EE 232E Graphs and Network Flows
Homework 1

Professor Roychowdhury

Prepared by:

Arunav Singh (304 760 844)
Eric Goldfien (603 887 003)
Steven Leung (304 777 142)

NOTE: Questions 1, 2, and 4 are programmed in Python while Question 3 is programmed in R due to a function (aging.barabasi()) that we needed but could not be found in Python.

Question 1:

Part a:

Three undirected random networks with 1,000 nodes for three different probabilities (0.01, 0.05, and 0.1) of drawing an edge between two arbitrary vertices were generated using the *Erdos_Renyi(n,p)* function. The degree distributions for each of these networks are shown below in Figures 1.1-1.3.

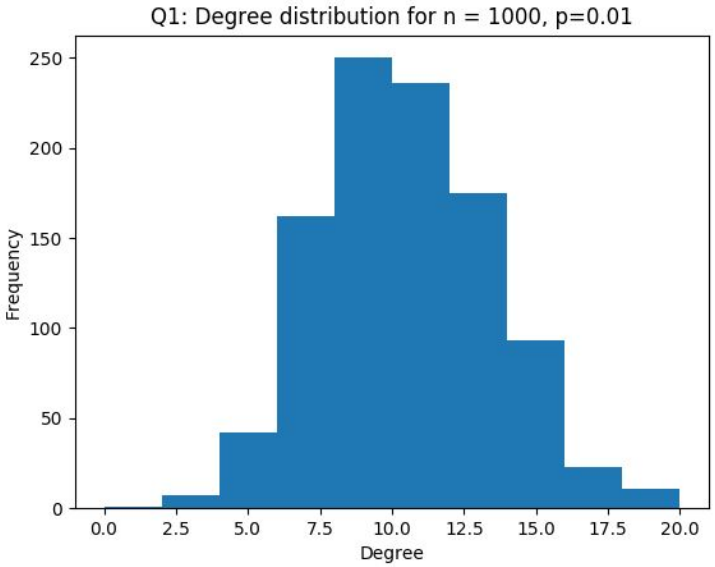

Figure 1.1 Degree distribution for (n = 1,000 & p = 0.01)

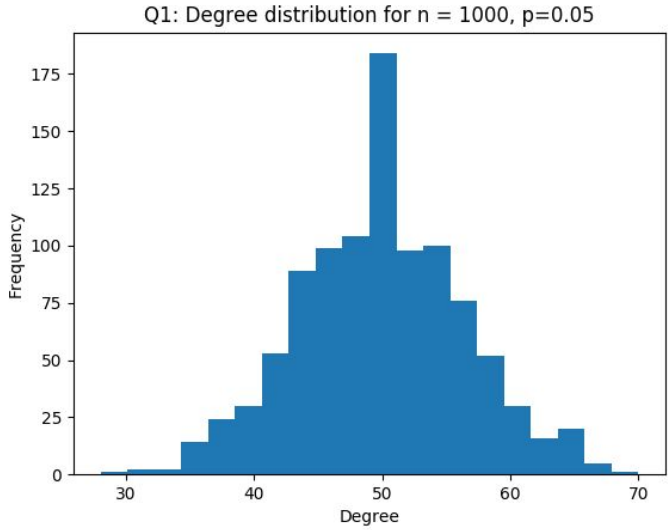

Figure 1.2 Degree distribution for (n = 1,000 & p = 0.05)

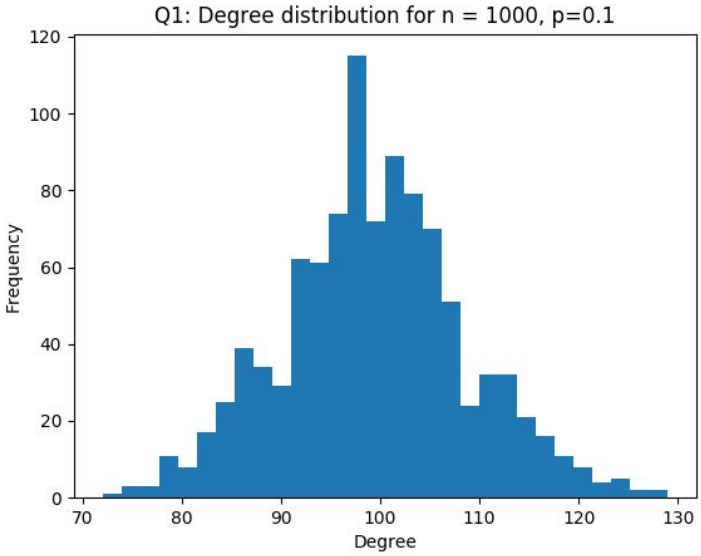

Figure 1.3 Degree distribution for (n = 1,000 & p = 0.01)

