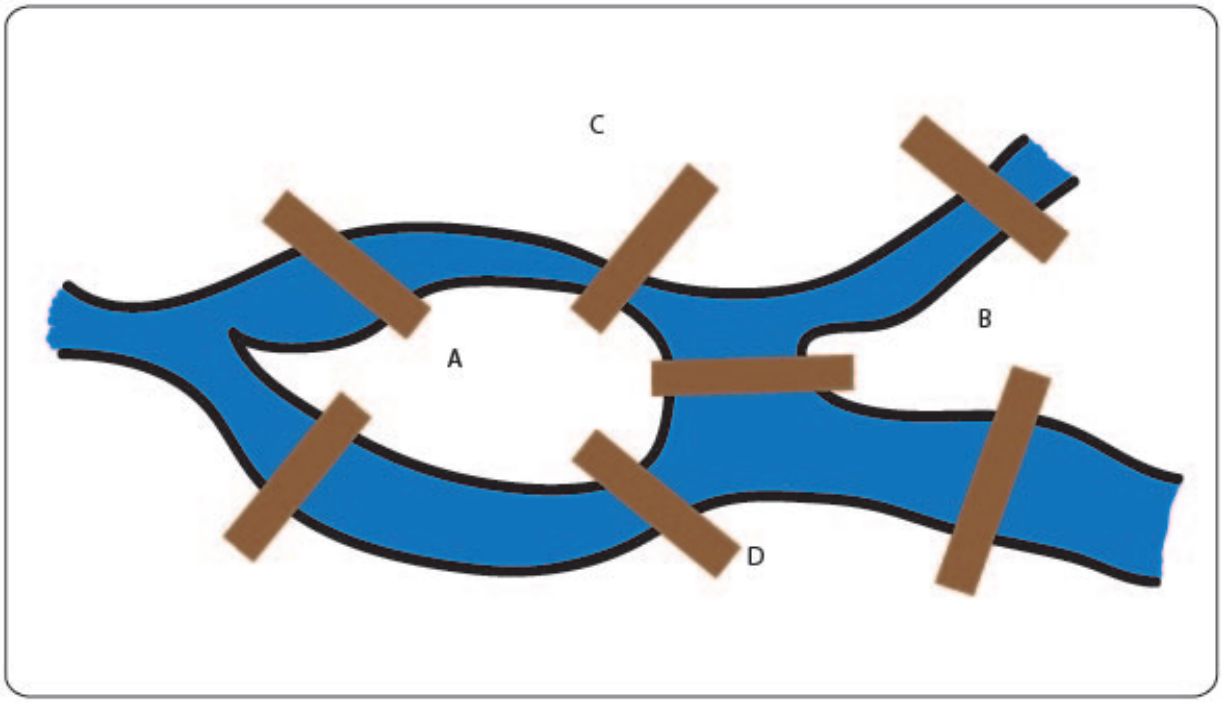
UNIVERSITY OF WATERLOO
Faculty of Mathematics
Waterloo, Ontario N2L 3G1

