# SOCIAL SIM

An investigation into how fake news, and other information spreads throughout a network over time via analysing patterns and trends observed

UPDATED 10/28/2019

# CONTENTS

# ABSTRACT

The propagation and spread of information is a potent force that is easier than ever to manipulate and control with today's web platforms. Social media networks, such as Facebook and Twitter, provide a means for people to share their thoughts, news outlets to spread their opinions and companies to promote their products. Features of such networks can expose people to information, even if they aren't following the original poster. This phenomenon is due to the fact that whenever someone likes a post, all their followers will see, which will subsequently display the original post and poster on their timeline, giving them the chance to like and follow.

A social network can essentially be represented as a graph, where the vertices are people and the edges are the connections between said people. A simulation can be used to investigate the way in which information spreads and how new connections are made through a middle-man. Over time, posts and users manifest themselves deeper and deeper throughout the network, which can vary based off a plethora of factors, such as who posts, what they post and who their followers are.

# BACKGROUND

Having more followers means having more influence, power and, in most cases, money. Through taking steps in time, one can observe the propagation of fake news, memes and posts within a network and observe how someone can gain popularity and stardom with a bit of luck and the right circumstances. The more followers someone has, the greater the chance of them being followed; however, this isn't always the case. If someone posts something, and someone with a lot of followers likes it, they have the chance to potentially leapfrog the most popular person in the network. This will be expanded upon in the investigation of the results.

A time-step is essentially a 'step in time'. An event file is first loaded in, which allows nodes to be added/removed, people to post and people to follow/unfollow each other. After all these actions have been carried out, posts propagate themselves throughout the network, giving each person (who's following the poster), the chance to like it and spread it to all their followers. Running a simulation with a single-step approach ensures the spread of information is visualised better instead of recursing down and being carried out all at once.

The aspects of the code being investigated are the run-times of the time-step, the patterns revealed by the time-step, the correlation between likes/follows in a network, how differently sized networks scale, the rise and fall of most popular people in networks, and simple time comparisons for different methods in the simulation, such as followSort (similar algorithm to likeSort) and the removal of nodes in the graph. Insertion of people will not be investigated as the times reported were extremely trivial - O(1).

# METHODOLOGY

Through the use of genfiles, multiple random sets of data (of any given size) can be tested and compared. Due to this, it's easy to spot outliers and remove them from the sample set of networks. To see how all the algorithms scale and data varies between different networks, the tests will be performed four times, followed by averaging the data and plotting the results from files of size 50, 250 and 1,000.

The commands run to get the results will be displayed above each graph/diagram, supplemented by the resulting log files in the 'ReportTests' directory. The event and network files for the visualisation of the graphs are also provided too, so the results can be replicated. In order to measure the time certain methods took, I utilise nanoTime. The start time is recorded through using `startTime = System.nanoTime()` before the method is called, followed by `endTime = System.nanoTime()' afterward. The total time taken is calculated through `timeTaken = (endTime - startTime) / 1000000`, the division converts nano seconds to milliseconds.

Furthermore, other tests will be performed on networks with smaller event files and seeing how they compare to ones with the same sized event files - these will be shown visually in the investigation. The difference in time taken taken to execute these two different situations will also be measured.

I suspect that the time-step may take a long time to execute, especially on larger network/event files - given the algorithm has a time complexity of $O(n^4)$. Although scaling poorly, it should output some interesting results and statistics given the information in the log file(s). The data stored within the logs are: how many followers each person has, the total amount of posts in the network, the total amount of follow actions, the total amount of like actions and the time taken to execute the time-step. Through using System.nanoTime() and converting the output to milliseconds, the time taken will confirm the prediction of the time-steps taking longer and longer to run as the network builds up more data to iterate over. Depending on the results, I assume there'll be a relationship between the total amount of follows and total amount of likes in the network.

