Girls Talk Math:
Networks Group

1 Introduction

By this point in high school, you've probably taken a lot of classes on math–pre-algebra, algebra, geometry, algebra II, calculus, etc. etc. And it might seem like everything at this point is about how we manipulate equations and understand functions. What we're going to teach you here probably will seem different–and maybe not like any of the math you've seen before, but requires the same amount of rigor and has the same usefulness.
To start out with, we define a *graph* not as a relation between a domain and range in Euclidean space, but as a structure in which certain pairs of objects are related to one another and others aren't. These objects are called *vertices* (or nodes) and the relations are called *edges* (or links). If these vertices represent some kind of meaningful data (and perhaps contain *attributes*), then we typically call the graph a *network* (the terms are often used interchangeably. Such networks are used in social media and business (predicting suggestions for 'who to follow on instagram, or what books you might like to buy on Amazon), as well as in physical/biological sciences (mapping what regions function together in the brain).

2 Graph Theory

Graph theory is the mathematical study of (wait for it) graphs. Let's begin our study by labeling the edges and vertices of a particular simple graph (below).

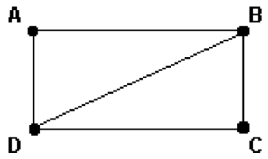


Figure 1: A simple graph

We use letters in this depiction, but these could in general be numbers, names, etc. The vertices here are A, B, C, and D. The edges are AB, BC, CD, AD, and BD. If two nodes share an edge, they are said to be *adjacent*. When we list these vertices, the order is not important—BD is the same edge as DB—in general, this is not always true, but it is true

for *undirected* graphs, which are the ones we will study here. Notice also that nodes A and C are adjacent to two edges each–but B and D are each adjacent to two three edges. The number of edges adjacent to a particular node is called that node's *degree* (so A has degree 2, etc.)

In the previous graph (Fig. 1), one would travel from node A to each other node in the graph, through its edges. If this is true, the graph is said to be *connected*. In the graph below (Fig. 2), the graph is disconnected, as F and G cannot be reached (through traveling via edges) from the rest of the graph.

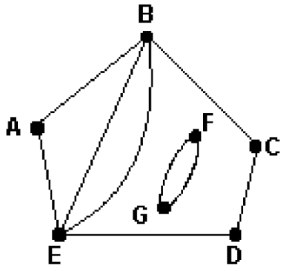
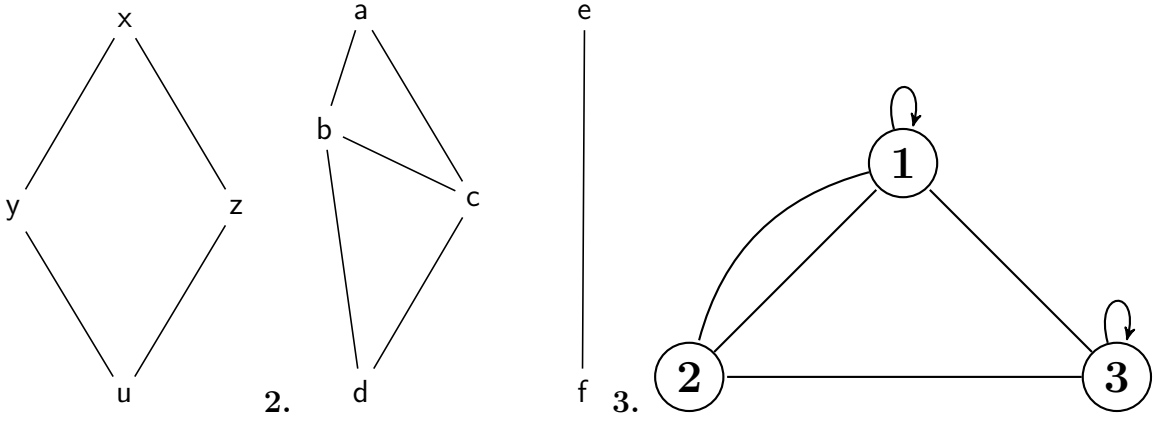


Figure 2: A disconnected graph

If a graph is disconnected, we can divide it into *components*. Each connected subgraph is called a component–in the above graph, F, G, and the corresponding edges form one component; A, B, C, D, E, F, and the corresponding edges form the other component. Saying that a graph is connected is equivalent to saying that it has one component. *Important: Saying that two nodes are adjacent is more specific than saying they are connected (these are frequently confused).*

Exercise 2.1 For each figure below, list the vertices and their degrees, as well as the edges. Decide whether of not the graph is connected–if it is not, how many components does it have.