Centre for Education in
Mathematics and Computing

Grade 6 Math Circles
Graph Theory
November 17/18, 2015

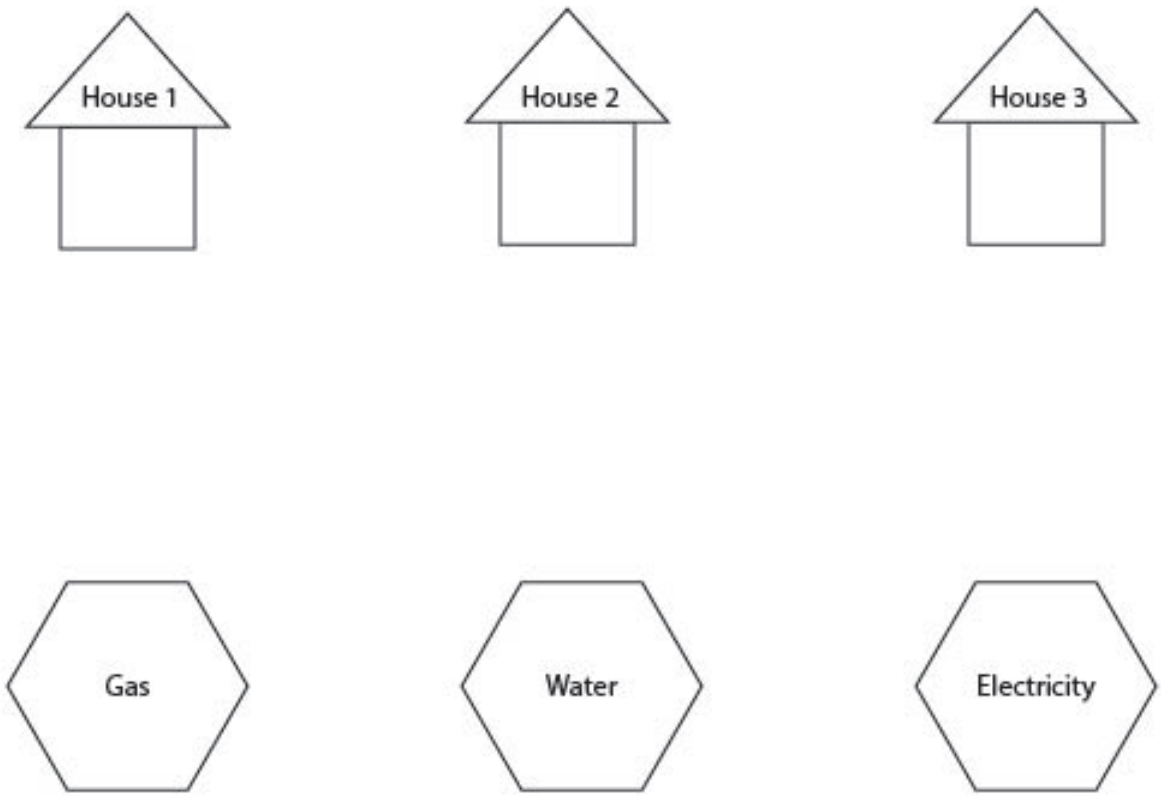
The Seven Bridges of Königsberg

In the mid-1700s there was a city named Königsberg. Today, the city is called Kaliningrad and is in modern day Russia. However, in the 1700s the city was a part of Prussia and had many Germanic influences. The city sits on the Pregel River. This divides the city into two main areas with the river running between them. In the river there are also two islands that are a part of the city. In 1736, a mathematician by the name of Leonhard Euler visited the city and was fascinated by the bridges. Euler wondered whether or not you could walk through the city and cross each bridge exactly once. Take a few minutes to see if you can find a way on the map of Königsberg below.



The Three Utilities Problem

Suppose you have a neighbourhood with only three houses. Now, each house needs to be connected to a set of three utilities (gas, water, and electricity) in order for a family to live there. Your challenge is to connect all three houses to each of the utilities. Below is a diagram for you to draw out where each utility line/pipe should go. The catch is that the lines you draw to connect each house to a utility can't cross. Also, you can't draw lines through another house or utility plant. In other words, you can't draw House 1's water line through House 2 or through the electricity plant.