In order to visualise the data, a plethora of techniques will be employed to plot statistics against each other for comparison. Bar graphs with trend lines will be used for the like/follow relationships - these are particularly helpful to see if there is any correlation between the two (supported by their respective coefficients of determination). Furthermore, a line graph will aid the visualisation of seeing the increase in execution time each time-step, and perhaps reveal an underlying trend - which I suspect will develop and become more clear as the sample sizes get larger. A column chart will be used in order to see who the most popular followers are after each time-step. This'll allow for a greater analysis of the network as a whole and affirm that someone doesn't need to have many followers to begin with in order for them to become the most popular person in the network.

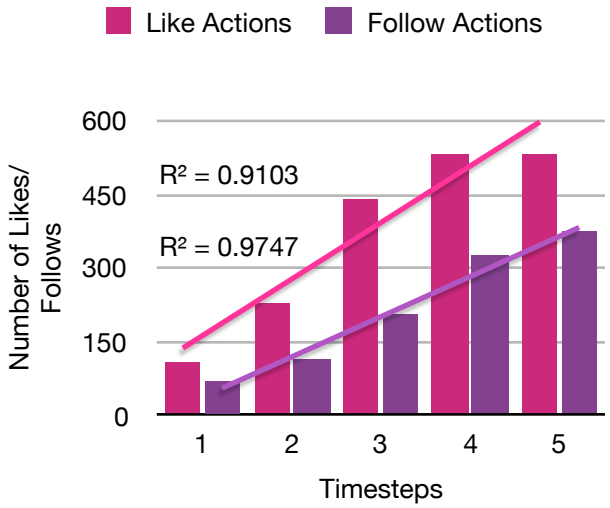
I also suspect the size of the event file will impact the final results/progression of the network quite heavily - as if there is a network file of size 100, but only 50 events, there'll be less posts/ follow events within the network, meaning less people will be connected and less total follows/ likes after the final time-step. This will be observed through making a visual representation of the network via graphing the adjacency list output after each time-step in terminal (made possible due to a small network size and open-source code online - see references). The size of the event file will also heavily impact the time-taken to execute a time-step. If there are less events, there's less data to iterate over, meaning faster method calls. This data will be compared against evenly sized network/event files times through the use of a line graph and plotting the result(s) after each step.

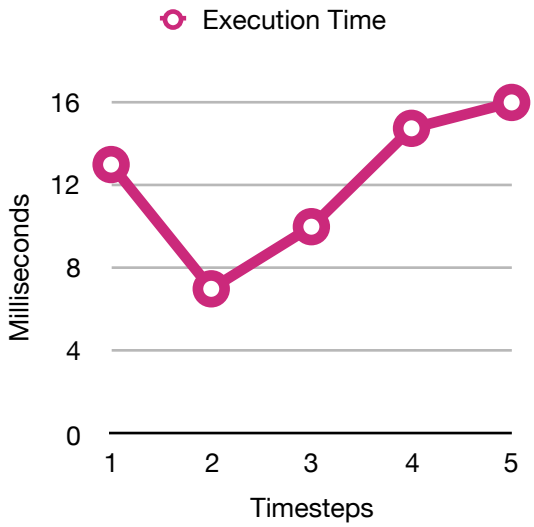
The sorting of the most followed people and most liked posts in the network may also scale poorly with larger data sets - as an $O(n^2)$ algorithm is used to iterate over all the vertices and followers/posts in the network, followed by a heap sort, which is O(nLogn). The removeMiddle function within DSALinkedList may produce surprisingly quick results however - as its best case is O(1) (if the node is at the start/end of the list), and its worst case only O(n). The follow sort and remove function will be plotted on line graphs and compared using different sized sample sets to see the scalability of the algorithms.

All graphs will be drawn up using the average of four different simulations with entirely random network/event files to make data less-bias, as sometimes there'll be many unfollow/remove node events due to the random nature of the file generator. I suspect that there'll be gradual slow-down in growth throughout the network during later time-steps due to less follows being available. As people become more connected, there'll be less and less opportunities for new connections to be made without introducing extra events. This trend may be universal across all measurements and graphs.

