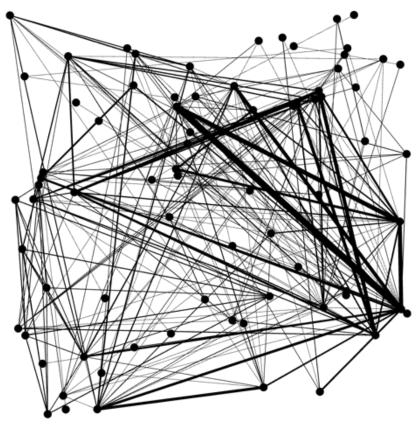
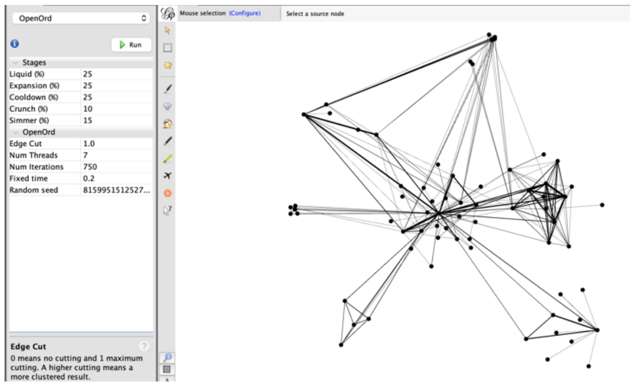
CNT 5805 – Project 1 Les Misérables

1. Analyzing this network of Les Misérables will allow us to discover and show the levels to the relationships that play out in the story. Due to the weighted nature of this graph, it will be possible to see which characters interact with each other the most and in addition, which characters have little relation with each other. Without reading any of the story, we should be able to find which characters are the most important pieces to the plot.
2.



There definitely isn't much information to be found from initially loading in the data, however the first thing I notice is that there are a couple relatively thick triangles popping out in this graph. There also doesn't appear to be a major hub in this graph which tells me the story is unlikely to one dominant character, but instead may have many characters that are important to the story (If anything, the hub may be one of the nodes toward the bottom right of the graph, but it's too early to tell). Finally, the average degree will likely be small in comparison to the large number of characters (nodes) that are present in this graph.

3. **Algorithm 1: OpenOrd**

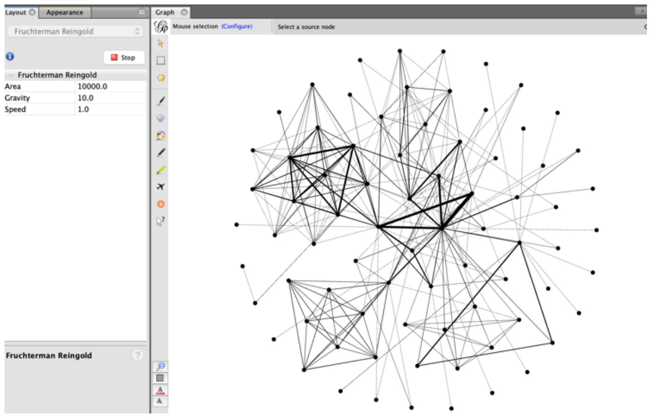


Stages	
Liquid (%)	25
Expansion (%)	25
Cooldown (%)	25
Crunch (%)	10
Simmer (%)	15
OpenOrd	
Edge Cut	1.0
Num Threads	7
Num Iterations	750
Fixed time	0.2
Random seed	8159951512527...

Edge Cut
0 means no cutting and 1 maximum cutting. A higher cutting means a more clustered result.

Here is the resulting graph from the OpenOrd algorithm. According to the Gephi website, this algorithm is supposed to show the clusters within a graph by cutting off long edges between nodes. It's built upon the Fruchterman Reingold layout which has an Edge Cut value of 0 (no clustering), while this layout can choose any other value between 0 and 1 for the Edge Cut (the higher the number, the greater the clustering). In this layout, I set the Edge Cut to 1 to allow for maximum clustering. The graph has developed roughly 7 clusters of nodes, with 2 main clusters shown in the middle. It should be noted that I don't think this layout has shown us much and I believe that is because this graph is too small for what this algorithm is intended to do. I would only use this layout in the future if my graph had at least a few thousand nodes, not 77.

