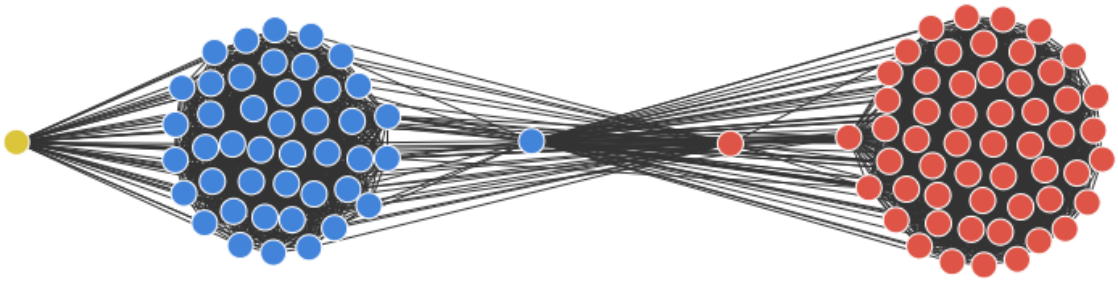
CS/INFO 3300; INFO 5100
Homework 8
Due 11:59pm Wednesday, November 2

1. For this homework, you will visualize two provided datasets, **senate.109.rollcall.nodes.csv** and **senate.109.rollcall.edges.csv**. These datasets encode a graph of US Senators during the 109th congress (2005 to 2007). Edges have been drawn between senators who share similar voting patterns. Senators who almost always disagree will not be connected.

Example:

Problem 1:



A. Following your <p> element, create an **SVG element 800px in width and 200px 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 both datasets** into memory. Finally, use **d3.scaleOrdinal** to build a color scale to show party affiliation. Set the domain and range of the scale manually so that **"Dem" maps to a blue color, "Rep" to red, and "Ind" to yellow**.

B. Construct a **d3.forceSimulation** model for your network diagram. You can use the data from **senate.109.rollcall.nodes.csv as nodes** in the model. Your model should include the following forces:

- A **linking force for edges** in the network. Use data from **senate.109.rollcall.edges.csv** to build your links. Source and target correspond to the **"icpsr"** property of nodes in this dataset, so **be sure to set .id() properly for this force.**

- 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 -150 should work fine.

- A **y-positioning force** that pulls all nodes towards the **middle (i.e. height / 2).** Set its strength to something less than 1.0 so that it doesn't crush everything into a line.

- An **x-positioning force** that pulls nodes to **different x locations** based on their "party" property. This will help show divisions between political parties. Nodes where **"party" is "Dem" should be pulled towards width*0.25**; nodes where **"party" is "Rep" should be pulled towards width*0.75**; all **other nodes should be pulled towards 0** (hint: you can create a

function that uses "d" from the nodes data inside of the .x() call for your forceX() just like you would for .attr() in a data join). Set its strength to a value less than 1.0 so that it doesn't crush everything into a single point.

(We are not using a centering force for this graph visualization – you are welcome to experiment with including it, but **do not put it in your final submission**)

C. Make a function, **render(),** that **uses a data join to draw edges** and a **data join to draw nodes**. **Draw the edges first** so that they do not appear to be on top of the nodes. You can choose the appearance of your edges, but **make sure that opacity remains at the default of 1** for performance reasons. **Draw circles 8px in radius for each node** and **set their color using the color scale you made in A**. Give them a **1px outer stroke** in a dark grey 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.

BONUS: (no extra credit offered)

Adjust your code so that you can **drag nodes around the screen using your mouse** (hint: use d3.drag() instead of writing your own drag framework). Use .fx and .fy parameters on your nodes in order to deliver smooth animations, and reheat the simulation as necessary to permit node movement. When the user starts dragging a node, **the name of the senator should appear in a text label**. The label can either be placed in a corner of the SVG canvas or follow the mouse. The label should disappear when the drag ends.