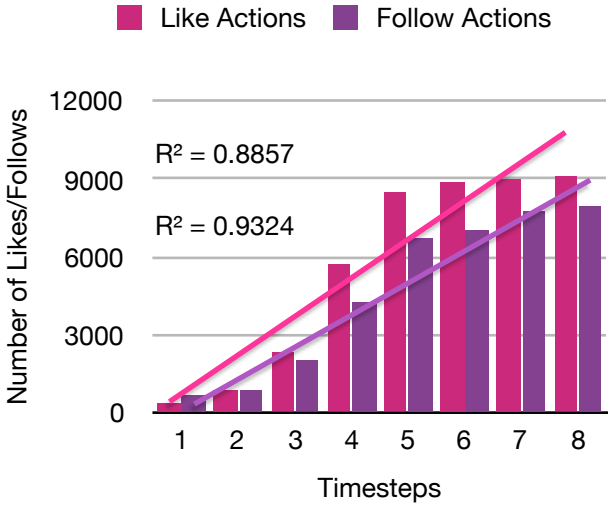
# RESULTS

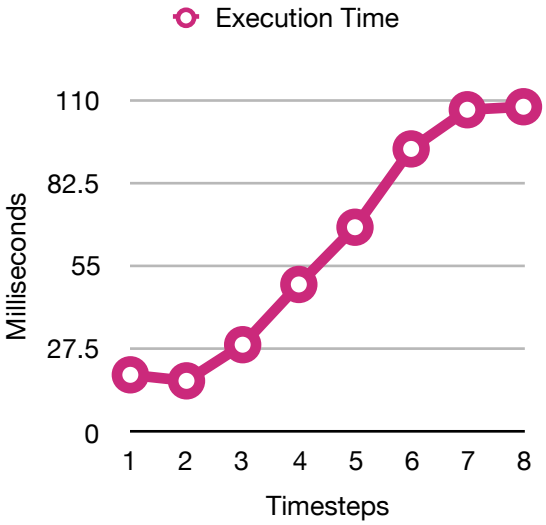
java genfiles 50 50 (x4 averaged out)
java SocialSim -s network.txt event.txt 1.0 1.0



