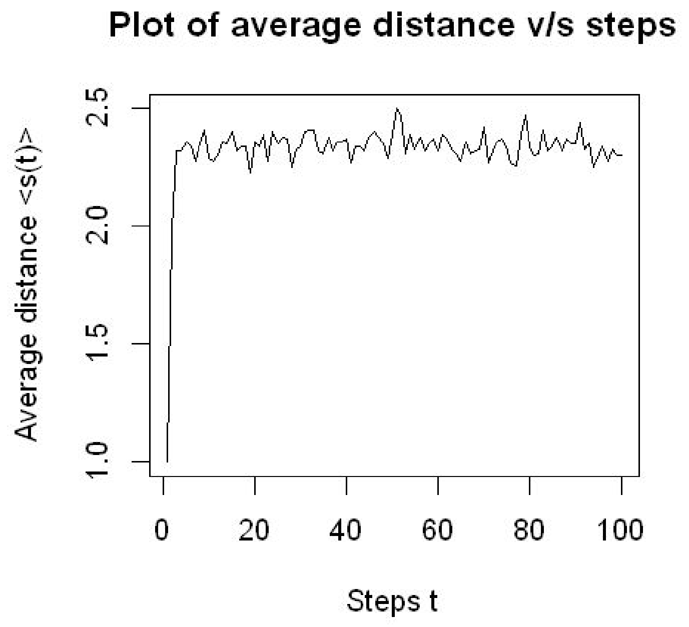
The value of average distance goes up to 4 for 100 steps. If we observe the plot of standard deviation, we can see that the values are much larger as compared to the random network with 1000 nodes.

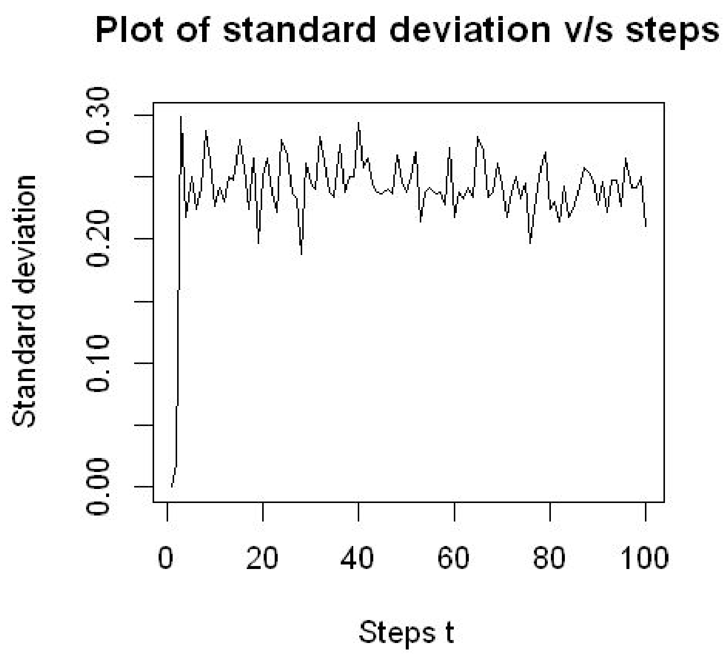




For 10000 nodes:

As seen from the plots below, the random network generated with 10000 nodes and 0.01 probability is a dense graph as the diameter is just 3. The value of average distance increases rapidly compared to previous cases. After that, the average distance converges.





From our analysis for 100, 1000 and 10000 nodes in the random network, we can say that diameter of the network does play an important role in the values measured for average distance and standard deviation.

It can be seen that the networks with small diameter converge much faster than the ones with larger diameter. Moreover, for networks with small diameter, the values of average distance are confined to a smaller range hence leading to a small standard deviation.

Thus, we can conclude that the network with 10,000 nodes converges fastest and has lowest standard deviation while the network with 100 nodes has larger values for both average distance and standard deviation.

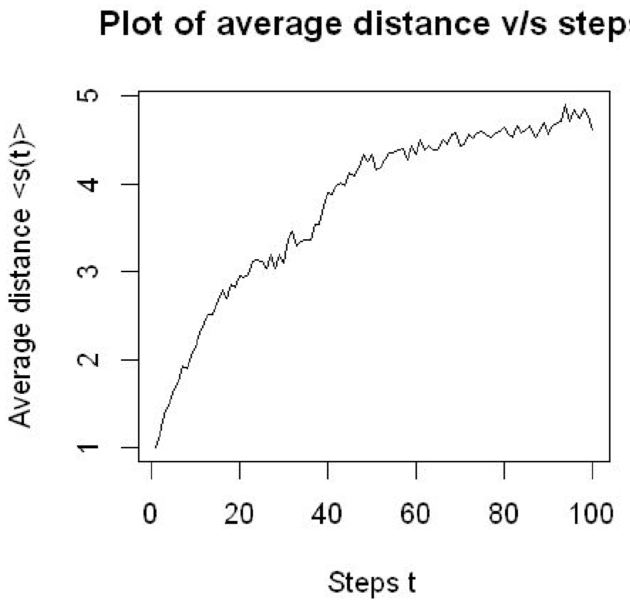
#Nodes	**Diameter**
100	10
1000	6
10000	3

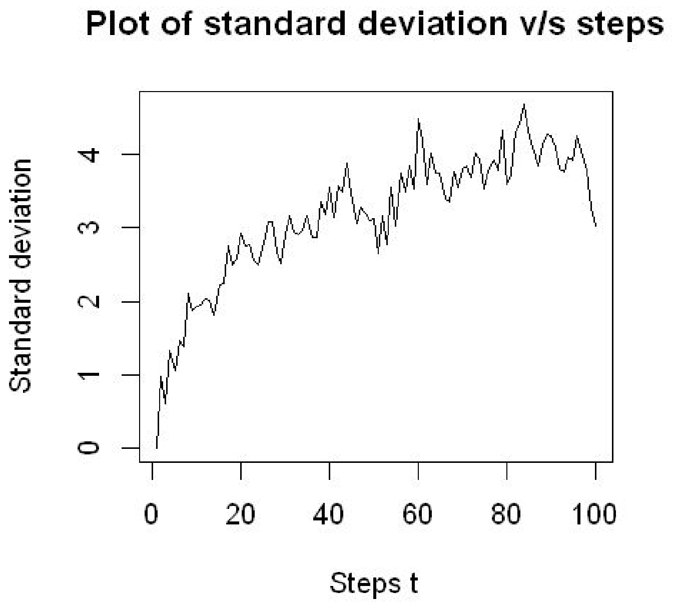
2.2 Random Walk on networks with fat-tailed degree distribution

2.2 (a) In this part, we work with preferential attachment networks which are also called fat-tailed degree distribution networks. We generated an undirected preferential attachment network using barabasi.game() consisting of 1000 nodes, where each new node attaches to 1 old node.