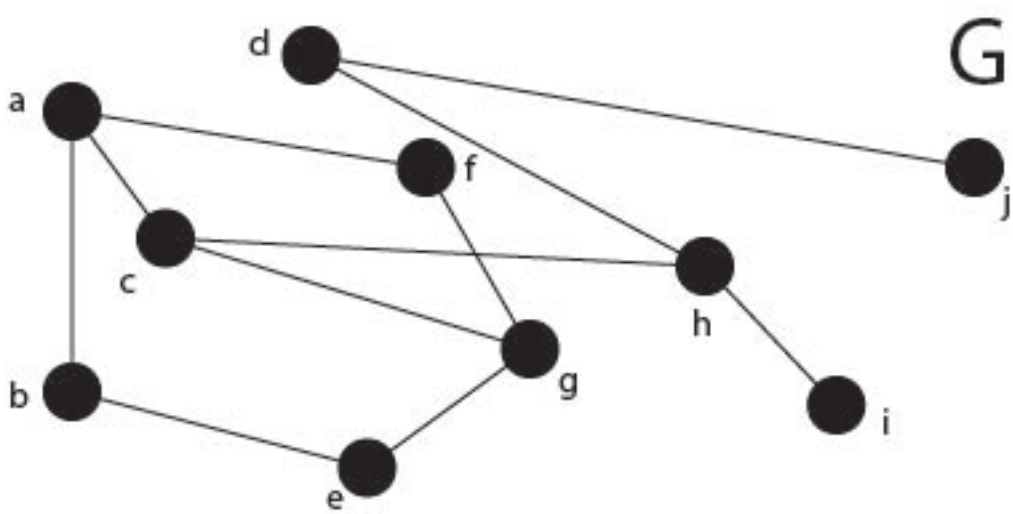
Introduction to Graph Theory

Both of these problems are examples of Graph Theory. Graph Theory is a relatively young branch of mathematics, and it was Euler's solution to the 7 Bridges problem in 1736 that represented the first formal piece of Graph Theory. But what is Graph Theory? In order to answer that question and to explore some of the applications of Graph Theory, we first need some definitions.

Definitions

- A **vertex** (plural: **vertices**) is a point. In drawings, we usually represent these as circles.
- An **edge** e is an unordered pair of vertices. For example, $e = a, b$. In drawings, edges are represented by lines between the two vertices.
- For an edge $e = \{a, b\}$, we call a and b the **endpoints** of edge e
- A vertex a is **incident** with an edge e if a is an endpoint of e
- Two vertices a and b are **adjacent** if there is an edge e with a and b as its endpoints
- The vertices adjacent to a vertex a are called the **neighbours** of a
- A **graph** $G = (V, E)$ is comprised of a set V of vertices, and a set of edges E.
- A **walk** is a sequence of vertices where each vertex is adjacent to the vertex before it and after it
- A **path** is a walk that doesn't repeat vertices
- A **cycle** is a path that begins and ends at the same vertex

These definitions can be difficult to understand as abstract concepts. To make these definitions more concrete, consider the graph G below:



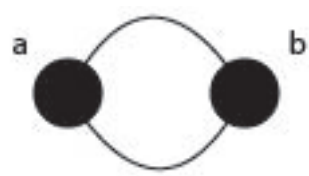
Example 1

1. List the vertices in G
2. List the edges in G
3. List the edges incident with vertex c
4. List the vertices adjacent to g
5. Find a path from j to b
6. Find a cycle in G

Graph Rules

In different applications, graphs can have a wide variety of rules associated with them. For example, in some applications graphs can be *directed* so they only go from a to b but not from b to a (think of it like a one-way street). For the graphs in this lesson we're going to focus one graphs that satisfy these rules:

- Our graphs will be undirected (i.e. like a two-way street)
- Graphs will not contain duplicate edges. What this means is there will only be one edge from a to b. We will not have situations like the one below:



- Graphs will not have self-loops (edges that start and end at the same vertex). Below is an example of a self-loop:

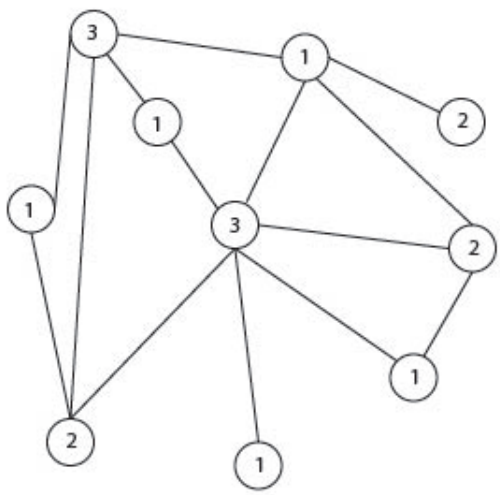
Colourings

How many colours does it take to colour a map of the United States? What about a map of Africa?

