Miles Honsberger

Project 1 Report

MAT 331

9/29/22

Question # 7

The main difference in this section of the project is that the probability value now depends on the quotient of the ‘t’ value divided by the ‘n’ number of total vertices. The value of ‘n’ is predetermined and remains constant whilst the value of ‘t’ starts from zero and increases to a defined max ‘t’ value by a chosen increment. Because the ‘values’ are naturally much greater than the ‘t’ value the ‘p’, probability, is much smaller with increasing ‘n’ values. For example, if ‘t’ was fixed to 1, the ‘n’ values of 20, 60, and 120 yield increasingly smaller ‘p’ values. Smaller p values then result in a lower probability of each vertex connecting to components. That is why in the graph with multiple n values where the t value remains at 4, that we see increasingly smaller average normalized sizes of the largest components compared to their respective t values.

```
expLargPlotter(20,4,0.1) #n=20 blue
expLargPlotter(60,4,0.1) #n=60 orange
expLargPlotter(120,4,0.1) #n=120 green
expLargPlotter(300,4,0.1) #n=300 red
expLargPlotter(600,4,0.1) #n=600 light purple
expLargPlotter(1000,4,0.1) #n=1000 dark purple
```

```
[<matplotlib.lines.Line2D at 0x7f117e7c2290>]
```

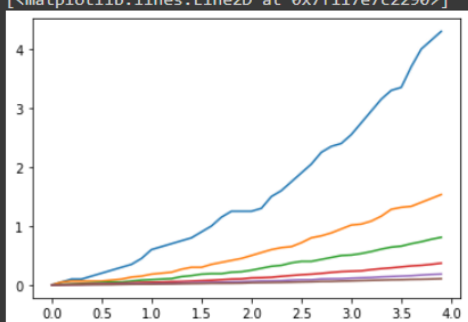

Figure 1: Plot of average normalized sizes, y-axis, vs t values, x axis. The smallest n values show a steep trend whilst the greater n values show a more shallow trend.

Assuming in an instance where the t value remains constant, or at 4 as in the previous test, and the n value is changed to a much greater value, 10,000 for instance, we would expect to see the plot show an increasing shallow trend. As we further increase the n value we see the trend become more shallow, approaching the x axis, that is to say that the average normalized values become closer to 0. It’s also worth noting that an exponential trend is slightly noticeable with n = 20. Upon further exploration with smaller values of n, we find that the exponential trend becomes more noticeable with smaller n values and that the curve straightens with higher n values.

```
expLargPlotter(5,4,0.1) #n=5 blue
expLargPlotter(10,4,0.1) #n=10 orange
expLargPlotter(20,4,0.1) #n=20 green
expLargPlotter(60,4,0.1) #n=60 red
expLargPlotter(120,4,0.1) #n=120 violet
expLargPlotter(300,4,0.1) #n=300 dark purple
expLargPlotter(600,4,0.1) #n=600 not distinguishable
expLargPlotter(1000,4,0.1) #n=1000 not distinguishable
```

```
[<matplotlib.lines.Line2D at 0x7f117e4e4410>]
```

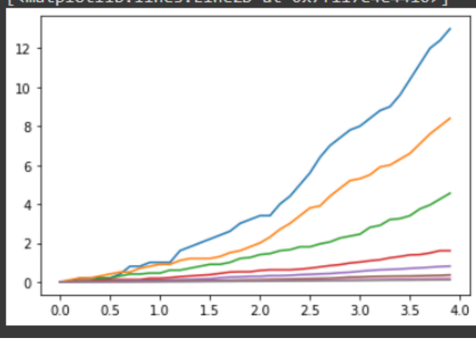

Figure 2: Plot of average normalized sizes vs t values. Done off-hand to explore the general nature of the trend

Here, n values of 5 and 10 were added to the plot, and in the lower values of n, a curve is much more noticeable on its trend line than higher values. Thus we can also conclude that the relation between the y and x values is of exponential nature.

Question #8

The nature in this second part of the experiment is dealing with ratios between the largest component and the second largest component. Since the adjacency lists are randomly generated, the ratios between the largest and second largest component can vary since both components sizes are also randomly generated. Whereas before the ultimate result was the average size of one single component, now the answer is the size relative between two components. As a result, the trend between the ratios and t values does not follow a linear trend. However, a similar pattern between multiple graphs can be observed.

```
plotRatio(10,4,0.1)
```

```
[<matplotlib.lines.Line2D at 0x7f8983a1b910>]
```

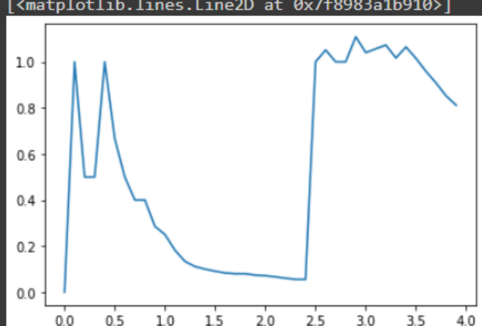

Figure 3: plot of the ratio between the two largest component sizes, y axis, vs t values, x axis

Here we notice a distinctive shape. A sharp spike, followed by a dip and then a spike again that plateaus. This pattern is worthy of note as we find this same trend with multiple lines plotted to the same axes.

```
plotRatio(20,4,0.1) #blue
plotRatio(60,4,0.1) #orange
plotRatio(120,4,0.1) #green
plotRatio(300,4,0.1) #red
plotRatio(600,4,0.1) #violet - light purple
plotRatio(1000,4,0.1) #dark purple
```

```
[<matplotlib.lines.Line2D at 0x7f89834535d0>]
```

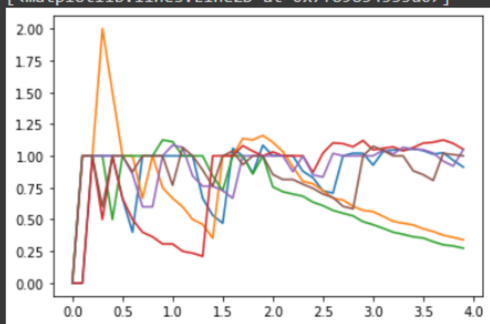

Figure 4: plot of the ratio vs t values with several different values of n vertices

Despite very noticeable outliers, the same trend described before is seen here. In regards to the relationship between the set t values and the nature of the graph, we find this to be the same as in the previous question. The t and n values are used to calculate the probability of a vertex connecting to some components. Higher t values mean a higher probability and higher n values mean a lower probability, in these cases the n values are adjusted the most making the probability of vertices connecting much lower. The result of a reduced p value (t/n) is smaller components. However, the p value has no linear influence on the ratio of the two largest components which is why we see this erratic nature in the graph.