1. Nodes and their degrees:

Edges:

Connected?
Components:

2. Nodes and their degrees:

Edges:

Connected?
Components:

3. Nodes and their degrees:

Edges:

Connected?
Components:

Were you confused on the last exercise? This was admittedly intentional. Nodes 1 and 3 each have an edge unto themselves–also called a *loop* (or self-edge). These contribute two to their degree (so node 1 has degree 5 and node 3 has degree 4). Also, this graph has two edges from node 1 to node 2 (*12*)—a graph with multiple edges between the same pair of nodes is called a *multigraph*.

2.1 Euler Paths and Euler Circuits

An important question in mathematics was asked to Leonhard Euler in the late 1700s in the city of Königsberg, Prussia. Using the map of the city below, he was asked, is there some route in the city wherein one would cross each of the seven bridges once and only once? Now, this may not seeem to have to anything to do with graph theory, but it in fact led to the first journal article in the subject. He redrew the figure above as a graph, considering the land masses to be vertices and the bridges to be edges.

Exercise 2.2 Using the above reasoning, recreate Euler's representation of Königsberg. Label the degree of each node.

Now, with this tool, the problem is not so hard—the problem can be restated: are there any paths through the graph such that one crosses all of the edges once and only once. Upon careful consideration, you may see that even degree vertices are useful to this determination—in

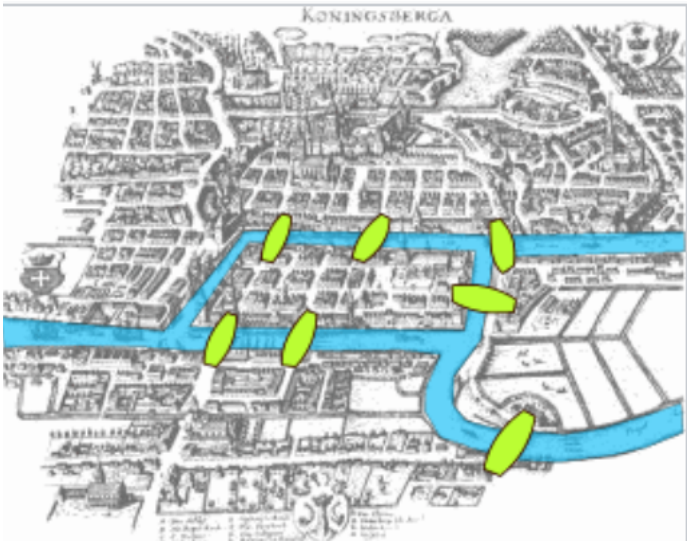


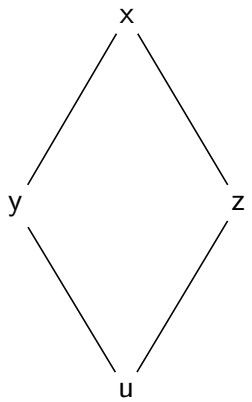
Figure 3: A map of the city of Königsberg, showing the seven bridges and the land masses which they connect.

a degree 2 vertex, one can enter and leave the vertex through two edges (if it is four, this can happen twice, etc.) But with odd degree nodes, one may get 'stuck' at the vertex. In fact, for there to be such a path through the graph as Euler desired, there must be two or less odd degree vertices. Such a path is named an *Euler path*. Because, as you showed above, there are more than two vertices of odd degree in the Königsberg graph, there are no Euler paths in the associated city.

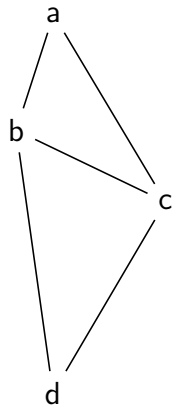
Theorem 1. *In order for there to be an Euler path (in which each edge is traversed exactly once) in a graph, that graph must be connected and have two or less vertices of odd degree. If we restrict the path to circuits (in which the path must end at the same point at which it starts), the graph must have exactly zero vertices of odd degree (and must be connected). Such a path is called an Euler circuit or Eulerian cycle.*

Exercise 2.3 Do the graphs from Excercise 2.1 have Euler paths? Euler circuits? Why or why not?

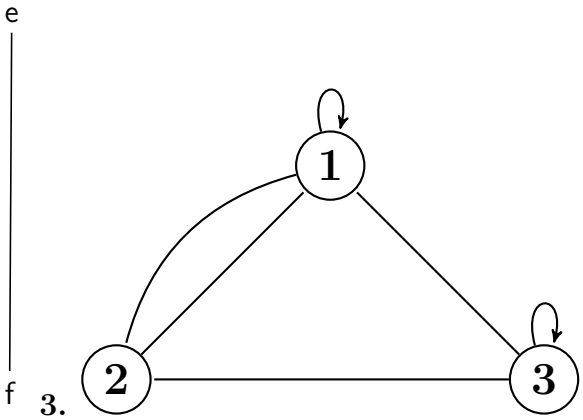
1.