Algorithm 2: Fruchterman Reingold

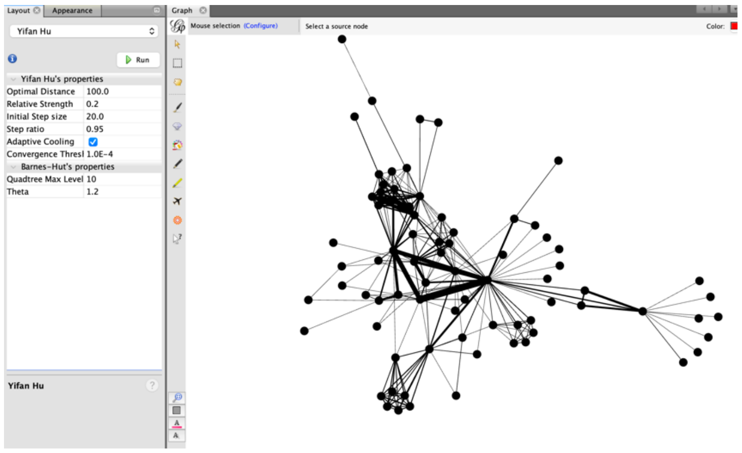


Fruchterman Reingold

Area	10000.0
Gravity	10.0
Speed	1.0

The Fruchterman Reingold algorithm is designed to make a graph that has edges of a roughly uniform length. By having a roughly uniform length, the idea is that the edges will never intersect which can give the user a better idea of how concentrated certain parts of the graph are. The algorithm runs iteratively to pick the best length for the edges. This process means that large graphs will take a very long time to run through this algorithm as it passes through each iteration (no issues with a graph this small though). The graph is certainly easier to read than the original graph, having spaced out a little more and gotten rid of some of the initial hairball issues, however this algorithm doesn't appear to have told us much. It has, however, made it somewhat clearer that there is a hub to look out for in this story and it appears to be the node in the very center of this graph.

Algorithm 3: Yifan Hu



Yifan Hu

Yifan Hu's properties

Optimal Distance	100.0
Relative Strength	0.2
Initial Step size	20.0
Step ratio	0.95
Adaptive Cooling	✓
Convergence Thresl	1.0E-4