2.2 (b) In this part, we select a random node to start from and then the random walker begins a walk just like in the previous part. We then measure the distance of the walker at each time step from this starting node for time steps from 1 to 100. We repeat this procedure several times with different and random starting nodes to find the average distance, <s(t)> which is the shortest path length at each time step. We further measure the standard deviation of this distance using the formula mentioned earlier.

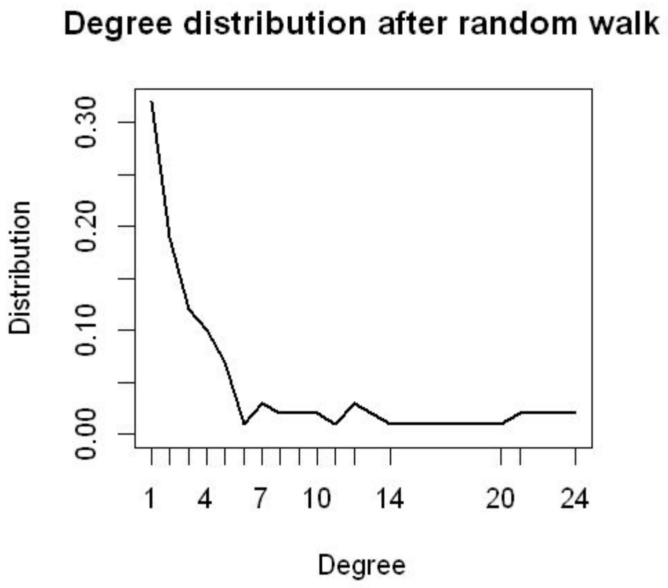
To observe the change in behavior of average distance and standard deviation with the steps, we plot both of these quantities against number of steps taken by the random walker.

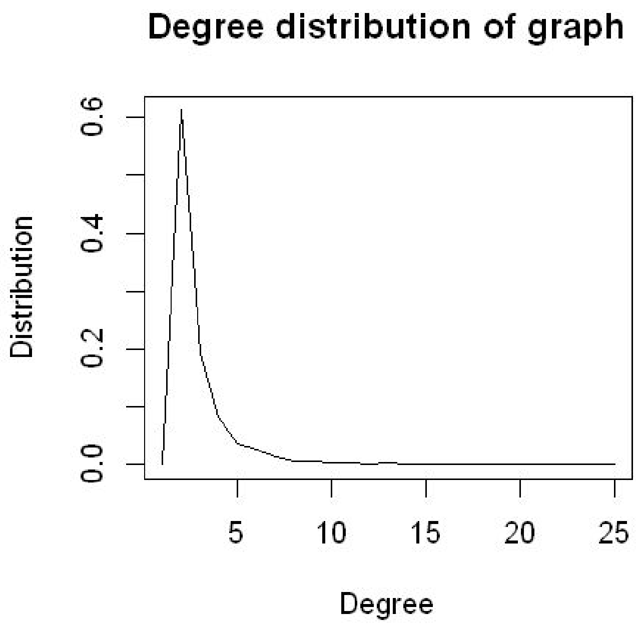




Observations:
As the random walker starts from the initial node and walks through the graph, the average distance of the node visited at each time step increases rapidly in the beginning and then it converges to 5. As seen from the plot above, the standard deviation is in the range of 1 to 4. It eventually converges to 4.

2.2 (c) In this part, we measure the degree distribution of the nodes that were reached at the end of random walk. And then we compare it with the degree distribution of the graph.

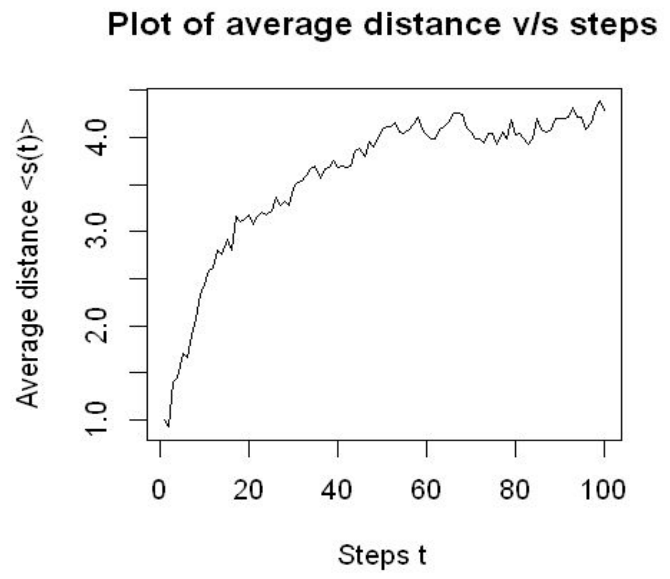


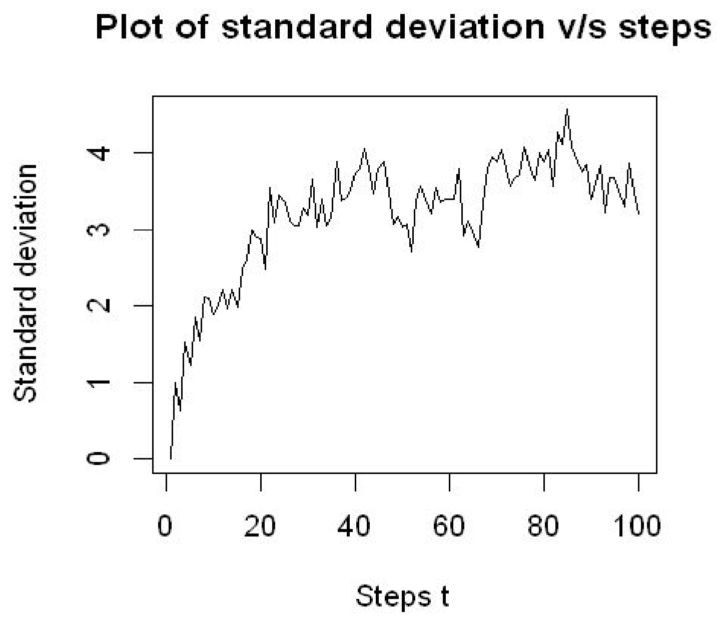


Observations:
From the plots of degree distribution of graph and the distribution at the end of the random walk, we can say that the overall distribution is quite similar. Moreover, the distribution is somewhat resembles to exponential distribution. We can also say that most of the nodes have degrees less than 5 and there are very few nodes with higher degree values.