Part b:

100 different networks for each probability were generated and the probability that the networks are connected along with the average diameter of these networks were calculated. The purpose for averaging over 100 networks is that these networks are random so an expected value (mean) is sought. Note that the averaging of values will be performed through the remainder of the assignment for the same reason. The function *is_connected()* was used to check if a network is was connected and the function *diameter()* was used to calculated the diameter of these networks. *Table 1.1* summarizes the results.

Probability (p)	Probability that Network is Connected	Average Diameter
0.01	96 %	5.36
0.05	100 %	3.0
0.1	100 %	3.0

Table 1.1 Summary of Network Connectivity and Average Diameter

Part c:

The threshold probability such that the network is connected (defined as p_c) for values of $p > p_c$ and disconnected for values of $p < p_c$ was found by starting at a value of p=0 (which would result in a disconnected network) and then increasing the value of p by 0.001 until the network is connected. The value of p where this switch occurs will be p_c. Performing this experiment 100 times and averaging the values gives $p_c =$ **0.00773**.

Part d:

The value of p_c can be analytically derived for the Erdős-Rényi model using the following formula below. It can be seen that the numerical and analytical method for finding p_c result in a similar value.

$$p_c = \frac{\ln(n)}{n} = \frac{\ln(1000)}{1000} = 0.00691$$

Question 2:

Part a:

We used the Barabasi() function to generate our undirected network for 1000 nodes. The degree distribution is shown below in *Figure 2.1*. **The average diameter across 100 iterations was: 17.32.**

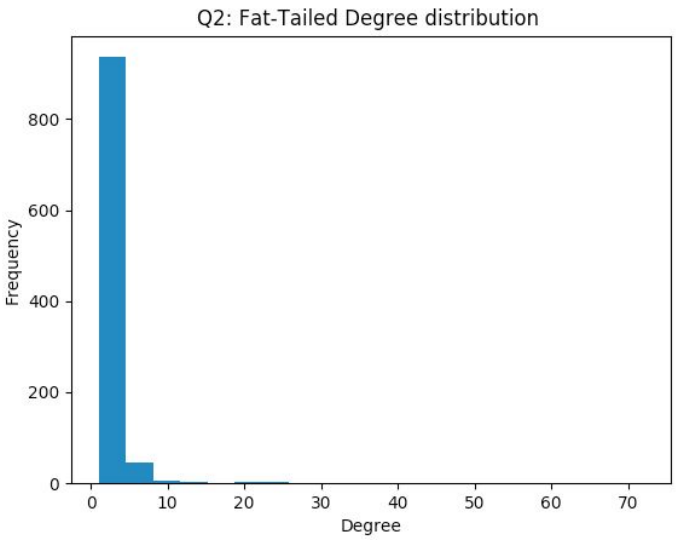

Figure 2.1: Degree Distribution for 1000 node network

Part b:

The probability of the network being connected across 100 runs was 100%. The giant connected component (GCC) is found by using the following line in python *community_fastgreedy().as_clustering().giant().* The GCC compared to the rest of the network is shown in *Figure 2.2.* The blue nodes correspond to the GCC while the red nodes correspond to the rest of the network.