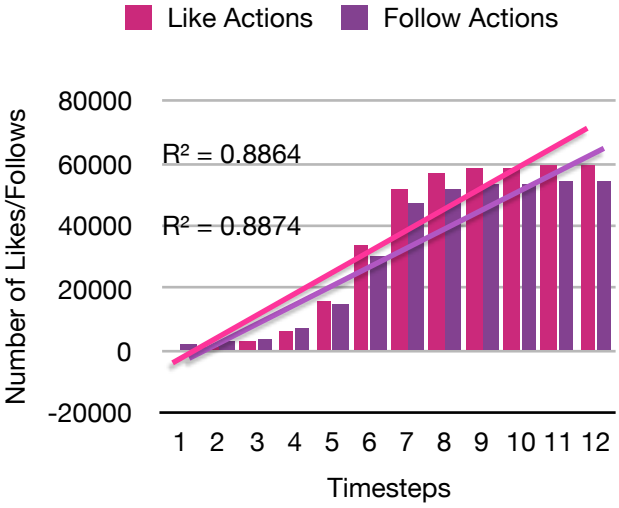



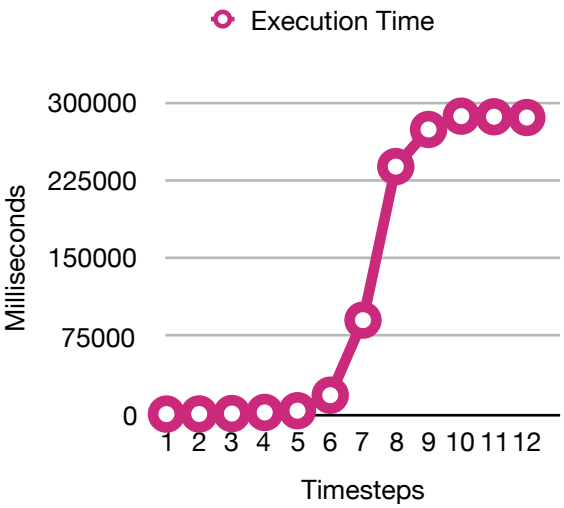

java genfiles 250 250 (x4 averaged out)
java SocialSim -s network.txt event.txt 1.0 1.0




java genfiles 1000 1000 (x1)
java SocialSim -s network.txt event.txt 1.0 1.0



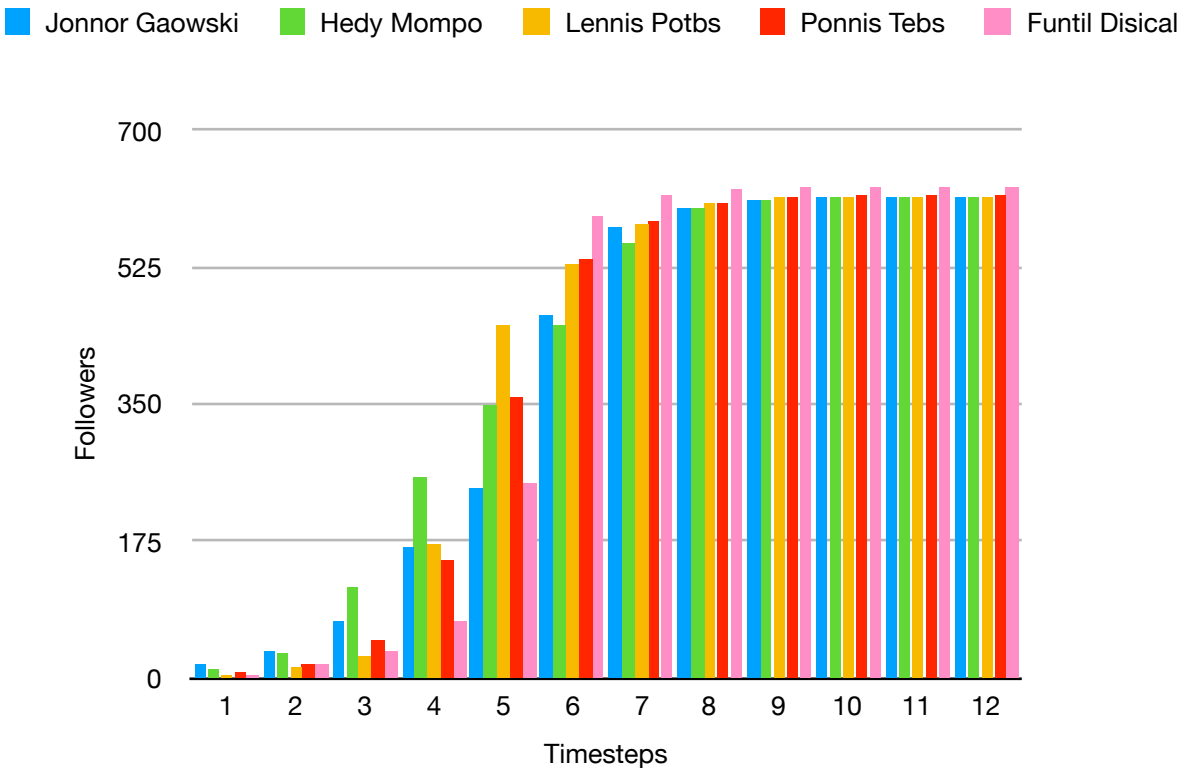



The results above indicate a trend corresponding with the total amount of follow/like actions made. The final time graph, as well as the like/follow graphs are reminiscent of a sigmoid curve, which can be seen gradually developing and refining as the sample size increases. Charles Darwin observed this as the 'Carrying Capacity' and how a species' population can't exponentially grow forever. He examined that there was resistance from the environment as food supply declined due to the population being too large. In a way, the networks created above have a similar trend. The possible likes/follows that could be made are eventually exhausted, hence plateauing once reaching a certain point in the network's life span.