2.2 (d) We now repeat the above calculations for 100 and 10000 nodes in the random network.

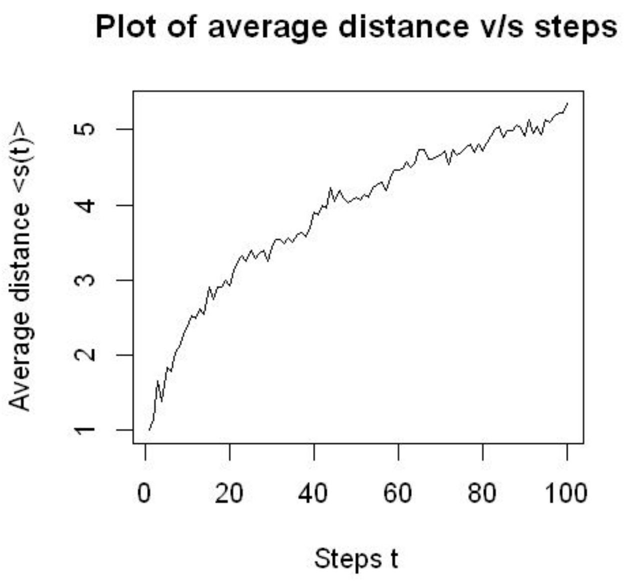
For 100 nodes:
As seen from the plots below, the value of average distance goes up to 4 for 100 steps. If we observe the plot of standard deviation, we can see that the values are very similar to the random network with 1000 nodes.

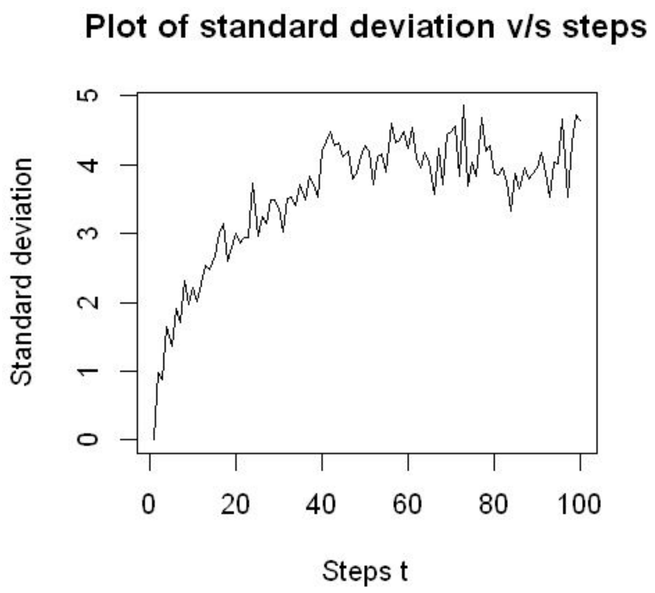




For 10000 nodes:

As seen from the plots below, the random network generated with 10000 nodes has average distance that converges slower compared to previous cases. The standard deviation is similarly distributed as in the previous cases.





From our analysis for 100, 1000 and 10000 nodes in the random network, we can say that diameter of the network does not play a major role in the values measured for average distance and standard deviation.

Although it can be observed that the networks with small diameter converge somewhat faster than the ones with larger diameter. The overall shape and characteristics of corresponding graphs is very similar in nature.

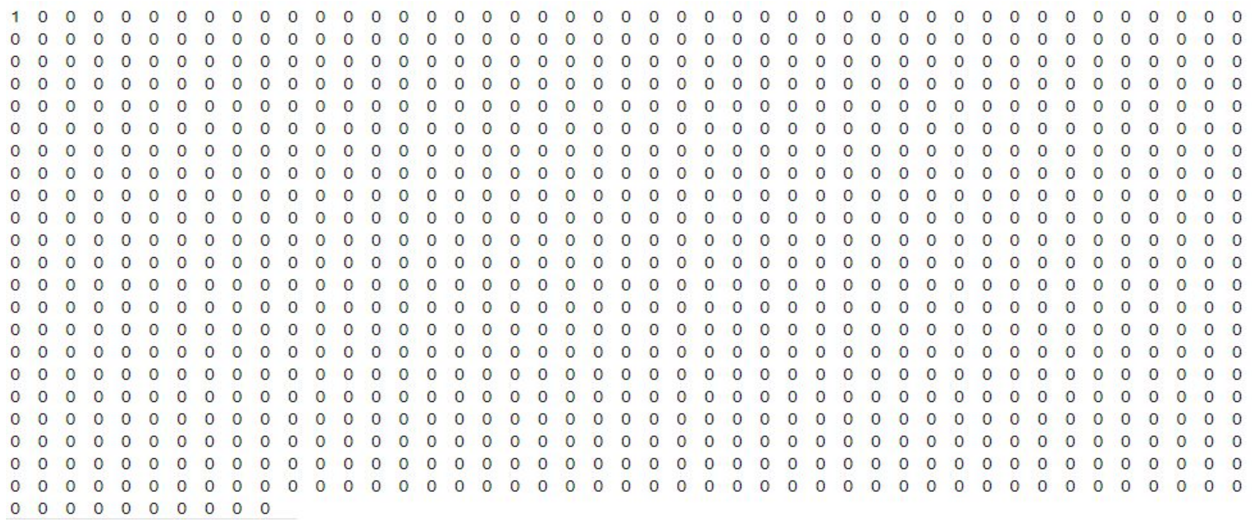
#Nodes	Diameter
100	11
1000	24
10000	30

2.3. PageRank

The PageRank Algorithm was devised by Google and is named after ‘Larry Page’ who is one of the founders of Google. The primary idea was: ‘A page is important if it is pointed by many important pages’. Thus, according to Google, PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

In this section of the project, instead of using the inbuilt page_rank function of R, we simulate our own PageRank algorithm. We do this with the help of random walk.