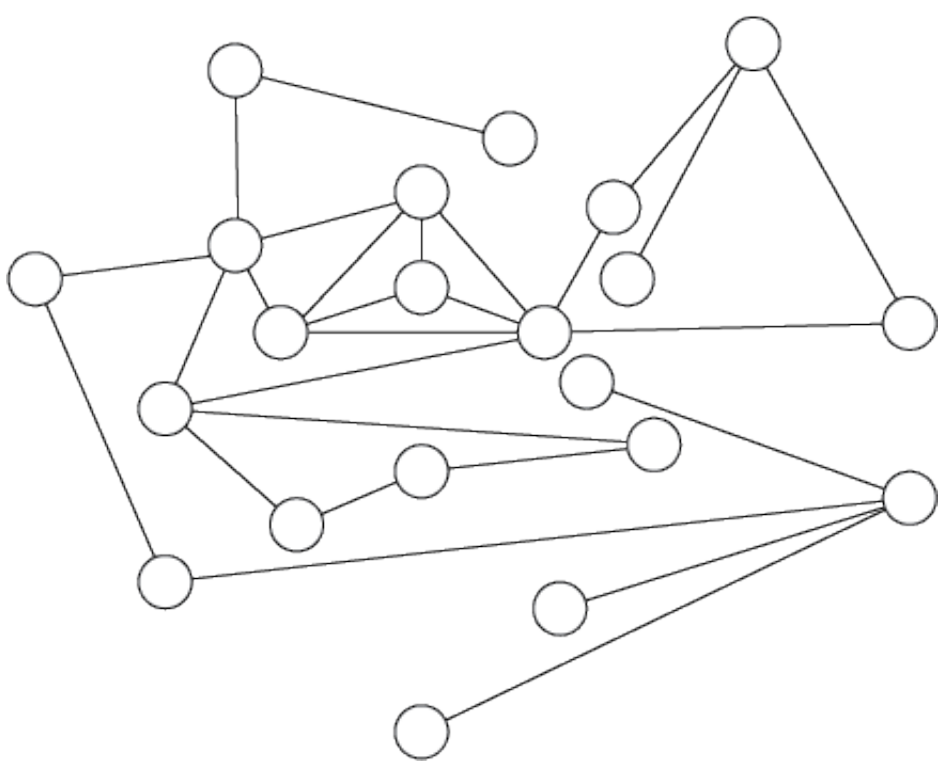
While it may not be obvious, this problem (a more general form of it) has been one of the central problems in Graph Theory for a very long time. It turns out that we can colour any planar graph in just *four* colours. In fact, this problem was initially posed in 1852, and a correct proof was not submitted until 1976, over 100 years later.

But what does it mean to colour a graph? It's easy to understand how to colour a map, we simply colour each region a different colour than it's neighbours. It turns out that colouring a graph is essentially the same thing. We assign each vertex a different colour than the vertices adjacent to it. Even though we talk about colours like red or green, we usually just label the vertex with a number, and each different number represents a new colour.

Below is an example of a colouring. Notice that this graph can be coloured using just three colours.

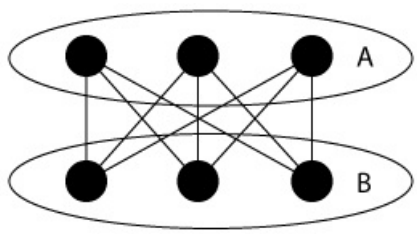


Try colouring the following graph with as few colours as possible. How many colours did you need?



Colouring Bipartite Graphs

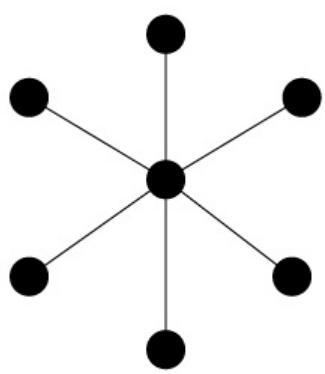
A **bipartite graph** is a graph where we can split the vertices into two groups, A and B, and all of the edges have one endpoint in A and one in B. This means that for any vertex in A, all of the vertices it is adjacent to are in B. One example of a bipartite graph is $K_{3,3}$. We can see that if we call the top row of vertices A and the bottom row of vertices B, then all of the edges go from A to B, and none from A to A or B to B.

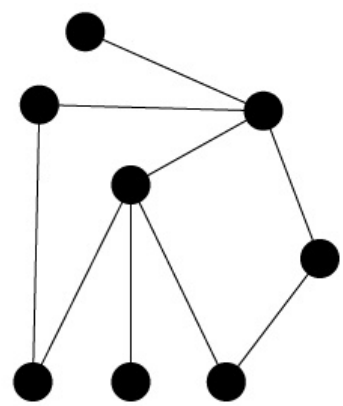


But bipartite graphs don't have to have their groups of vertices spaced as nicely as the above graph.

Exercise 2

While they might not look like it at first glance, both of the following graphs are bipartite - can you find the groups of vertices A and B?





Think about colouring these graphs. Is there anything special that you notice?