Barnes-Hut's properties

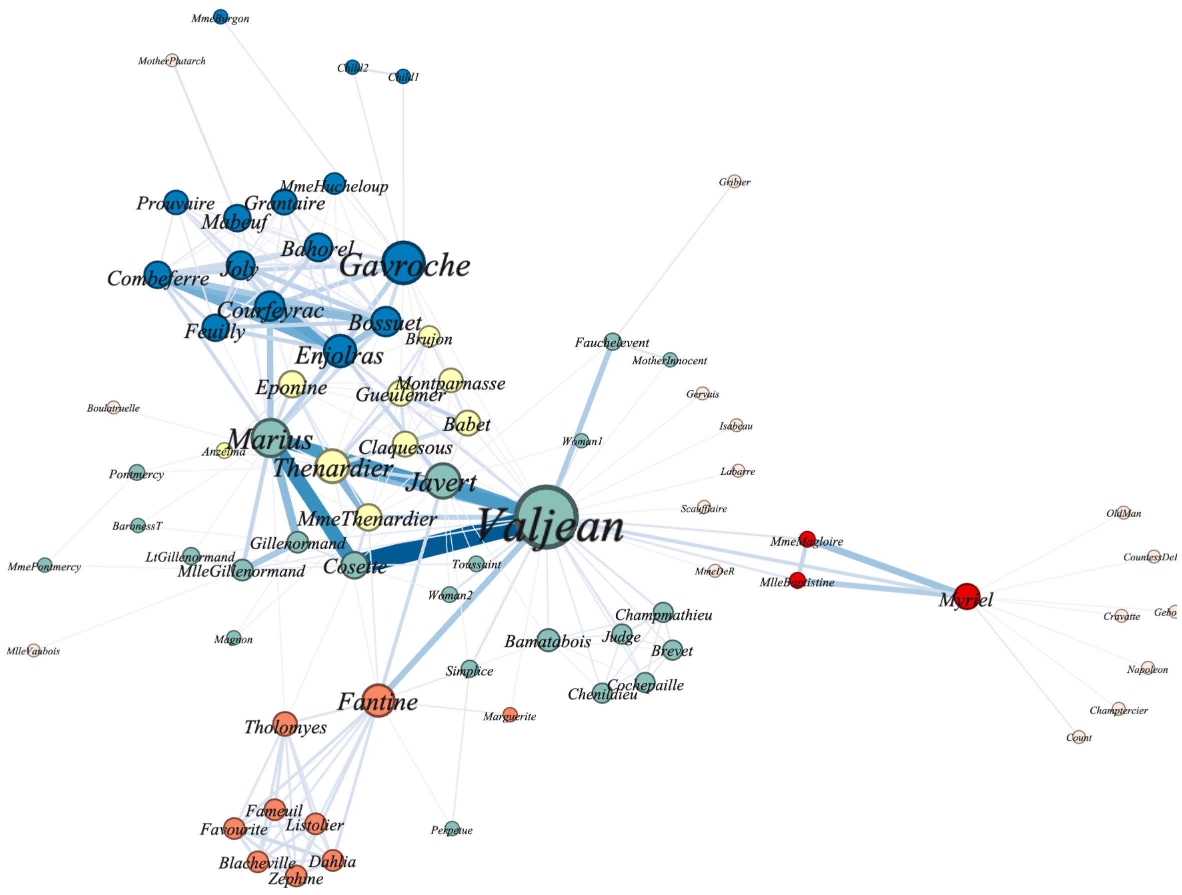
Quadtree Max Level	10
Theta	1.2

The Yifan Hu algorithm, like Fruchterman Reingold, is force-directed and aims to reduce the complexity of the graph by using multilevel techniques. This algorithm works well on large graphs but most importantly is an efficient choice that will be much faster than something like Fruchterman Reingold. In the Les Misérables graph, the Yifan Hu algorithm has developed a few clusters with a high clustering coefficient, unlike the clusters found by running the OpenOrd algorithm. In addition, there are a few clear hubs in this graph formed by the thick triangle in the middle (It appears all 3 can be considered hubs here). I think this graph has made some clear readability improvements over the original, which was erratic and far too dense.

Most Useful Layout:

I think the answer in my testing is to go with the Yifan Hu algorithm. The first two algorithms didn't show any intelligible clustering like the Yifan Hu algorithm did. In the case of the OpenOrd algorithm, the graph was split into clusters, but the clusters didn't appear to show much of anything because most of the nodes were still connected mainly to nodes across the graph instead of in the clusters. The Fruchterman Reingold algorithm is certainly a solid option, but in this graph it's much harder to identify clusters than it is while using Yifan Hu. Considering that Yifan Hu is a fast and efficient algorithm, even in the situation where it and Fruchterman Reingold appear to provide similar benefits, I would still be drawn to select Yifan Hu instead. OpenOrd is the only layout I didn't consider using and I feel that Yifan Hu is the best choice.

4.



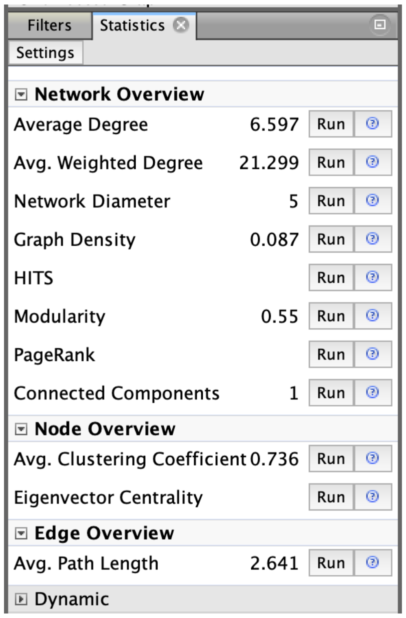
*I had to do a label adjustment to allow for readable labels and larger nodes, so the clustering in the upper left of the graph has been somewhat split apart versus what could be seen in the preliminary graph from before.

*The color coordination in this graph was done through modularity – each color represents a different cluster of nodes that generally share the same neighbors. It seems that the Yifan Hu algorithm did a fairly good job of separating the clusters from one another, however many of the nodes in the middle that belong to the light green cluster have been scattered across the graph. If I were to do this again, I would probably choose to look at all possibilities when it comes to layout to ensure that I choose the best one (This layout does still appear useful though).

The first thing I quickly noticed is that the largest node is Valjean, and that character has links to nearly every node on the graph except for the clusters in the bottom left and top left. I find this intriguing because those two clusters are extremely isolated from the rest of the graph and both clusters have nodes displaying clustering coefficients of 1 or nearly 1. For example, in the bottom left cluster, Zephine, Dahlia, Listolier, etc. all experience the same neighbors and should show a clustering coefficient of 1 when the stats are broken down later in this project.

There also appears to be a split in the story where we have the left side of the graph versus the right side of the graph, all connected by this character Valjean. The right side of the graph shows roughly 20 characters that have relation to themselves and Valjean only and have no relation to any of the characters in the top left of the graph (The same can be said for the bottom left cluster – no relation to the busiest section of the graph in the top left). I would reasonably deduct from the graph that it’s likely that the bulk of the story takes place surrounding the characters in the top left, while the characters on the right may be asides that Valjean experiences throughout the main story.