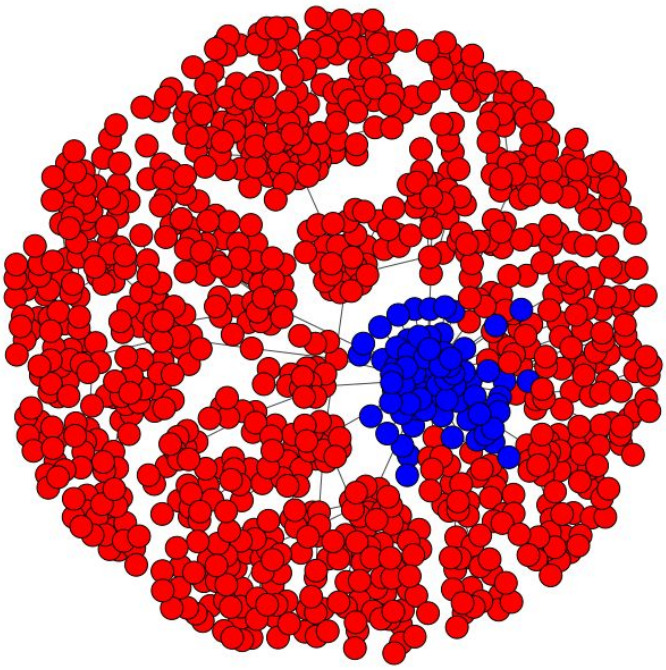

Figure 2.2 Giant Connected Component (in blue)

The community structure is found by looking at the number of communities that are generated from the fast greedy method. After performing this experiment 100 times, a distribution for the community can be plotted (*Figure 2.3*). **The average community size was 36.71.**

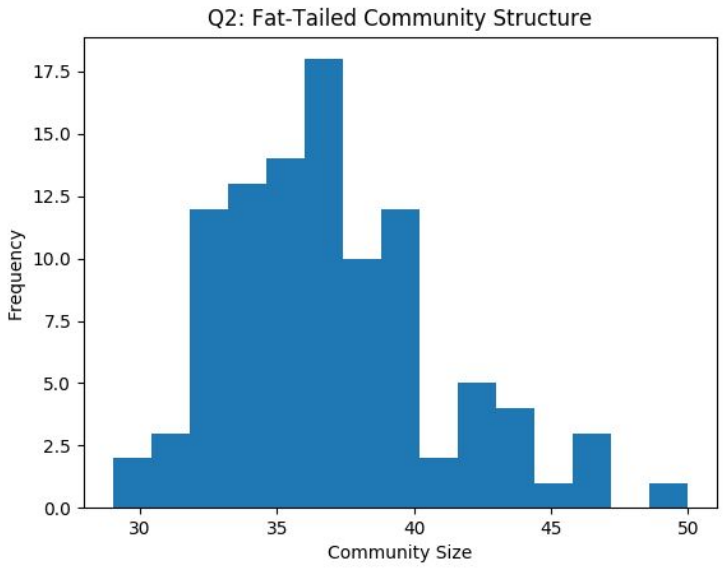

Figure 2.3 Fat-Tailed Community Structure

The average modularity over 100 independent trials was 0.9228. The reason for this high modularity is due to our use of the Barabasi model as new nodes get attached to higher degree nodes. This creates a densely populated regions with sparse connections between each of the regions, or communities, hence giving way to a high modularity. It can also be seen from the plot in *Figure 2.2* that nodes within communities are densely connected and connections between nodes of different communities are sparse. This is a result of a high modularity. Another way to see this is that independent communities can be easily identified in the graph.

The modularity can be found by using the *modularity(VertexClustering_obj)*. However, since the *community_fastgreedy()* function in python returns a VertexDendrogram object, it has to first be converted to a VertexClustering object using the *as_clustering()* function before using the *modularity* function.

Part c:

Generating a similar network this time with 10,000 nodes, **the average modularity over 100 independent trials was 0.9748**. This modularity is slightly larger than that of the smaller network.

Part d:

By picking a random node i and then a random neighbor j of that node, the degree distribution of node j is shown below in *Figure 2.3*. The plot was generated from running 500 iterations

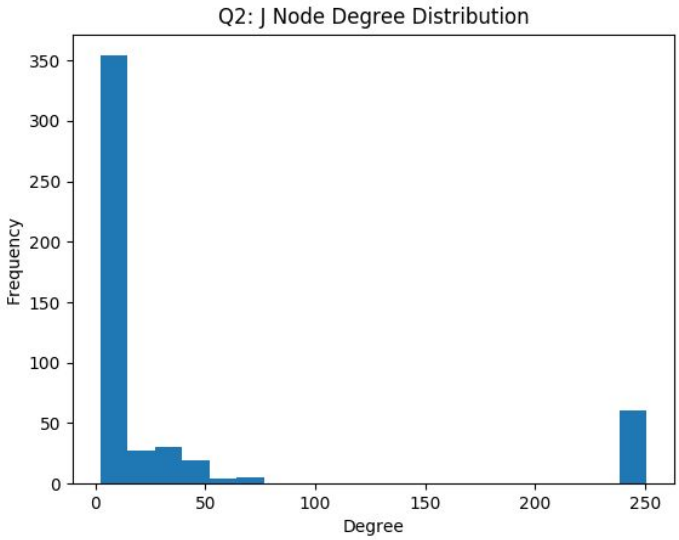

Figure 2.3 Degree Distribution of node j

<u>Question 3:</u> **NOTE: Question 3 was programmed in R**

Part a:

A random graph can be created from simulating its evolution using the *sample_pa_aging()* function. This function uses a Barabási–Albert (BA) model to generate a random graph using preferential attachment, when a new vertex is added to the network the probability it attaches to a previous vertex is based on how many edges the existing vertices already have, the vertices with the most edges and the oldest age are given the highest preference. For the network with 1,000 nodes, the degree distribution is shown below.

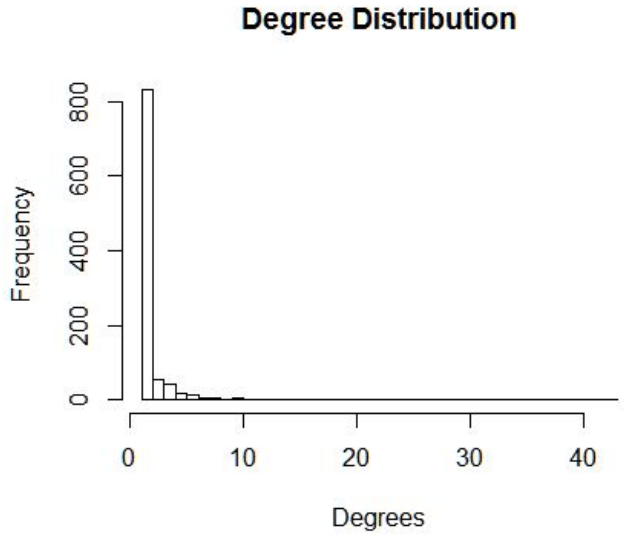

Figure 3.1 Barabasi Degree Distribution (n = 1,000)

Part b:

The community structure was found using the *fastgreedy.community()* function. The plot of the community structure was found by plotting the *membership()* of the community structure generated by fastgreedy. From the graph below you can see there are 35 different communities with the largest community having over 100 vertices.

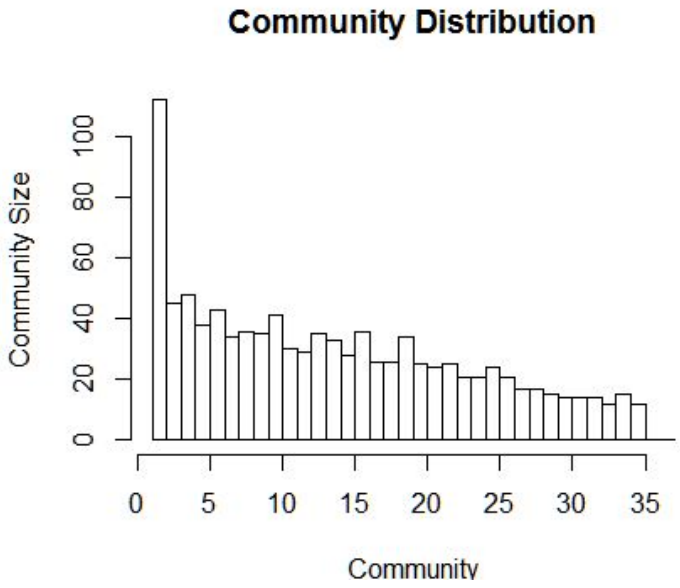

Figure 3.2 Community Structure from Fast Greedy Method

The modularity for a 1000 node Aging Barabasi network is 0.9315752.

The figure below shows one of the Barabasi networks. It can be seen from the plot that nodes within communities are densely connected and connections between nodes of different communities are sparse. This explains why the modularity for this type of network is high. Another way to see this is that independent communities can be easily identified in the graph.