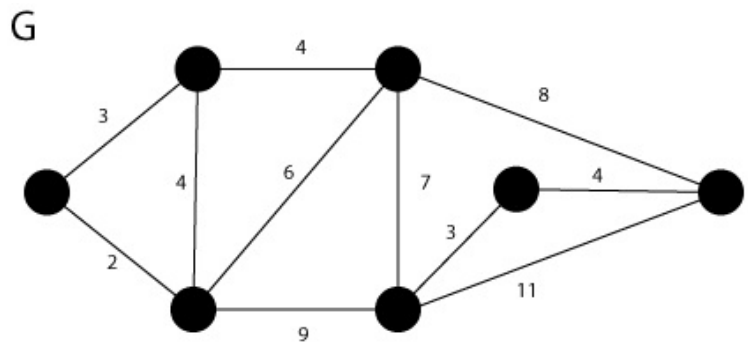
Weighted Graphs & Minimum Spanning Trees

So far, we've only dealt with unweighted graphs. Now we are going to look breifly at weighted graphs. In a weighted graph, each edge is given a numerical **weight**. These weights can represent many different things depending on the problem.

Today, we will be looking at **minimum spanning trees** and how we can construct these trees. A **tree** is simply a graph with no cycles. A **spanning tree** is a tree that contains all of the vertices of the original graph. A **minimum spanning tree** is a spanning tree with the smallest total weight of *all* spanning trees. We can find the weight of a spanning tree by adding up the weights on each of its edges.

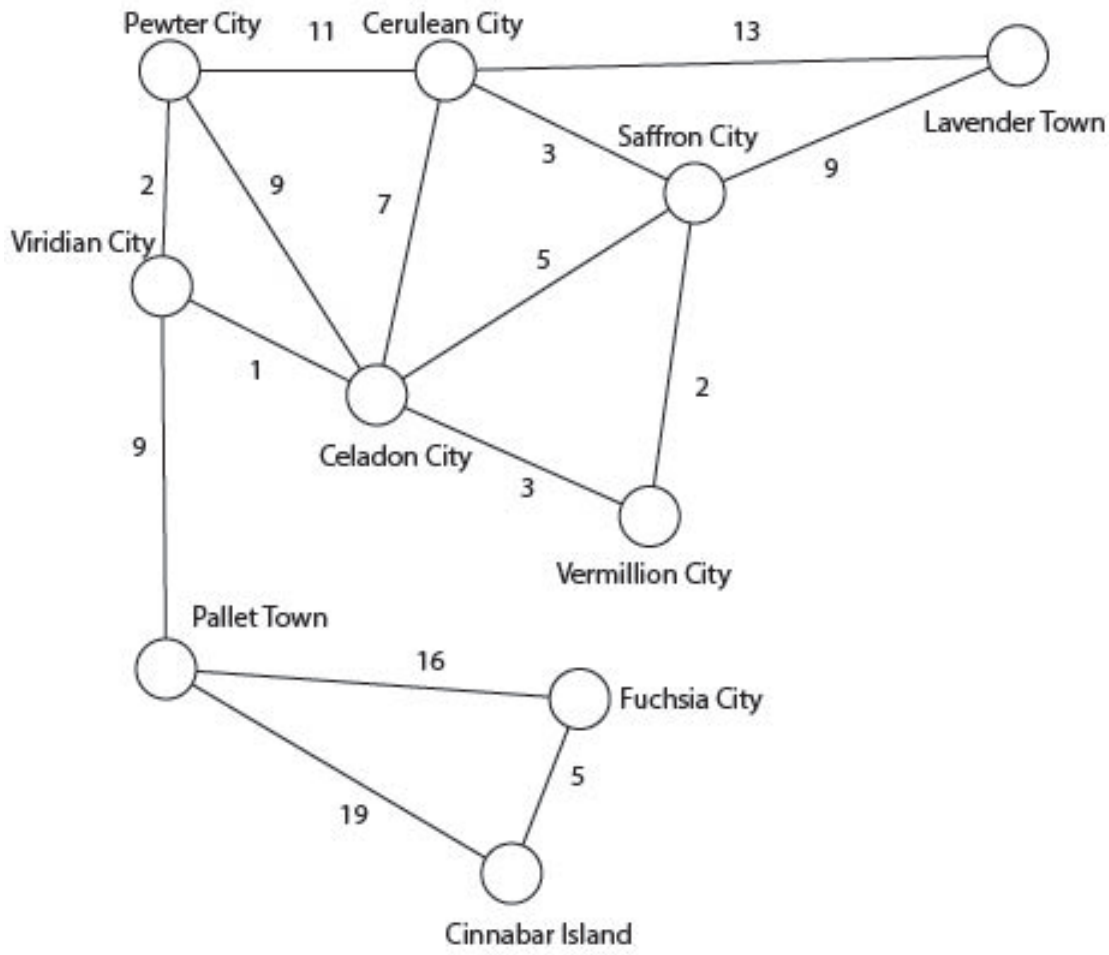
Exercise 3

Consider the graph G below.