Initially, the network growth is slow as there are only a small amount of follower relationships loaded in via the network/event files - this is commonly referred to as the 'lag period'. Over time, these numbers increase as more and more people become connected, at a point even going through a phase of exponential growth. There are plenty of available follows/likes that could occur. Since we set the probabilities of both these instances happening to 1.0 (100%, i.e. if someone can like/follow in a time-step, they will) we can observe patterns and trends without random probability getting in the way. A lower probability would increase the number of time-steps needed to eventually plateau, but the results would, for the most part, remain the same. The reason that the time plateaus, as well as the data, is that a similar amount of information is being iterated over each time, since it isn't changing much outside of the growth period.

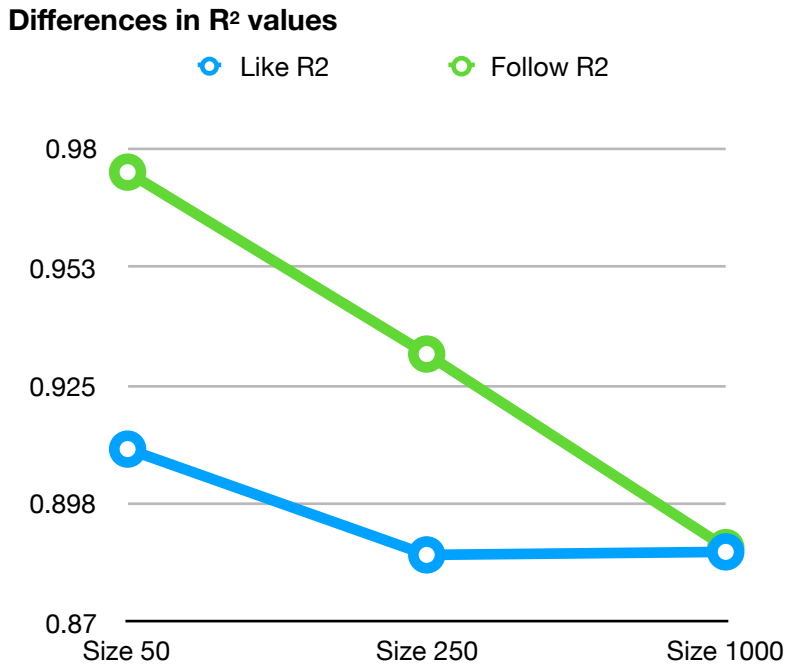
It makes sense that the exponential growth period displays a sharp jump, in both like/follow actions and in time taken to execute. Since there are more followers to iterate over in all nodes, it takes longer for the CPU to go through every single person within the network. Furthermore, if we look at the data of who's the most followed, it also follows a trend which corresponds to the sigmoid curve.

**Most Popular Followers (1000 sample size)**




The above graph displays some interesting figures. During the period of exponential growth, some people's followings almost triple in size, and a few unlikely contenders from earlier on in the network rise to fame. Despite Jonnor Gaowski having the most followers at the start, others quickly seem to leapfrog him. Towards the end, everyone becomes equal (for the most part), with Funtil Disical maintaining a steady 627 followers and others following closely behind (Jonnor, Hedy, Lennis with 615, Ponnis with 616). This emphasises the fact that it doesn't matter how many followers a person has - it's who their followers are.

It's also worth discussing the coefficients of determination and what they mean. When comparing the average runs of three different sample sets, we can observe the $R^2$ variables and how they get closer and closer together.