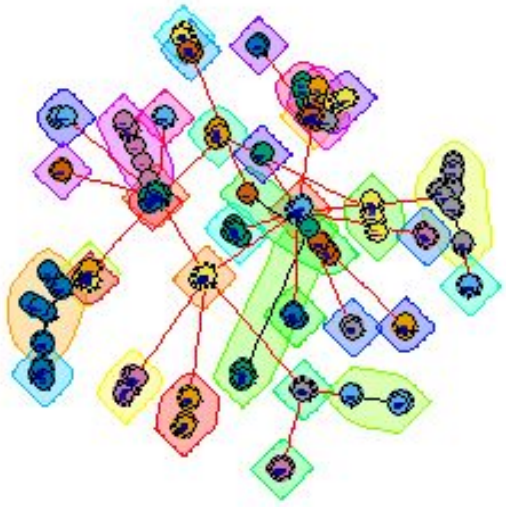

Figure 3.3 Plot of Aging Barabasi Network

Question 4:

Part a:

A directed network created from the forest fire model can be achieved using the *Forest_Fire(n,fw_prob,bw_factor)* function. For our network, we decided on 1,000 nodes (n = 1,000), a forward probability of 0.4 (fw_prob = 0.4), and a backwards probability of 0.2 (bw_factor = 0.2/0.4). The In-Degree and Out-Degree distributions are shown in *Figure 4.1* and *Figure 4.2*

The in and out degree distributions are shown for a forward probability of 0.4 and backward factor of 0.2/0.4. Our code generates distributions for 5 different forward factors to show how changing the forward probability changes the degree distributions but for simplicity, only the case of 0.2 is shown below.

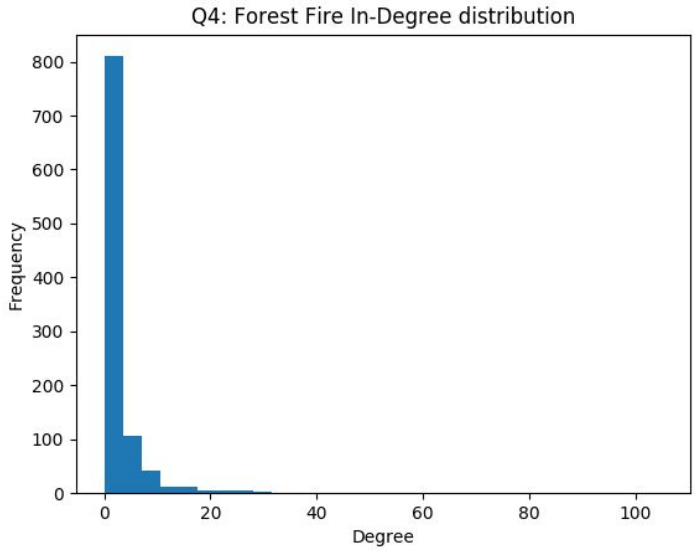

Figure 4.1 Forest Fire In-Degree Distribution (n = 1000, fw_prob = 0.4,bw_factor = 0.2/0.4)

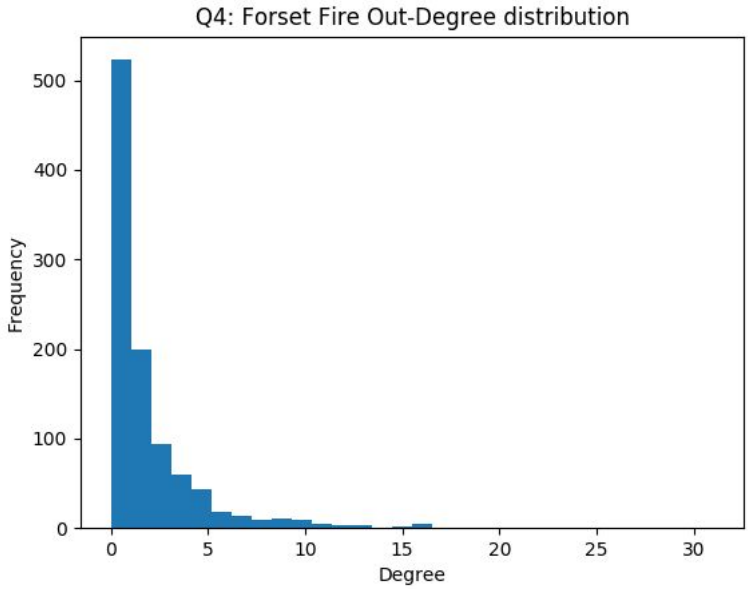

Figure 4.2 Forest Fire Out-Degree Distribution (n = 1000, fw_prob = 0.4,bw_factor = 0.2/0.4)

Part b:

The average diameter was found by averaging diameter measurements over 100 created networks which resulted in an average value of 10.04.