1. True or False: G is a tree
2. Find any tree in G
3. Draw any spanning tree of G. (Highlight or circle the edges of G that you are including in your spanning tree)
4. Calculate the weight of your spanning tree.
5. Try and find another spanning tree with a lower weight

One particular application that we will be looking at is building a train system to connect a group of cities. Our goal is to do this for as low a cost as possible. The graph below is a map of the region. The vertices represent cities, and the edges represent potential railroad tracks. The weights on each edge represent the cost to build a railroad between the two cities.



Exercise 4

On the graph above, try and connect all of the cities with railroads for the lowest total cost.

Think about what edges you had in your railroad system, then answer the following questions about your approach to this problem.

1. Were there any edges that you definitely wanted to include?
2. What about ones you wanted to avoid?
3. How did you pick which edges to include?

Prim's Algorithm Instead of simply trying every possible combination, mathematicians has developed an *algorithm* or series of steps to follow to determine the minimum spanning tree. One such algorithm is known as Prim's Algorithm. We will follow the steps of this algorithm to find the railway system with the lowest cost.

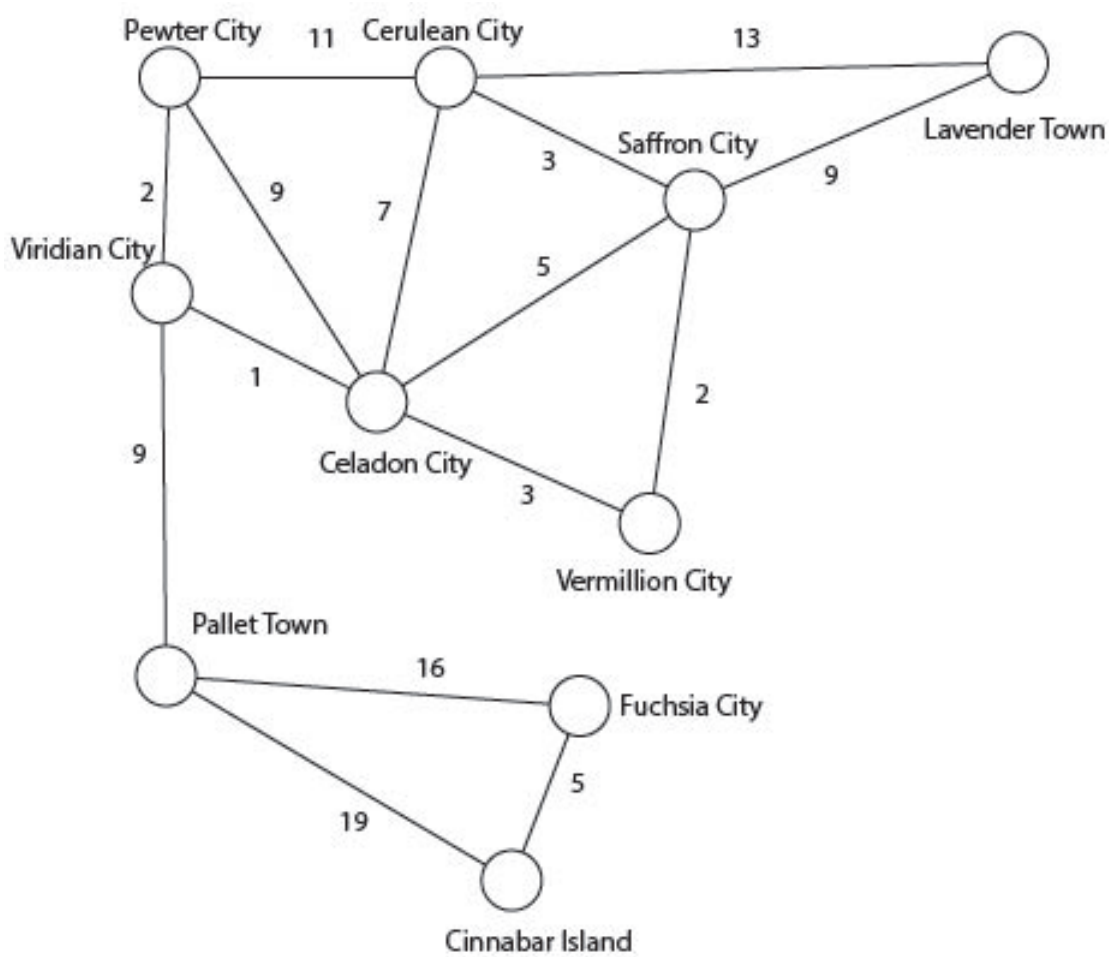
Step 1. Select a starting vertex (In our example, we will choose Pallet Town)