2.3 (a) Firstly, we create a directed random network with 1000 nodes using the preferential attachment model with m=4. We do a random walk on this network. The random walk is done 100 times and each random walk comprises of 100 steps. The probabilities of visiting each node is as follows:



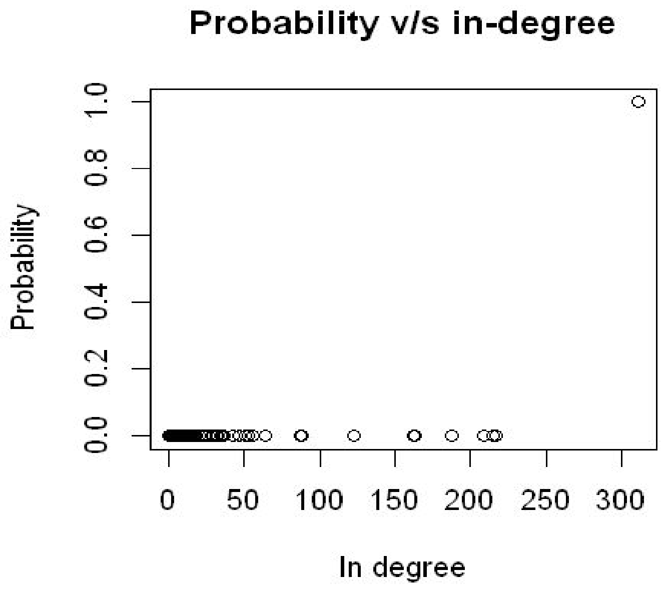
```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Analysis:

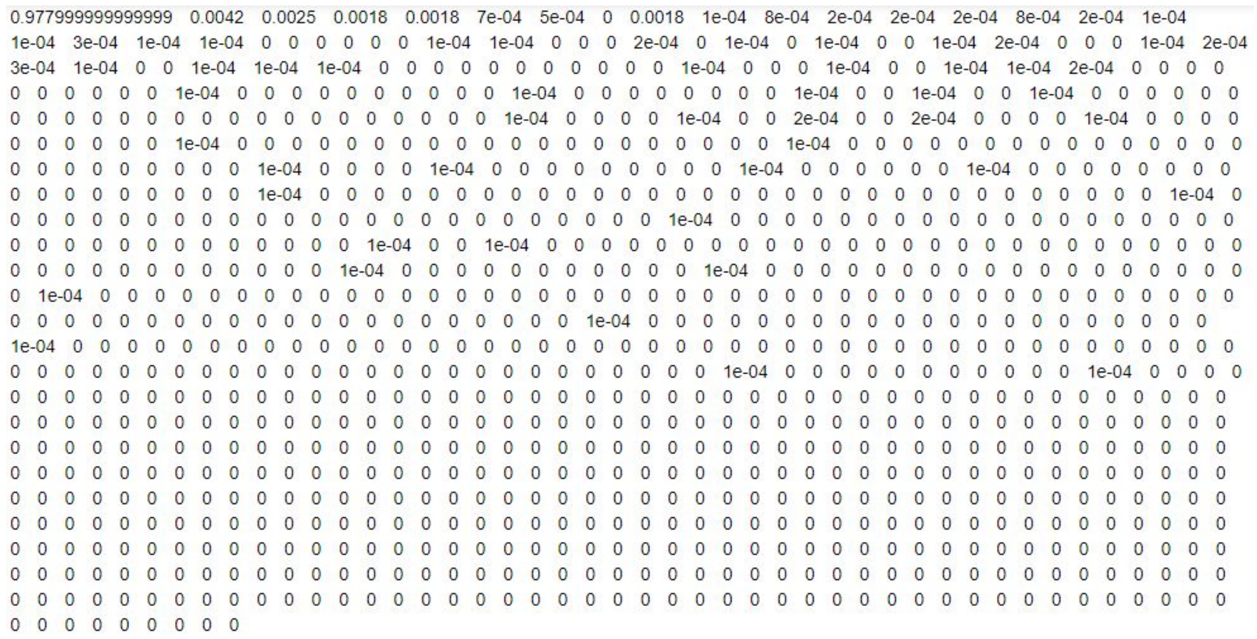
After each random walk is performed, we observe the last node which is visited in that particular walk. This last node for every random walk is always seen to be the first node of the preferential attachment model created. We count the number of visits of the last node in every random walk and take average. Since, every random walk ends with the first node of the network, the probability of this node is always seen to be 1. Why is this so? While creating a random network, we created a preferential attachment model with m=4. This essentially means that during the creation of the network, any newly added node will always be connected to the first node, either directly or with the help of a path(one or more nodes). Such a connection is always possible because of the fact that this network is a preferential attachment model. This means that the nodes 2,3,4 and 5 will always have a direct connection to the first node (because m=4). After this, any new nodes added to the network will connect to the first node directly or indirectly.

Hence, given a considerable number of steps taken, a random walk will always end in the first node because that random node will always be connected to the first node and the first node does not have any out degrees. Thus, we get the probability as 1 for the first node. Likewise, since the random walk does not end with a node other than the first node (provided that we have considerable number of steps in our random walk), the probability of all the other nodes is 0.

The following plot also stands as a witness to the above mentioned analysis which says that only one node has a probability of 1 and the rest have a probability of 0. This node is the first node of the network with an in-degree of 311. Therefore, we can safely state that the probability is related to the degree of the node.



We then tried another approach of simulating PageRank using random walk. In this, instead of counting the visit to the last node, we counted the visits to all the intermediate nodes too. The probabilities of visiting every node is observed to be as follows:

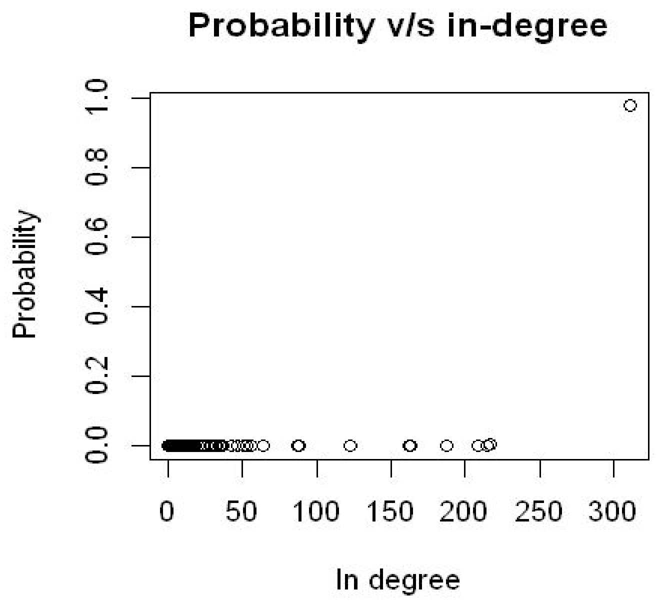