Part c:

To use the *community_fastgreedy()* function to find the community structure, the network first has to be converted from a directed network to an undirected network using the *as_undirected()* function. Then the functions *community_fastgreedy(), modularity(VertexClustering_obj)*, and *subgraphs()* can be used similar to Question 2. Upon performing 100 independent trials, the distribution of the number of communities in the community structure is shown below in *Figure 4.3*. **The average community size is 43.55 vertices.**

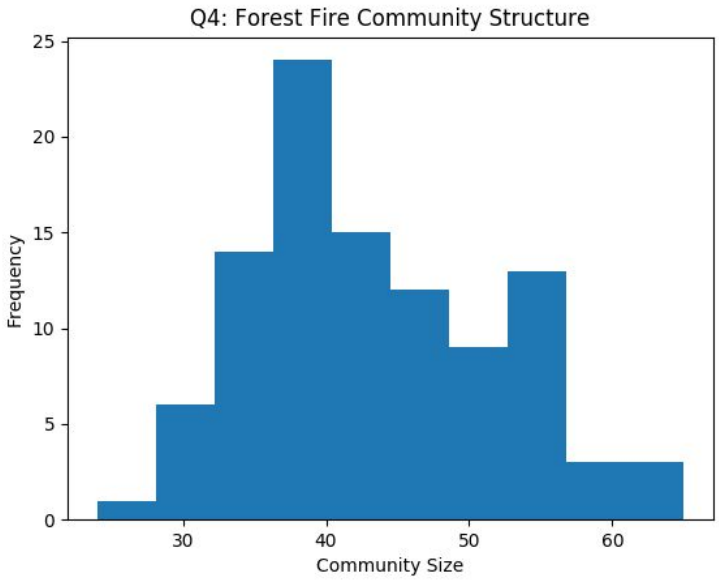

Figure 4.3 Community Structure of Forest Fire Model

The average modularity over 100 independent trials is 0.7013. From figure 4.4 below, it can be seen that the connections between nodes within communities and between nodes of different communities have a similar density. This explains why the modularity is not as high as that of question 2 and 3. Going back to the example in question 2, it is seen from the graph that independent communities of the graph cannot be easily identified.

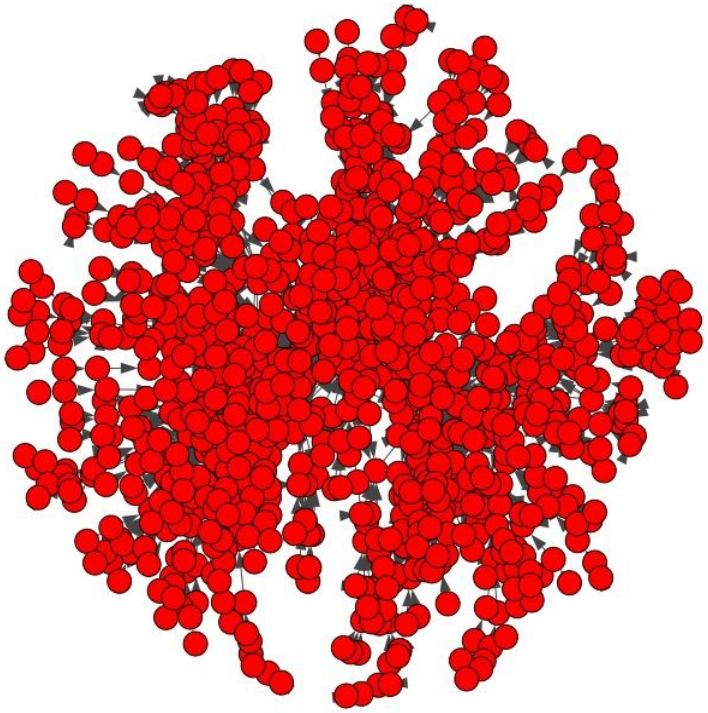

Figure 4.4 Plot of Forest Fire Graph N=1000