2.



3.



1. Euler circuit, Eulter path, or neither?
Why?

Can you draw the circuit/path if there is one? (Do so on the graph)

2. Euler circuit, Eulter path, or neither?
Why?

Can you draw the circuit/path if there is one? (Do so on the graph)

3. Euler circuit, Eulter path, or neither?
Why?

Can you draw the circuit/path if there is one? (Do so on the graph)

In the questions above, we asked you to draw the Euler circuits/paths without any prescribed method. For larger graphs, this is actually quite hard to do from guessing—however, there is a method known a *Fleury's algorithm* for doing this (we skip this in favor of learning some more interesting things later, but when we input graphs into Mathematica, this is the method that the software uses).

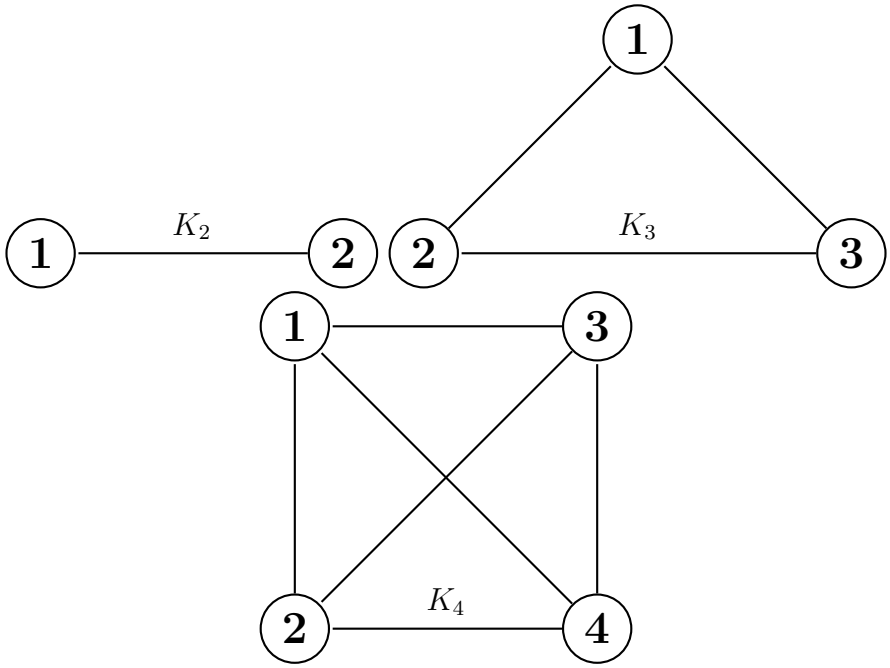
2.2 Other Results from Graph Theory

Before we move onto programming with graphs, we discuss a few other results from graph theory. The first term we will define is a *complete graph*, which is a graph in which each node is adjacent to every other node in the graph. Can you guess how many edges are in this graph?

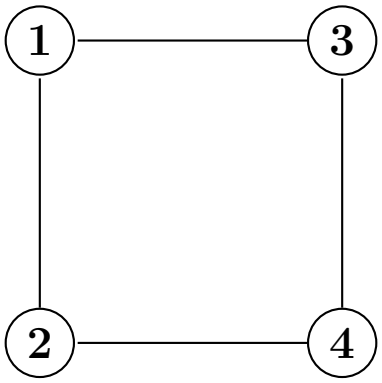
To start you off, there are n nodes in the graph. How many edges do each of these nodes share? _____. After we account for the fact that this counts each node twice, we see that there are _____ edges in the complete graph.

The complete graph on n nodes is often denoted by K_n. As we show below, K_2 is just an edge shared by two vertices, K_3 is a triangle, and K_4 is a rectangle with diagonals. These graphs have $2(2-1)/2 = 1$, $3(3-1)/2 = 3$, and $4(4-1)/2 = 6$ edges, as we show below.

In addition to compelte graphs, we have *regular graphs*, graphs in which each vertex has



the same degree. Regular graphs may or may not be complete, but complete graphs are always regular—for example, each node in K_3 has degree 2 and each node in K_4 has degree 3. Below, we show the 2-regular graph on 4 nodes, which is simply a rectangle.