```
0.977999999999999  0.0042  0.0025  0.0018  0.0018  7e-04  5e-04  0  0.0018  1e-04  8e-04  2e-04  2e-04  2e-04  8e-04  2e-04  1e-04
1e-04  3e-04  1e-04  1e-04  0  0  0  0  0  0  1e-04  1e-04  0  0  0  2e-04  0  1e-04  0  1e-04  0  0  1e-04  2e-04  0  0  0  1e-04  2e-04
3e-04  1e-04  0  0  1e-04  1e-04  1e-04  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  1e-04  0  0  1e-04  1e-04  2e-04  0  0  0  0
0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  1e-04  0  0  1e-04  0  0  1e-04  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  1e-04  0  0  2e-04  0  0  2e-04  0  0  0  0  1e-04  0  0  0  0
0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
1e-04  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0  0  0  0  0  0  0  0  1e-04  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0
```

Analysis:

Here, too, we observe that the probability of the first node is highest and is almost equal to 1. Intuitively, this is because of the fact that every node is connected to the first node, hence the probability of visiting this first node will always be higher than that of any other node. As a result, this first node will be visited in every random-walk. As compared to the previous observation, here, there are also some very negligible (close to 0) probabilities found for some of the nodes. This is because of the fact that in this approach, we are counting the number of visits to all nodes and not only the last node. Because of the meagre number of visits to some of the intermediate nodes in the network (which finally end up connecting to the first node only), we get an infinitesimal probability of visiting those nodes. Since the total probability needs to sum up to 1, the probability of visiting the first node (with in-degree 311) reduces by a meagre amount which is equal to the probabilities of visiting other nodes in the network.

The following graph depicts the above analysis:



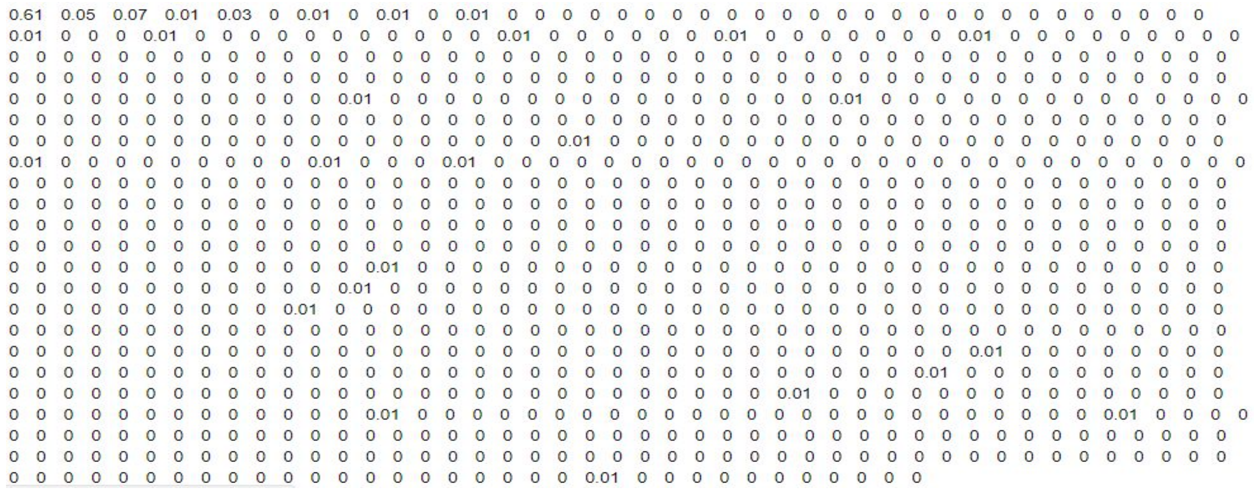
2.3 (b) Random walk with transportation

In this part, in order to compute the probabilities of node visits, we consider the teleportation probability. Here, alpha = 0.15. Thus, the probability of visiting a node is modified as follows:

P = (1-alpha) * P' + alpha/N

In this equation, P' denotes the probabilities of visiting each node as obtained from the transition matrix. The second term in the above equation denotes the probability of jumping to that node. This is equal to the alpha multiplied by 1/N which is the probability of having a random jump or a random visit to that node. We multiply by 1/N because we know that the probability of visiting each node is the same for all nodes. This means that even if the P' value of a particular node is 0, there will be a slight probability of visiting the node which is equal to alpha/N. In real life networks, having such a teleportation probability is useful in scenarios in which a network encounters dead ends or crawler traps.

The probabilities seen after performing this random walk are as follows:

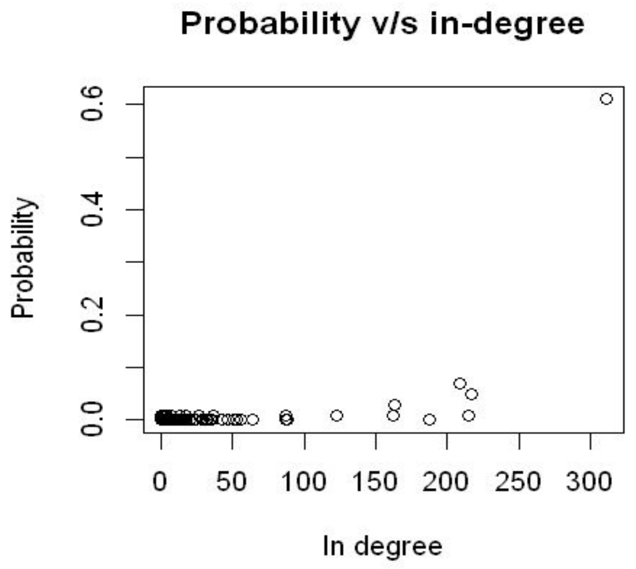