Step 2. Call S the group of all the vertices currently connected to your tree. Call V all the other vertices. (Note that the first time you reach this step, S will just be your starting vertex)

Step 3. If S contains all of the vertices in the graph, stop. You have your minimum spanning tree!

Step 4. Find all of the edges with one endpoint in S and one endpoint in V. Select the edge with the smallest weight and add it to your tree. If there is more than one edge with the smallest weight, you can pick any of them to add. (For now, to add an edge to your tree, just circle or highlight the edge)

Step 5. Return to step 2



Problem Set

1. For each graph below, answer the following:

   (a) List the edges and vertices of the graph.

   (b) How many edges and vertices are there?

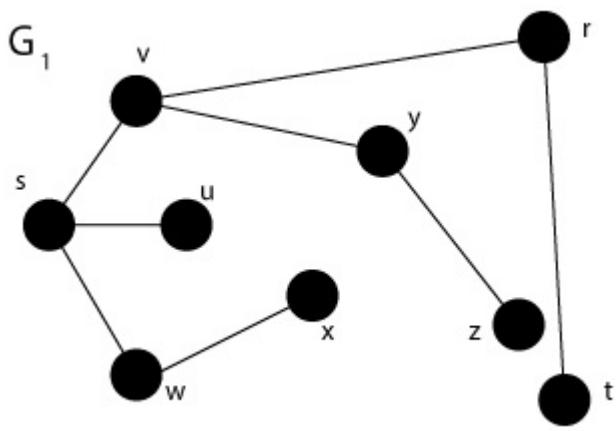
   (c) List the neighbours of the vertex v

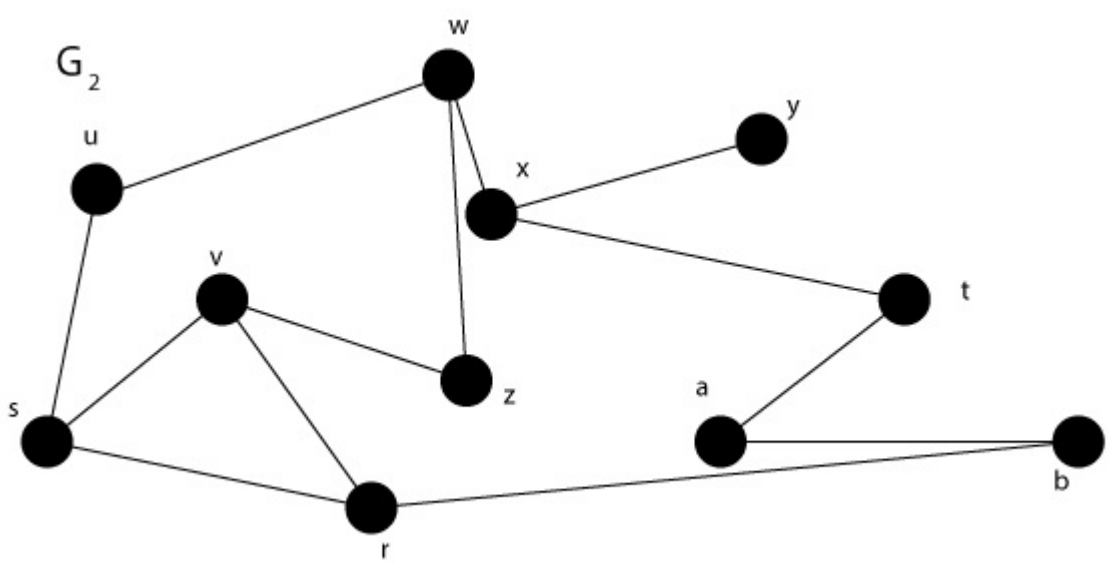
   (d) How many edges are incident with s

   (e) Find a walk between s and t. Is your walk a path? Why or why not?