Exercise 2.4

1. Draw a 1-regular graph on four vertices. Is it connected?

2. Is there a 1-regular graph on three vertices? If not, can you explain why?

3. Draw K_5. How many edges does it have?

4. Does K_3 have an Euler circuit or path? What about K_4? K_5? Can you make a statement about which complete graphs have Euler circuits/paths and which do not?

2.3 Computing with graphs in Mathematica

At this point, most of the tools we will learn are more relevant to larger graphs and data analysis, which makes paper computation difficult. Therefore, we're going to learn some basic graph computation in Mathematica. In this section, we cover how to input a graph, how to draw a graph, how to compute the degrees of a graph's vertices, and how to compute it's Euler circuits/paths.

An *edgelist* is the primary tool for graph data in mathematica. It is simply the list of edges in the graph. For example, an edgelist for K_3 would be $1-2$, $1-3$, $2-3$. To input this in Mathematica, we type "Graph[1¡-¿2,2¡-¿3,1¡-¿3]". We may also explicitly list the vertices– say we want to input the previous graph, but also list a fourth vertex not connected to the other three. Then we could input "Graph[1,2,3,4,1¡-¿2,2¡-¿3,1¡-¿3]". In general, the format is "Graph[$v_1, v_2, ..v_n$,$e_1, ..., e_m$]". We may also modify the edge and vertex colors, as well as list the vertex names i.e. "Graph[1,2,3,4,1¡-¿2,2¡-¿3,1¡-¿3,VertexStyle-¿Green,EdgeStyle-¿Blue,VertexLabel-¿"Name"]". We could also use letters or names instead of numbers i.e. "Graph[A¡-¿Francesca, B¡-¿Katrina,B¡-¿A]".

We can save the triangle graph by inputting "g="Graph[1,2,3,4,1¡-¿2,2¡-¿3,1¡-¿3]". This

allows us to get the degrees of each vertex by inputting "VertexDegree[g]".
Mathematica 'knows' certain types of graphs–i.e. the octahedral graph, a 4-regular graph on 6 vertices. To find its Eulerian cycles, first type "g=GraphData["OctahedralGraph"];" and then "FindEulerianCycle[g]".

Exercise 2.5
1. The Harary graph $H_{k,n}$ is a connected graph in which each of n nodes has at least degree k—this is very similar to a k regular graph. In many cases, they are equivalent, but in cases where the k regular graph on n vertices does not exist (i.e. number 2 of the previous exercise set)–in these cases, the Harary graph is the closest thing. First, construct the Harary graph $H_{3,5}$ by inputting "g=HararyGraph[3,5]". Find the degrees of its vertices and draw below, listing these degrees.

2. Construct the complete graph on 7 vertices by inputting
"g=Completegraph[7,VertexLabel− >"Name"]". Find one of its Eulerian cyclces. To show the steps of the cycle, type "Table[HighlightGraph[G,Part[First[%],1;;i]],i,Length[First[%]]]". What is the first edge traversed?

The last?

At this point, students should complete the "Graph Theory Additional Questions" worksheet.

3 Network Theory

In the previous section, we presented a quick survey of classical graph theory. We hope this was interesting in its own right, but here we change our direction and shift more to the application-based study of graphs, which is often called *network theory*. Most of the exercises for this section will be in mathematica, as the computations involved are relatively hard and sometimes require *probabilistic calculations*.

3.1 Random Graphs

Thus far, we have considered regular and complete graphs—while these are new to you, they are relatively boring to the well-studied mathematician. Most network studies involve some variation of *random graphs*, which are graphs generated from probabilistic calculations. For example, a particular random graph might be generated by placement of 10 edges at random between 8 nodes, so that nodes A and B share an edge with the same probability as any other two nodes in the graph. This type of graph, in which m edges are randomly placed between n nodes, is called an Erdős-Renyí (ER) graph, and is denoted by $G(n, m)$. This

is, by standards of network scientists, a very simple model—and so most applications make additional assumptions not present in ER graphs. However, there are some applications of ER graphs in their own right–for example, one might use an ER graph to study the spread of a contagion wherein a few people (nodes) are infected, and the disease spreads along the network edges.

However, ER graphs are really just the start of random graphs. Two other types of commonly used random networks are referred to as graphs from the *Barabási-Albert (BA) model* and the *Watts Strogatz (WS) model*.

Graphs of the BA type are generated from a 2-regular graph, adding n nodes and k vertcies at each step, but in such a way that the probability of an edge being added to a particular vertex is proportional to that vertex's degree at the step. This process creates a wide *degree distribution*, such that some nodes have very high degrees while others have very low degrees—in particular, BA graphs have been shown to be good models for many applications. The most famous example of this is in internet linkages across the web—highly visited and linked to webpages will tend to get more and more traffic over time, so that yet more sites will link to them i.e. *the rich get richer*. If a site is less linked to, then fewer visitors will see the site and link to it in the future. This creates the wide degree distribution seen in BA networks. Another example relevant to the BA model is the use of different food items in recipe databases (See Figure 4).