```
0.61 0.05 0.07 0.01 0.03 0 0.01 0 0.01 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.01 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.01 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0
```

Analysis:

We see that the probability of the first node with the highest in-degree is still the highest: 0.61. However, it has decreased considerably and is not tending to (or nearing to) 1. Intuitively, this is because of the teleportation probability introduced. If for a particular node, though the actual probability of visiting the node is 0 in the transition matrix, there could be a random jump to that node. This means that the probability of the nodes except the first node can increase by a small amount (alpha/N). Since the sum of probabilities should equal 1, the probability of visiting the first node decreases.

Here, we calculate the probability of visiting only the last node and not the intermediate nodes visited during the random walk. Hence we see that the probability of most of the nodes still remains 0. However, if we consider the probabilities of visiting all the nodes during random walk, we observed that the least probability was now alpha/N (0.15/1000 in our case) and not 0. This verifies our argument stated above. The following graph shows the probability of visiting a node v/s the degree of the nodes. We observe that the first node still has the highest probability. The next higher probabilities are observed for the nodes with largest in-degrees smaller than the first node which is intuitively correct.



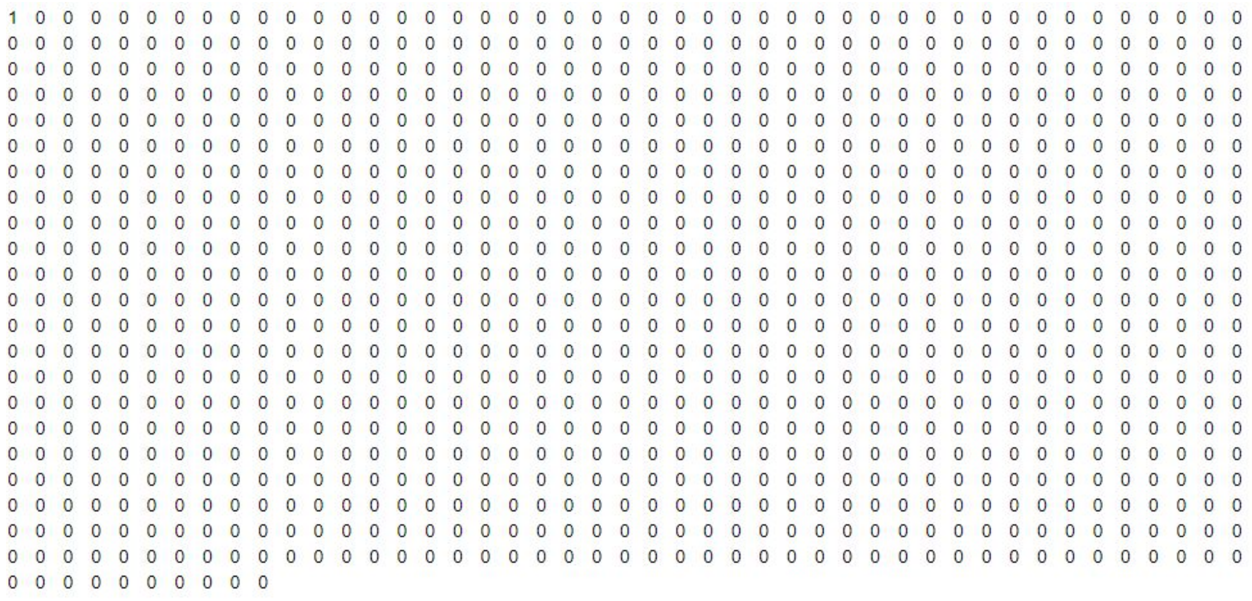
2.4. Personalized PageRank

We now work towards computing a personalized PageRank using random walk.
2.4 (a) Personalized PageRank is same as PageRank other than the fact that the random jump to a particular node is based on the probability of visiting that node. Thus, instead of using the teleportation probability of 1/N which states that every node has equal probability to be visited, we use the actual probability of visiting that node as the teleportation probability. Let's say that the actual probability of visiting a particular node is P (i.e. the PageRank) and let the probability of visiting it according to the transition matrix be P'. Then the actual probability of visiting that node (Personalized PageRank score) will be:

P_new = (1-alpha)*P' + alpha*P

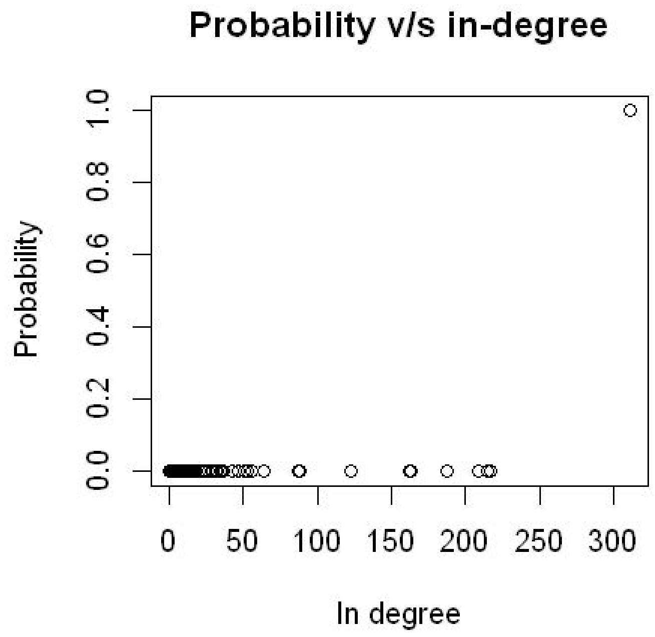
where P(PageRank) is the teleportation probability in this case.

The probabilities of visiting each node are as follows:

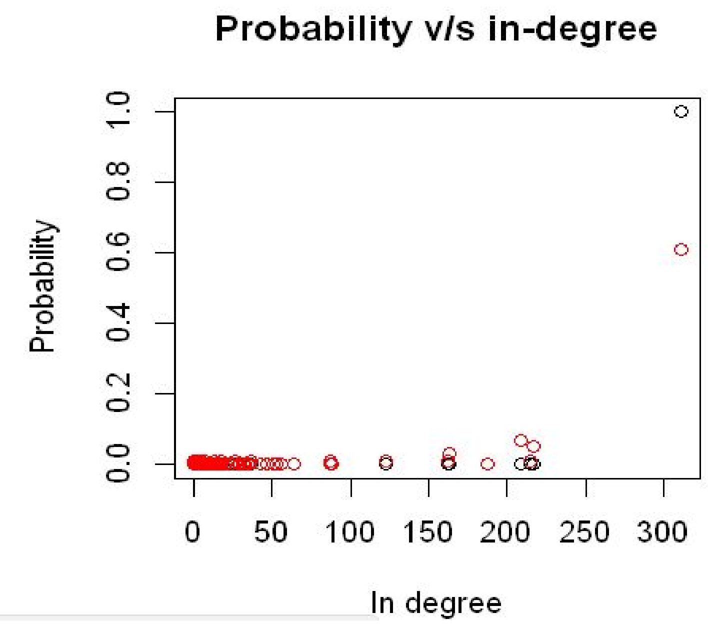