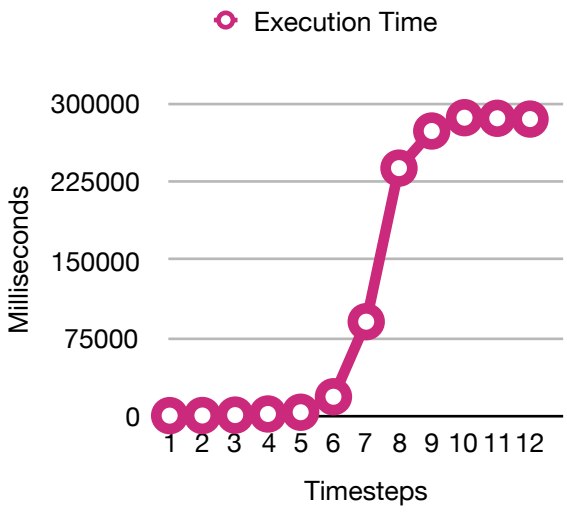



Larger sample sets often provide more precise estimates, which is proven by the coefficients of determination having a 10% difference between sample sizes 50 and 1,000. The green line declines in a linearly negative direction. Essentially, the amount of total likes performed in a network seemingly corresponds to the total amount of follows performed.

Experimenting with larger data sets tends to give a more accurate estimate and indication of future trends, however, it doesn't come without its pitfalls. Due to the $O(n^4)$ time complexity of the time-step, sample sizes over 250 take significantly longer than smaller ones. It's worth noting that both the network and event file were large when testing. If run with a network file of size 1,000 and smaller event file of size 50, the execution time is almost 800x faster (354ms vs 285,366ms at their peak times) as there's a lot less to iterate over:

java genfiles 1000 50 (x1)
java SocialSim -s network.txt event.txt 1.0 1.0

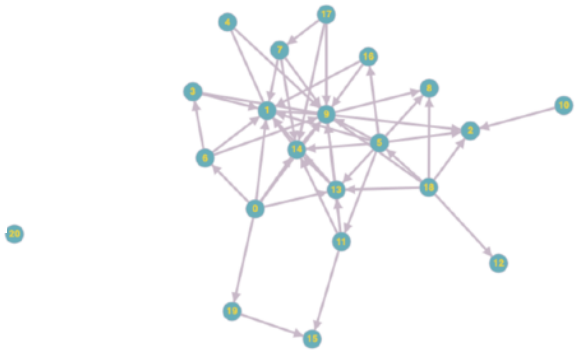

java genfiles 1000 1000 (x1)
java SocialSim -s network.txt event.txt 1.0 1.0



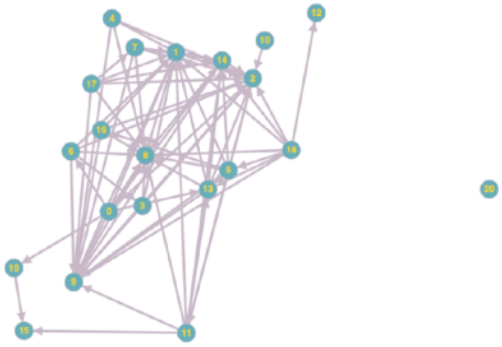
java genfiles 15 15 (x1)
java SocialSim -s network.txt event.txt 1.0 1.0

**0** (13 follows performed)

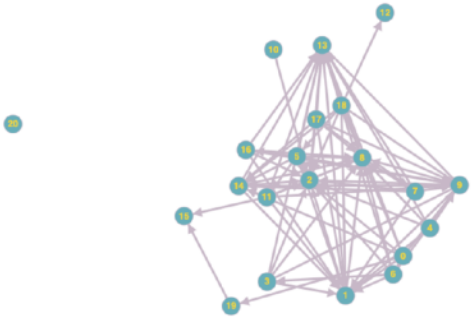
**1** (25 follows performed)