We won't cover the WS model here, but feel free to plot it in mathematica to get an idea of what kind of graphs it creates if you like.

Exercise 3.1 1. Create an Erdős-Renyí graph of 20 edges on 18 nodes $G(18, 20)$ in Mathematica by inputting "g1=RandomGraph[{18,20}]". Draw your graph here, and compare with someone around you.

2. Compute the average of the degrees of the vertices in your graph $G(18, 20)$ (usually we call this the *mean degree*). Is this the same as what people around you got?

3. Can you come up with a formula for the mean degree of $G(m, n)$ given m and n.

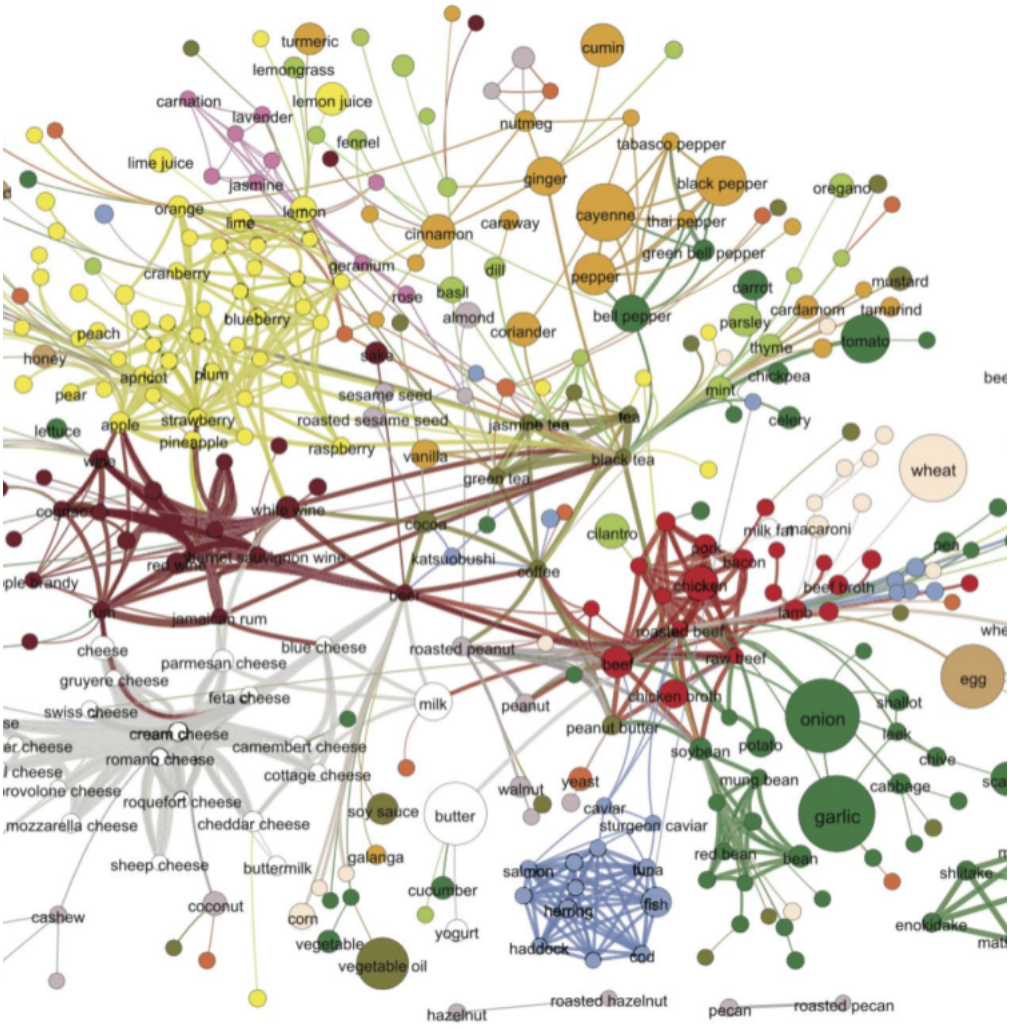


Figure 4: The size of each node represents how many recipes it is given in, while the thickness of an edge between two nodes represents the number of recipes featuring the two adjacent nodes together (Figure from blog: ScienceandFood).