```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

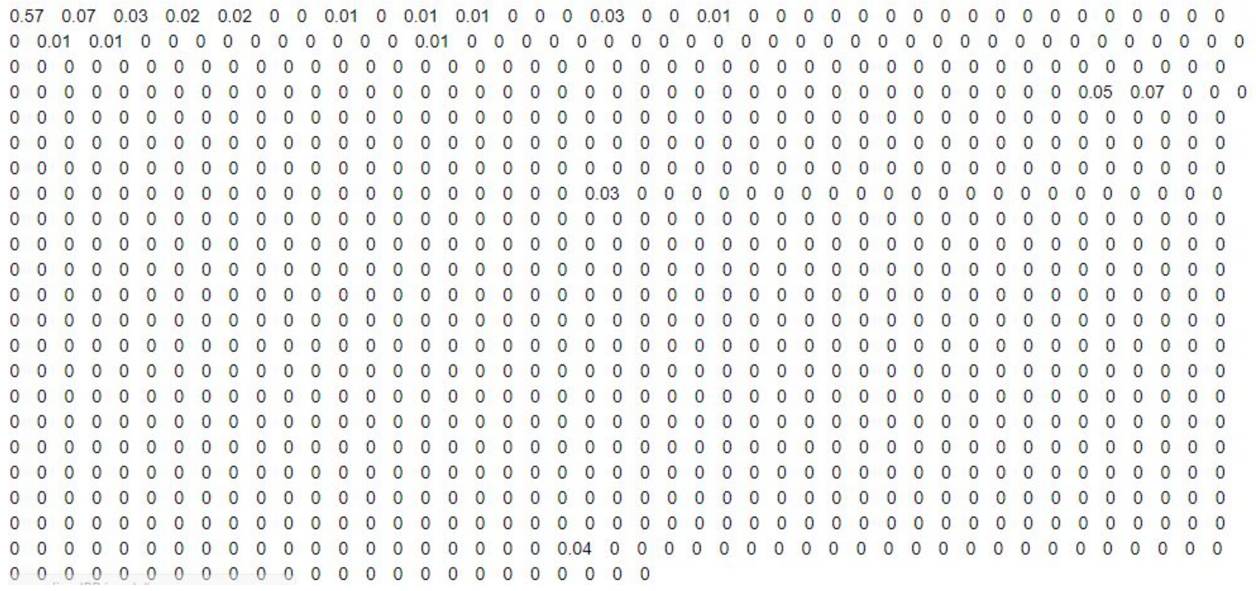
Analysis:
We observe that the probability of visiting the first node with the highest in-degree (of 311) is the highest with a value of 1. The probabilities of visiting all the other nodes is 0. Here too, we consider the probability of visiting the last node of a random walk and not that of visiting all intermediate nodes. So now why is the probability 0 even if we introduced alpha=0.15? It is because of the fact that the teleportation probability is now the probability of visiting every node as obtained from Q2.3.a (i.e. the PageRank). In Q2.3.a, we observed and verified that the probability of visiting any other node except the first one is 0. Hence, in the above formula, the P will now be 0 instead of alpha/N for nodes which cannot be reached from the current node. Because of P becoming 0, the second term is now redundant. Thus, we see that the probability of visiting a node with teleportation probability proportional to the node's PageRank is same as the probability of visiting the node without any teleportation probability. Here, we denote this as 'Personalized PageRank' (PPR). Therefore, the graph obtained in this case is the same as that of Q2.3.a.
PPR can also be called as a 'Trust Rank' in which a page is trusted only if it is linked by what you trust. This is same as the PageRank except for the random jump probability term (second term in our equation). The random jump factor is alpha*P is the page is trusted and 0 if it is not. Here, P stands for PageRank of the page. By relating this to our example, we understand that when P=1 (as per Q2.3.a), the P_new will include the second term. But, when P=0, that page is not trusted because it does not have a PageRank value hence we just not randomly jump to that page. In real life, this concept helps to solve the link-spam problem. PageRank can be vulnerable to link spamming if suppose a spammer creates a lot of pages and links them to a single spam page. This can be avoided by using Personalized PageRank or Trust Rank which ensures that good pages do not point to spam pages.



We now compare the above graph with that obtained in Q2.3.b. The red points in the graph below depict the probabilities as observed in Q2.3.b whereas the black points depict the probabilities as observed in Q2.4.a. As per our analysis of these two questions, we know that the highest probability of the first red node is less than 1 and the other red nodes may be slightly greater than 0 because of the teleportation probability 1/N which considers every node can be equally visited. For the black nodes (Q2.4.a), this probability is replaced by the PageRank as obtained in Q2.3.a which means that we trust only pages with good PageRank, hence we see that the same first node with the largest in-degree has now the highest probability which is equal to 1.