5. **Stats Panel:**



Network Overview

Average Degree	6.597	Run	ⓘ
Avg. Weighted Degree	21.299	Run	ⓘ
Network Diameter	5	Run	ⓘ
Graph Density	0.087	Run	ⓘ
HITS		Run	ⓘ
Modularity	0.55	Run	ⓘ
PageRank		Run	ⓘ
Connected Components	1	Run	ⓘ

Node Overview

Avg. Clustering Coefficient	0.736	Run	ⓘ
Eigenvector Centrality		Run	ⓘ

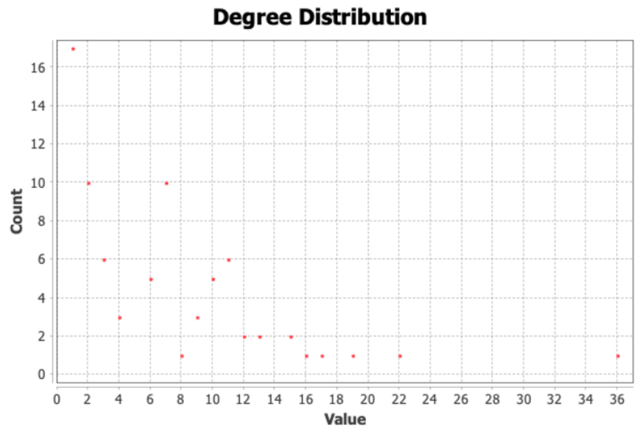
Edge Overview

Avg. Path Length	2.641	Run	ⓘ

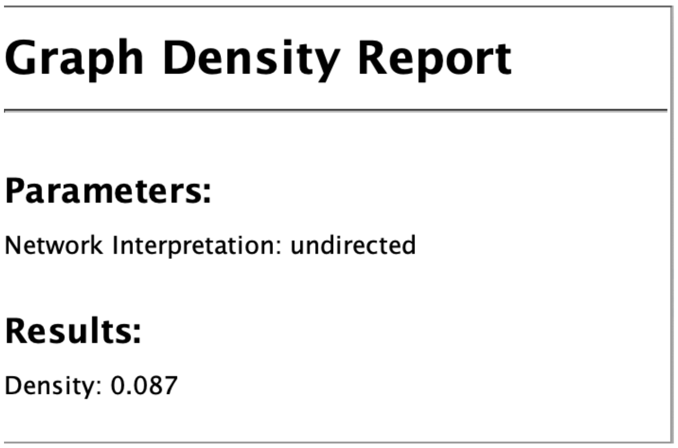
Dynamic

Results:

Average Degree: 6.597



The degree distribution is fairly loose; however, it follows the trend of the Power Law distribution as opposed to the Poisson distribution. This makes sense because the data *is not* random, so the theory tells us that real world data will fit better into the Power Law distribution.



Graph Density Report

Parameters:

Network Interpretation: undirected

Results:

Density: 0.087

To the left is the graph density report. This is a value between 0 and 1, where 1 equates to all possible edges in the graph filled out. This value can be calculated by using the formula:

$$\frac{2E}{N(N-1)}$$

where E is the number of edges in the graph and N is the number of nodes. For our calculation, there are 254 Edges and 77 Nodes and plugging into the formula will give us the answer of 0.087. This means our graph is quite sparse, which we can confirm by finding the maximum number of edges that this graph can have. By doing N choose 2, we can find that the maximum number of possible edges is 2,926 which is magnitudes higher than the 254 we have found in our graph.

I have already discussed communities/clusters a couple times throughout this report so far, but there do appear to be a few communities in this graph. The most isolated of those would be the orange community in the bottom left of the graph. In addition, the graph does have a few giant components. The most obvious of those would be the node of Valjean, who is the most connected character in the story (Gavroche, Marius and Javert are all larger nodes in the graph as well). Even more striking would be the triangle connection of nodes between Valjean, Cosette and Marius, all of whom share many chapters of the story together.

Gephi allows us to measure the modularity of a graph which is a measure of homophily – in our graph, the modularity is displayed as 0.55. The range of modularity is [-1,1] where numbers close to 1 will represent strong community structure and numbers close to -1 will represent no community structure. We can see that this graph does experience a strong measure of community meaning that homophily can be detected.

6. Final Contemplation:

In this project we were given some very simple data about the Les Misérables novel and I think there could be some additional data that would add to the study. For example, if Les Misérables includes a lot of dialogue, I would be interested in a dataset that shows how many times each character mentions another character while in conversation. There may be some characters that don't appear in many chapters together, yet there is still a close relationship between those characters in the plot (Characters do not live in the same area, characters are rivals in the story and do not spend time together, etc.).

I think this project taught me a lot – to start, I am brand new to Gephi and the Network Science sphere. I haven't personally read Les Misérables, so I feel that I may have been able to take more from this analysis or come up with some more interesting conclusions if I had been familiar with the characters and the story. However, this didn't stop me from learning a lot about the workings of Gephi starting from how different layouts can impact the look of a graph. It's quite incredible to see an algorithm take a messy blob of data in a graph and turn it into something readable that can display relationships and community clusters. While reading the theory in class, we learned that the degree distribution of real-world data would appear more like a Power Law distribution, and we were able to prove that to be true for at least this practical dataset. I think I generally just enjoyed being to take the things we learned in class and see those things appear while working around with this data.