4. Also, construct a graph $g2$ on $G(18, 5)$. Supposing this graph and $g1$ were infection networks wherein node 5 represented an infected person, and disease was transmitted along edges, how many nodes would be infected in each graph? Which indicates a 'safer' network in this context?

5. Construct a BA graph on 10 vertices by inputting "g3=RandomGraph[BarabasiAlbertGraphDistribution[10,3]]". How does this compare in your own words to the random graph $g1$?

We can give better illustration of the *degree distributions* by showing them in plot form. This is a bit advanced for those who have not had a statistics course, but hopefully we can teach you what these plots mean. Look at Figure . The horizontal axes represent the degree d. See that this varies between and. The vertical axis represent the *Probability of having degree d*. You can see that one either plot, there is a high probability that a node has degree $d =$, but this is much higher on the right plot.

3.2 Other Network Properties: Centrality and Clustering Coefficient

We have already seen several important network properties in action—the number of components tells us 'how many pieces' a network comes in. The mean degree tell us how densely connected a network is. And the degree distribution gives us an idea of the variation within the network. A different kind of property, called *centrality*, tells us about the relative importance of each node within its network.

The first kind of centrality is very simple and intuitive. A node's *Degree Centrality* is simply the degree of that node d, divided by its maximum possible degree, which is $m - 1$, where m is the number of edges in the network. A high degree node is usually an important player in the network–for example, garlic's use in the food network above has a high degree because it is used frequently in lots of networks. In friendship networks, the most popular units have the highest degrees, etc.

Now, to get an idea of the second type of centrality, think of Washington DC's subway system (or any other city's subway system). There are certain stops that several lines go through, and consequentially many passengers on different lines go through these stops (see Figure 5). A node's *Betweenness centrality* captures this, by taking into account how many

shortest paths between different nodes in the network pass through the node of interest.

Figure 5: If we consider the network wherein each stop in a node, and the paths between them are edges, nodes like Gallery Place (or L'Enfants Plaza) are important because they lie on the shortest path of many routes (ie from Judiciary Square to Metro Center, or to Archives). Gallery Place also has a relatively high degree, but betweenness centrality better captures its importance in the network, and would be more useful to engineers considering subway station repairs/closures.

Unfortunately, this calculation is much longer and more complicated. For a connected network, it is defined as the number of shortest paths passing through the node, divided by the total number of shortest paths in the network, *taking into account every pair of nodes in the network.* We will calculate this for one small network, and then let mathematica take it from there.

Examine Figure 6, and consider node 1–what shortest paths pass through it? Well, a shortest path from 2 to 5 passes through it (2-1-5), and this is the only shortest path from 2 to 5–this contributes $1/1 = 1$ to the calculation.. The same can be said for 3 to 5. As well, both paths from 4 to 5 pass through 1 (the 'top' way 5-1-3-4 and the bottom way 5-1-2-4), and this also contributes $2/2 = 1$. How about shortest paths form 2 to 3? One (2-1-3) of these goes through 1 while the other goes through 4 (2-1-4), so this contributes $1/2$ to the calculation.. Do any paths from 3 to 2 pass through it? No–therefore, we've considered all pairs of nodes and the shortest paths passing through 1. Adding up the contributions we found, we get

$1+1+1+1/2 = 3.5$. Now we divide this by the number of pairs in the graph which exclude 1—there are six of these (5-3,5-2,5-4,2-3,2-4,3-4), and find that the betweenness centrality of 1 is $3.5/6 = 0.58$. There are *many* other types of centrality, but only so little time (and

