Chandler Stone
Philip Brown
CS 4740.001
10 February 2022

Project 1

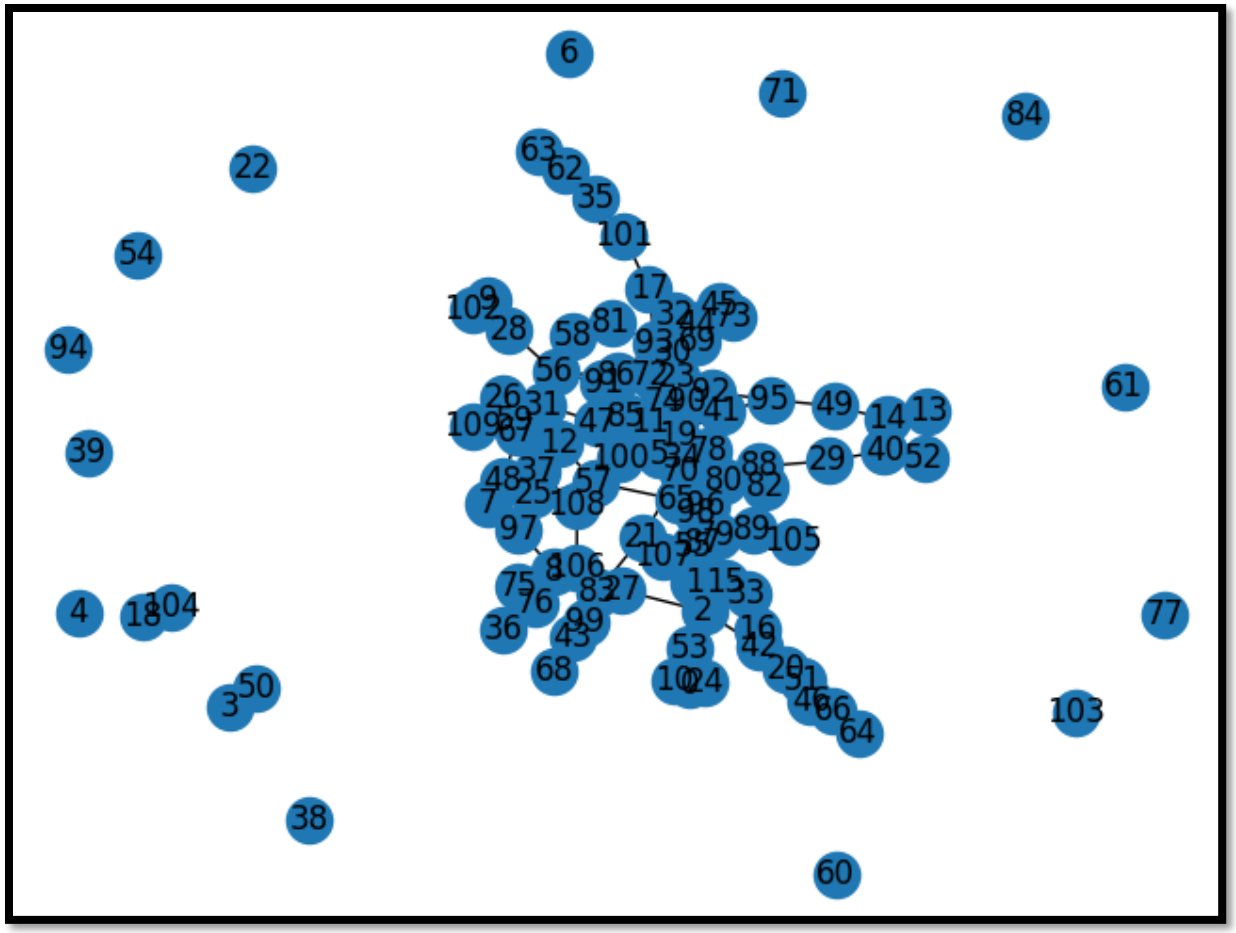
Deliverable 1:

I decided to use Python and NetworkX for my project. I defined my method as randomGraph with input variables of n and p. First, I created a new graph with n number of nodes. Next, I ensured that the input p value was between 0 and 1. Last, I iterated through all of the possible edges and added the edge if the p-value was greater than a randomly generated number.

