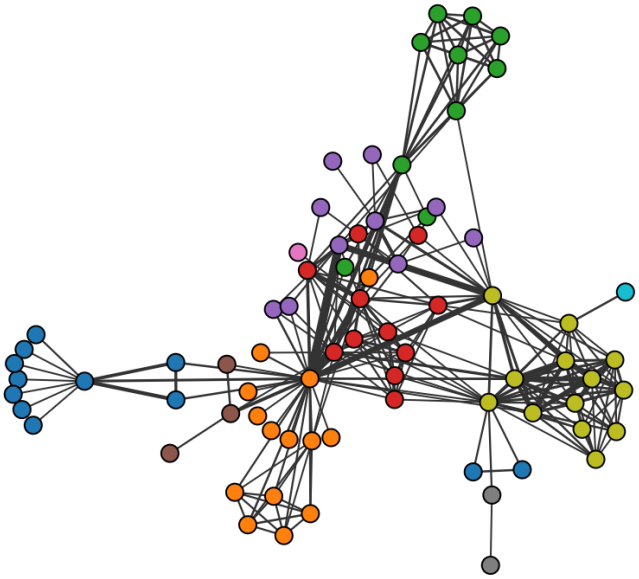
CS/INFO 3300; INFO 5100
Homework 9
Due 11:59pm Tuesday, November 12

Goals: Practice using d3 to create a network diagrams

1. You will visualize a network dataset, `miserables.json`. This dataset contains character "co-occurrence" in the novel Les Misérables. An algorithm counted the number of times that each character appeared in a chapter along with another character to generate the network. For example, if one character appears with another in three separate chapters, then the network connects these two characters with a `value` of 3. Characters have also been numbered based on `group` affiliation so that patterns might emerge.

Example (your diagram may not look identical to this):



A. Following your <p> element, create an **SVG element 600px in width and 600px in height**. Within a `<script>` tag, use d3 to create a **<g> element** within your SVG to contain your network diagram. Using either promises or await, **load the dataset** into memory. Use `console.log()` to examine the dataset. You will find that it contains an object with two keys, `nodes` and `links`. We recommend that you assign these two values to variables for ease.

Create two scales for your diagram. First, create a `scaleLinear` to help adjust stroke width of lines based on the `value` of each edge. We want thicker lines to appear for stronger co-occurrence connections. To do so, set the domain of your scale by computing the `d3.extent` of the `value` entries in the edges array. Set the range to be `[1, 5]`. Second, create a `scaleOrdinal` using the `d3.schemeCategory10` color scheme which you will use to color nodes by each character's `group` value.

B. Construct a `d3.forceSimulation` model for your network diagram. You can use the data from the `nodes` key in the dataset as **nodes** in the model. Your model should include the following forces:

- A **linking force for edges** in the network. Use data from the dataset's `links` key to build your links. Source and target in the edges array correspond to the `character` property of nodes in this dataset, so **be sure to set `.id()` properly for this force so that it knows what to look for in each of the nodes when connecting them.**

- A **many body repulsive force** between all nodes. Tune the strength of this force so that both clusters and outliers are evident and remain completely within the canvas. A value around `-20` should work fine.

- A **centering force** that keeps all nodes in the center of the chart. You do not need to set any strength value.

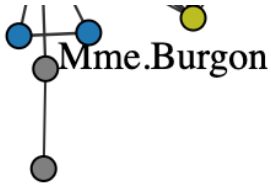
C. Make a function, `render()`, that **uses a data join to draw the edges** and another **data join to draw the nodes**.

Draw the edges first so that they do not appear to be on top of the nodes. Use your linear scale to set the `stroke-width` of the black connecting lines based on the `value` of each edge. **Make sure that `opacity` remains at the default of 1** for performance reasons.

Draw circles 5px in radius for each node and **set their `fill` color using the color scale you made earlier**. Recall that your color scale is supposed to be based on the `group` value for each node. Give them a **1px outer stroke** in a dark grey or black color. Be sure to use `join()` properly so that you only create nodes/edges once and update all of them each time `render()` is called.

Finally, add an `.on("tick", ...)` call to your force simulation to call your `render()` function. If your simulation quickly gets slow or has ghostly trails, check your join for issues with what it is selecting each time `render()` is called.

D. Add an `.on("mouseover", ...)` and an `.on("mouseout", ...)` event handler to each of your circles. This handler should show the name of a character when you put your mouse over each of the circles. Note that each of the nodes entries you used to generate the circle elements has access to the name of the character as well as the x and y position. Your character name pop-up should be offset from the circle so it is legible, but it is okay if it is harder to read in crowded areas of the chart.



(see next page for bonus item)

Part E of this assignment is completely optional.
Students who successfully complete part E will receive one free quiz grade drop. If you've missed a quiz, this is an easy chance to win back one of those 0 grades.

E. Instead of a mouseover, use the `d3.drag()` tool demonstrated in class notes. Use `.fx` and `.fy` parameters on your nodes in order to deliver smooth animations, and reheat the simulation as necessary to permit node movement using `alpha()` and `alphaTarget()`. When the user starts dragging a node, **the name of the character should appear in a text label just as it would in part D**. It does not need to appear when mousing over a circle if you complete part E, just make sure it appears when the user drags a node.