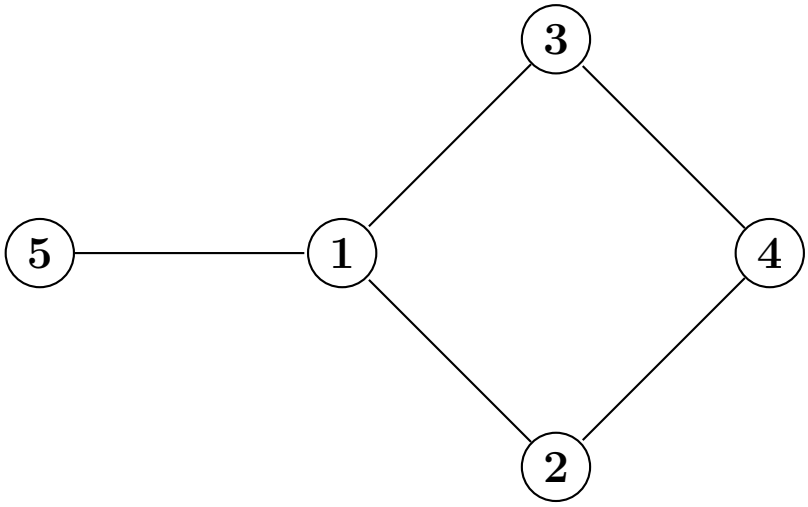
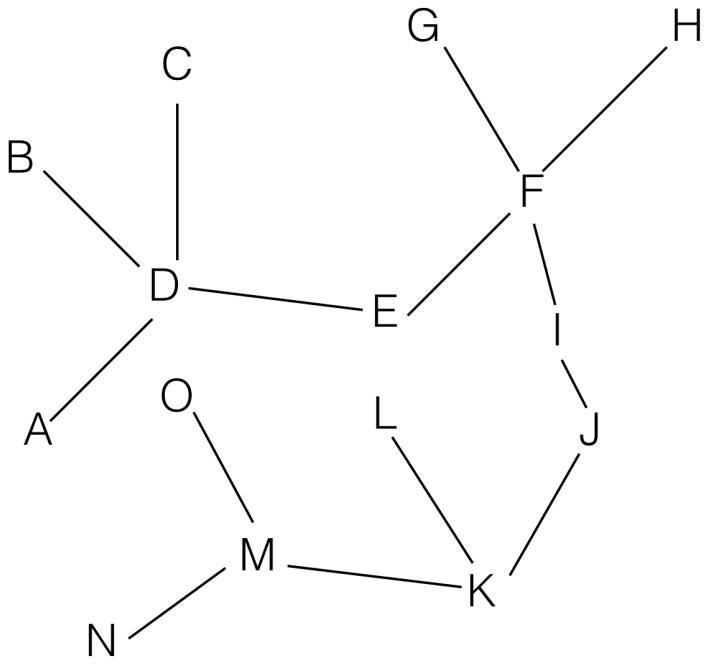


Figure 6: Betweenness Centrality example

many of these methods require exposure to math that most of you have probably not yet learned)—indeed, Google uses a variant of something called 'eigenvector centrality' to rank the results of our web searches. However, these two kinds of centrality we've covered are among the most important and will suffice for our study.

Exercise 3.2

1. Consider the graph below. Which node(s) have the highest degree centrality? Which do think has/have the highest betweenness centrality?



2. Now, input the graph into mathematica. Find which nodes have the highest betweenness centrality (if you named your graph g, you can input "BetweennessCentrality[g]"). Was your guess correct?

3. Plot the degree distribution (don't bother drawing here)...Does it look more like a graph from the ER model or a graph from the BA model?

3.3 Community Detection

The last thing we will learn about here is community detection in graphs. Often, when confronted with data, we look for some kind of underlying structure and organization to the data—what parts of the data cluster together? Are there groups of similar data points, or does everything seem random? A powerful method for deducing structure is *community detection*. This can be used in many applications—for example, to find cofunctioning units of neurons within the brain, to find social cliques from Facebook data, to find voting blocs in Congress, to find organization from criminal ties (see Figure 7), etc.

Often, community organization is easy to see in some data—communities occur when some nodes have more edges between them than others. But some networks have millions (or more) of nodes and edges—even in smaller networks, community organization may be difficult to see if the graph layout is poor. For example, see Figure 8. The left and right communities (purple and blue) seem randomly chosen, and furthermore the graph seems like

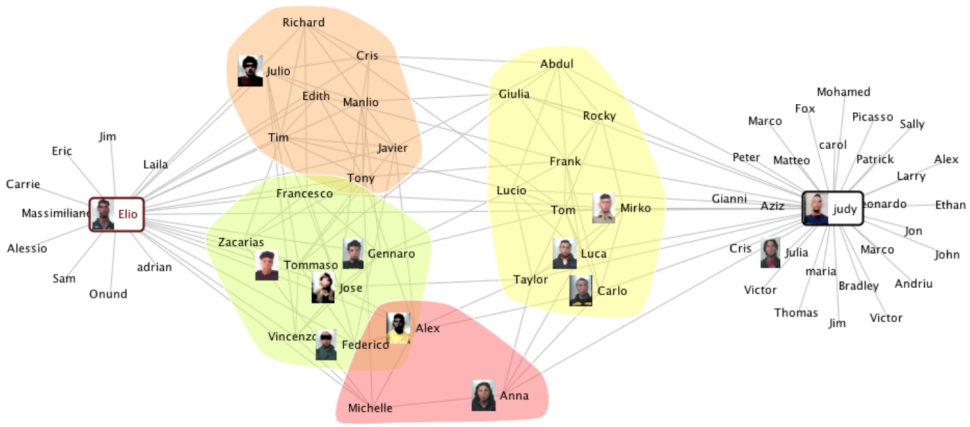


Figure 7: Network Information Reconstructed from records of cell phone calls between frequent offenders, *from Prof. Emilio Ferrarra's personal webpage*

it may just be an ER graph. However, with a different layout, we see that the graph is actually quite *modular* (it has well defined communities). Can you guess what the layout would be that maximizes the community structure?

