Nicholas Manfredi
110207186
MAT331: Project 3 Report

The algorithm that was designed to compute and display the connected components in problem (3) took approximately three days to implement and function properly. The algorithm would initially generate a random graph, *G* with a total of *n* vertices. These vertices would then be connected with edges, appearing with a probability *p*. Using the python package, **networkx**, I was then able to visually represent these graphs. This algorithm then, takes in graph G and returns a list of the components that are connected. The algorithm is then represented through a list showing which vertices are connected with one another. As the value of *n* and the number of edges increase, the running time scales exponentially. The time for *n = 10*, is approximately less than a minute, n = 50 is approximately two to three minutes, n = 100 takes approximately 20 minutes, n = 250 is approximately 40 minutes, n = 500 was completed in approximately an hour, and n = 1000, taking the longest, was approximately completed in two and a half hours.

For the optimizations made in part (4), initially the original algorithm was going through and visiting each components of the randomly generated graph and checking if the point was visited. If the point was not visited it would then add and store each component manually into a list. The optimized code uses the **sort** function. This function then takes in the list of components and sorts the stored numerical values in descending order. This shortens the run-time, meets the run-time condition, and increases the efficiency of the algorithm.

The kind of transition that occurs around *p = 1/n* is described as a *sharp phase transition*, for any constant $\varepsilon > 0$ and $p \leq \frac{1-\varepsilon}{n}$, then with high probability all connected components of the graph have the size O(log *n*) there are no giant components, while for $p \geq \frac{1+\varepsilon}{n}$ there is with high probability that there is a single giant component, with all other components having size O(log *n*). Testing random graphs with a very large *n* shows that the number of vertices within the largest components of the graph is with high probability proportional to the given *n*.

The results from part (6) display within the plot the ratio of sizes between the giant component and the second largest component. It shows the difference in size. It shows that when experiencing the transition around *p = 1/n* that, when below the threshold, the giant component in not significantly larger than the second largest component. Additionally, while above the threshold, the giant component gets significantly larger than the second largest component. This plot displays the size of each of the given components for the given *n*.

For other properties we can explore, on a similar note we can continue to test the connectedness of a random graph through Percolation theory; characterizing the connectedness of random graphs, focusing on those that are infinitely large. Additionally, on a tangent we can continue to explore whether a graph is planar/non-planar. For example, if some graphs are isomorphic or bipartite. Additionally, we can also test for edges being Hamiltonian (i.e a

Hamiltonian Circuit). Finally, we can test the coloring of a graph as well as determines its chromatic polynomial.