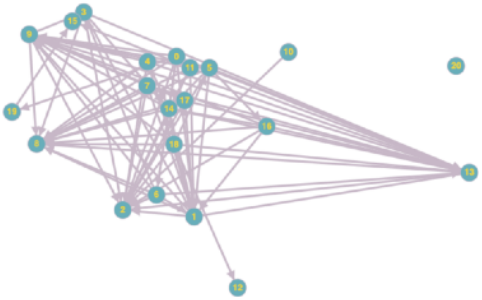
**2** (37 follows performed)



**4** (80 follows performed)



**5** (89 follows performed)



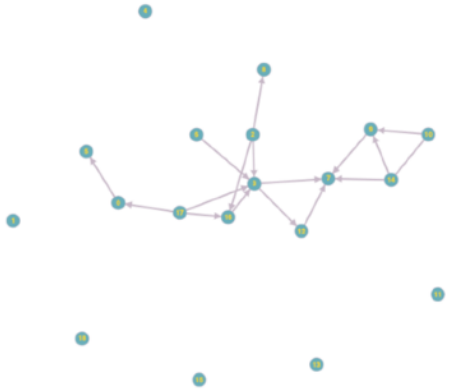
**6** (94 follows performed)

A visualisation of a network (with files both of size 15), show that connections start forming relatively slowly during the lag period, but as expected, sharply jump around time-step 4. Reminiscent of a sigmoid curve, connections eventually start to plateau (as seen in the later time steps). The first visualisation is the network without the events file loaded in - which explains why two nodes (10 and 12) gain connections/start following others after the first step. Node 20 remains lonesome for the rest of the simulation - which means that they aren't following anyone and have 0 followers (which is inevitable given the files are randomly generated).

The jump in connections toward the end of the simulation are much smaller than ones in the middle, which reaffirms the sigmoid curve model. Mitigated from the above graphs (due to space, but stored in logs) is time-step 3, which had 53 total follow actions. Only 5 new follows were made between steps 5 and 6, whereas there were 23 between steps 3 and 4.

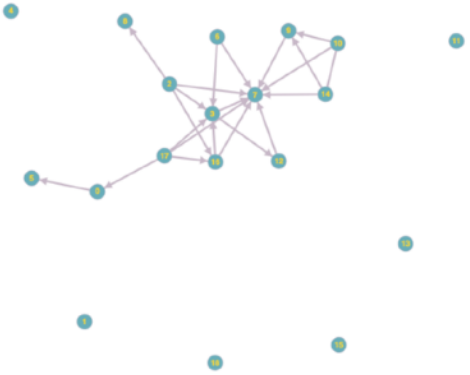
java genfiles 15 8 (x1)
java SocialSim -s network.txt event.txt 1.0 1.0