Without any prior experience or tools, this may not be straightforward even in this simple graph. A proxy for 'good community structure' is looking for a community assignment such that there are few edges between (not within) the different communities–looking at Figure 8, this would mean most of the edges should be within the two circles, not between them. This happens if we group together A, B, C, D and I into group, and E, F, G, H, and K into the other.
Redraw the graph with these assignments here. Circle you communities as in Figure 8, with A, B, C, D, and I on the left and E, F, G, H, and K on the right.

Indeed, community detection is a very active area of research, particularly here at UNC. One method is *modularity*-based community detection. Like many algorithms, this method relies upon optimizing (trying to increase) the output of a particular function. Intuitively, this function represents *how well a certain community structure fits the given data*—more specifically, modularity is defined as the *fraction of edges falling within (not between) communities, minus what would be expected in a random graph.* In favor of doing some applications,

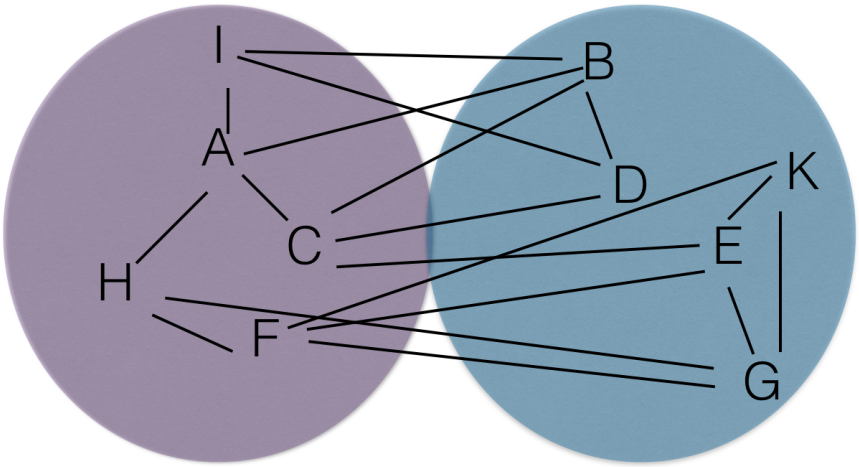


Figure 8: If we choose the communities as so (the blue and purple being the two communities), the community assignment tells us nothing.

we won't derive the actual modularity function itself. Mathematica can calculate the communities by optimizing this function for us.

In mathematica, input "g=ExampleData[{"NetworkGraph","ZacharyKarateClub"}]"—this data (frequently used in the community detection literature) represents a social network wherein nodes represent people and edges represent friendships within a karate club. Now input "FindGraphCommunities[g,Method− >"Modularity"]" and then "HighlightGraph[g,Subgraph[g,#]&/@%,GraphHighlightStyle− >"DehighlightHide"]" to see the communities in this network.
If this doesn't work, try "CommunityGraphPlot[%,FindGraphCommunities[%]]".

Exercise 3.3 Dolphins, like humans, are social animals, and animal behaviorists frequently study how they interact—as this has great implications to preserving their populations and those of other species. There is a great data set on the internet showcasing these studies—in the form of a simple network wherein there is an edge between two nodes (dolphins) if they frequently interact. We will investigate this network here.

First, download the data set 'dolphins.gml' at http://networkdata.ics.uci.edu/data/dolphins/
Now, import the data by typing into mathematica "dolphins=Import["/pathtofile/dolphins.gml",VertexLab
(pathtofile is the path to the directory in which the file is located—you may need to ask the instructor for help with this).
1.Now, find the communities and visualize them, using the methods described above. How many are there?

2.What is the mean size of a community in this network?

3. Can you say anything else about the community structure of dolphin social networks?

4. Suppose there was a conflict among the different communities, such that dolphins stopped interacting with dolphins of other communities. You can show what happens in this case by inputting:
"HighlightGraph[dolphins,Subgraph[dolphins,#]&/%,GraphHighlightStyle→"DehighlightHide"]".
Describe what happens, and interpret this in terms of the dolphin social network.

5. Are any nodes more central than others (using degree or betweenness centrality)? What would this mean in terms of animal behavior?

6. Plot the degree distributions. Does the graph resemble a BA network, an ER network, or neither?