Code:

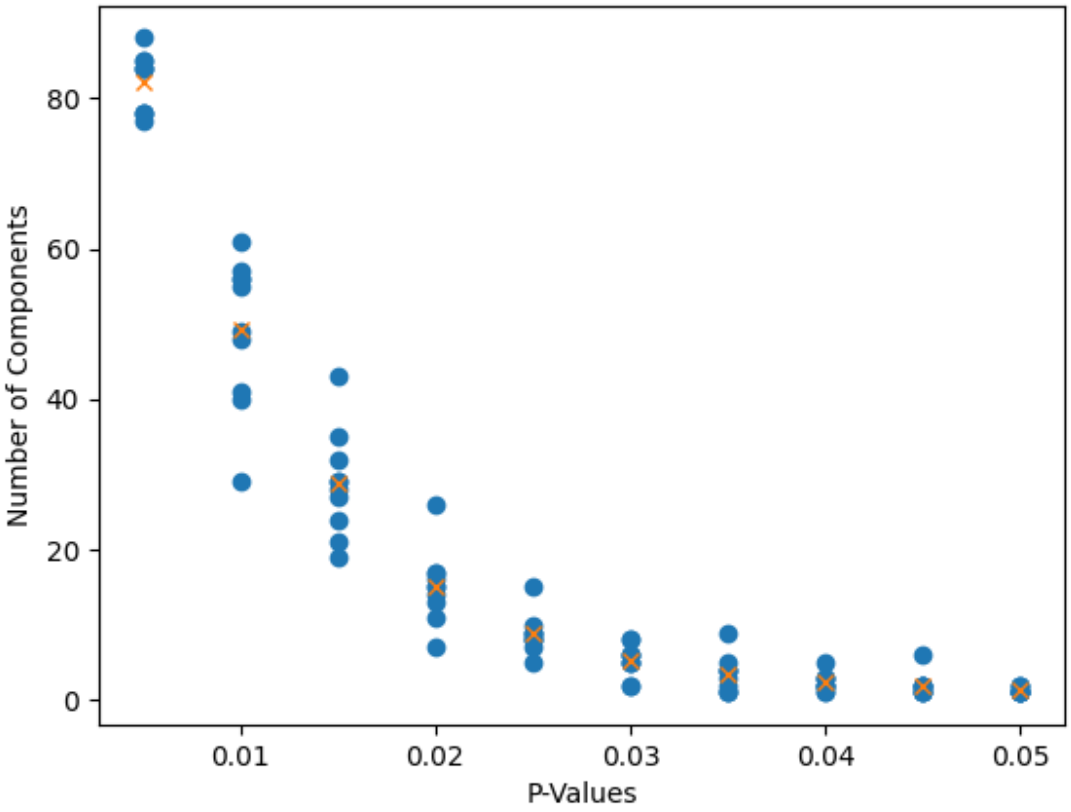
```
15  #PART 1
16
17  def randomGraph(n, p):
18
19      G = nx.Graph()
20      G.add_nodes_from(range(n))
21
22      if p <= 0:
23          return G
24
25      if p >= 1:
26          return nx.complete_graph(n, create_using = G)
27
28      edges = itertools.combinations(range(n), 2)
29
30      for i in edges:
31          if np.random.rand() < p:
32              G.add_edge(*i)
33
34      return G
35
36  checkGraph = randomGraph(110, 0.02)
37  nx.draw(checkGraph, with_labels=True)
38  plt.show()
```

Example plot with n = 110 and p = 0.02:



Deliverable 2:

I continued using n = 110 for the number of nodes. I chose to use p-values of 0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05. Each p-value has ten randomly generated graphs. As shown in the plot below, as the p-values get larger, the number of components decreases. This makes sense because when p is larger, the graph is more likely to be completely connected. The number of components is basically 0 when p is around 0.04 – 0.045. The averages are marked with the orange x.



Deliverable 3:

Last, I continued using n = 110 for the number of nodes. I reused the p-values of 0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05. Each p-value has ten randomly generated graphs. As shown in the plot below, as the p-values get larger, the average degree value increases. This makes sense because when p is larger, the graph is more likely to be completely connected. Thus, more connections come with more neighbors. The overall averages are marked with the orange x.