**0** (6 follows performed)



**1** (13 follows performed)

**2** (18 follows performed)

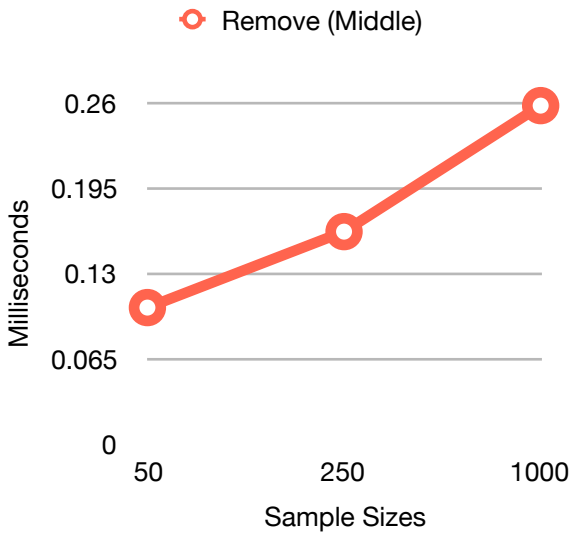


**3** (23 follows performed)

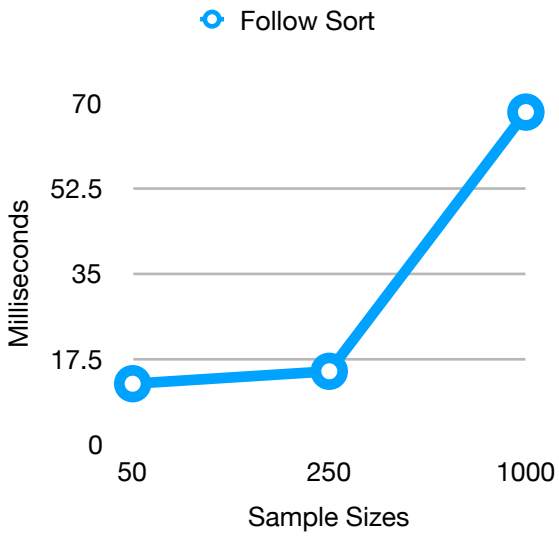
Creating a network of size 15 with a smaller event file develops noticeably different to a network with a larger event file - the period of exponential growth is much less clear. An extra node is added in the events, which explains why the first graph has 5 disconnected nodes and the others have 6.

A significant amount of people remain without relationships, which is a result of the event file being smaller and catering for less possible connections (via post and follow events). The total number of follows performed is also significantly less, which makes sense given the lower pool of available actions (23 vs 93 follows performed after the final time-step).

**Miscellaneous Statistics**




java genfiles (50, 250, 1000) (50, 250, 1000)
java SocialSim -i




java genfiles (50, 250, 1000) (50, 250, 1000)
java SocialSim -i

The remove method, as an O(n) algorithm at its worst case, scales almost linearly to the amount of nodes within the linked list. The results are surprising, only taking 1/4 of a millisecond when removing a node from the middle of a linked list of size 1,000. Follow sort on the other hand (which follows the same algorithm as likeSort in simulation.java), scales much worse as it not only uses an $O(n^2)$ algorithm to traverse the data, but also a heap O(nLogn) to sort the data. There's a drastic jump in time taken to sort the amount of followers in a network of size 250 compared to a network of size 1,000 (taking 4.5x longer).

# CONCLUSION

Through using a simulation to mimic the spread of information in a network, multiple trends and patterns can be observed and used to predict future simulations of almost any data size. The sigmoid curve model seems to be a universal constant across different network models - which proves the prediction of progression eventually slowing down as the amount of actions possible become exhausted. A correlation between the total amount of post likes and follows indicates that the two are closely related, show by the $R^2$ values only being 0.001% apart with a sample size of 1,000. This is further backed up by the two $R^2$ values getting closer and closer to each other as the sample size increases.

The idea that someone does not need many followers in order to become the most popular person in the network was also proved. This was shown via the network size of 1,000, where the most followed person, was actually one of the least followed at the start, affirming the importance of who someone's followers are. One single like from the most followed person in the network could expose all of their followers to the original poster - which could drastically change the shape of a graph. Furthermore, the events file was observed to have a significant impact on the outcome of a network's progression, as shown by the graphic visualisations and varying sizes of files. Through using graphs and statistics to map out various algorithms and processes, one can see the complexities of a network much clearer and hence draw conclusions about the future. Further investigations could be held to see if any trends change in larger data sets and if the harsh time executions remain the same over even more varying event files.

# REFERENCES

Sigmoid Growth Curve | BioNinja. (2019). Retrieved 26 October 2019, from https://ib.bioninja.com.au/options/option-c-ecology-and-conser/c5-population-ecology/sigmoid-growth-curve.html

Creating graph from adjacency matrix. (2019). Retrieved 27 October 2019, from https://graphonline.ru/en/create_graph_by_matri