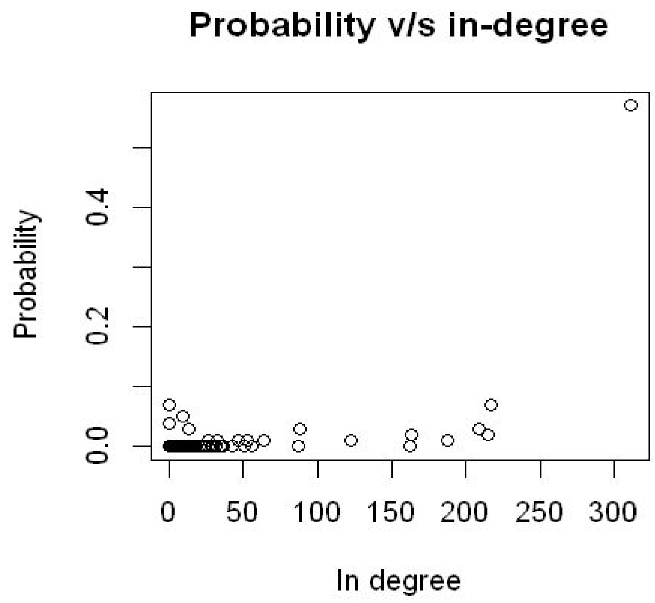
2.4 (b) We now randomly selected two nodes with median page ranks. Since every node except the first node have probabilities as 0, that means nodes with median page ranks could be any nodes except the first one. We then set the teleportation probabilities of these two nodes as 0.5. These two nodes which we picked up randomly using the sample function were node number 953 and 165. The teleportation probabilities of all other nodes were set to 0. We have used alpha as 0.15 as in part a of this question. The PageRank value in this case were as follows:



```
0.57 0.07 0.03 0.02 0.02 0 0 0.01 0 0.01 0.01 0 0 0 0.03 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0.01 0.01 0 0 0 0 0 0 0 0 0 0 0.01 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.05 0.07 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.03 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.04 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

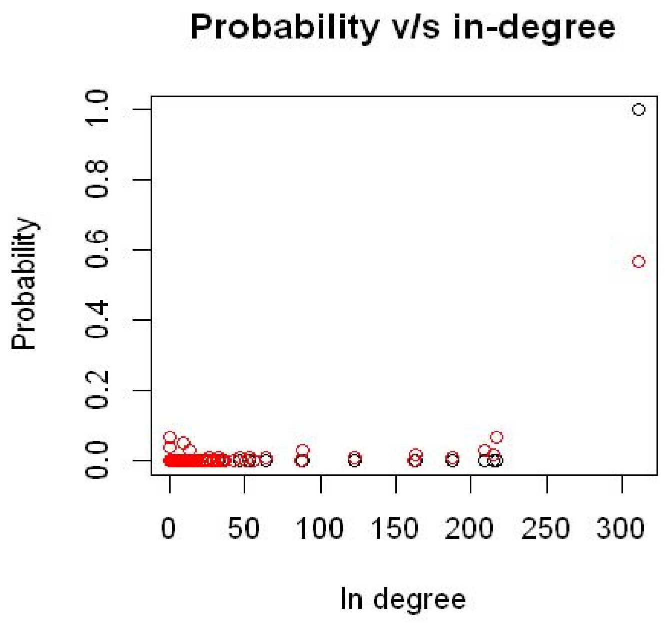
Analysis:

Here, we see that the first node still has the highest probability of 0.57. The fact that we have given 0.5 probabilities to nodes 953 and 165 means that we deliberately want them to be visited irrespective of what their PageRank is. These nodes now have probabilities of 0.04 and 0.07 respectively. This is because of the teleportation probability given to them. Now, we see slightly greater than 0 probabilities in some of the other nodes because these are the nodes which are reachable from either node 953 or 165. Because 953 and 165 have slightly more than 0 probabilities, the probability of landing up on a node which is connected to them will not be 0. Since the sum of all probabilities have to be equal to 1, the probability of the first node decreases. The change in PageRank values can be explained by this reason.



Comparison:

We now compare this (red points) with the PageRank of Q2.4.a (black points) where we used Personalized PageRank. In Personalized PageRank, we got the PageRank of first node as 1 and 0 for all other nodes. This was because a personalized PageRank with teleportation proportional to PageRank values is as good as having no transportation. However, in this question (Q2.4.b), we deliberately set the teleportation of two nodes with median PageRanks to 0.5. Which is why the PageRank values of some of the nodes increased by a small amount and a corresponding decrease of PageRank was found for the first node. This can be easily verified from the following graph.



2.4 (c) Effect of Self-reinforcement on PageRank equation

According to the research paper which presents Google[1], the PageRank of a page A which has pages T1….Tn pointing to its is given as :

PR(A) = (1-d) + d(PR(T1)/C(T1) + … + PR(Tn)/C(Tn))

In this equation, C(A) is defined as the number of links going out of page A and d is the damping factor. However in this equation, if we set the teleportation to be 1/N, it would mean that every node would have equal probability to be visited. The PageRank equation would then be:

PR(A) = (1-d)/N + d(PR(T1)/C(T1) + … + PR(Tn)/C(Tn))

However, in this section, we need to find the effect of self-reinforcement on PageRank equation. Thus, we need to change the teleportation probability to something which is proportional to the PageRank. This can be achieved by changing the teleportation probability of 1/N to P(A) i.e. the PageRank of A itself. Thus, the equation will be changed to:

PR(A) = (1-d)*PR(A) + d(PR(T1)/C(T1) + … + PR(Tn)/C(Tn))

This means, while calculating PageRanks, we are taking into consideration only trusted pages. Also, make sure the above PageRank equation is an iterative PageRank equation. Thus, the PageRank value of A will be used to compute the new PageRank value of A iteratively. Such is the effect of reinforcement on the PageRank of a page.

References:

[1] Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." Computer networks and ISDN systems 30.1-7 (1998): 107-117.