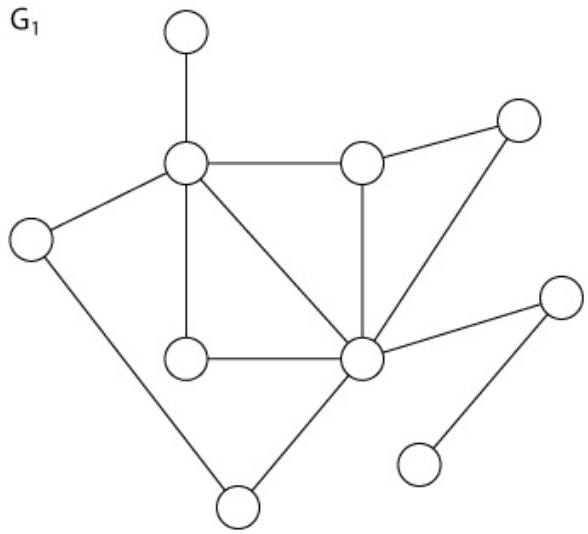
   (f) How long is the shortest path between s and t?

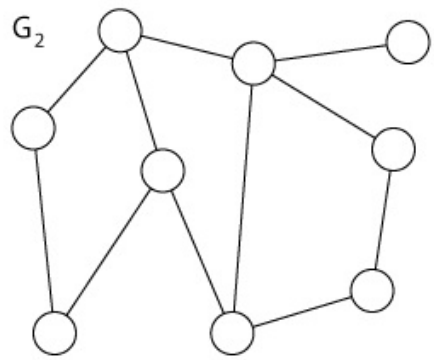
   (g) Is the graph a tree? if not, find a cycle.



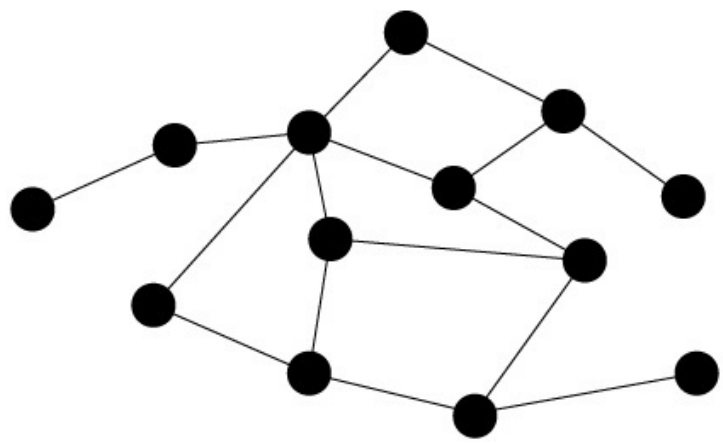


2. Colour the following graphs using the fewest colours possible. How many colours do you need?

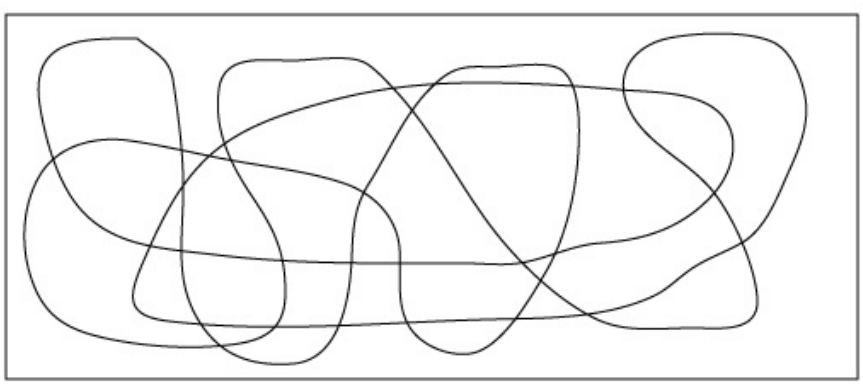




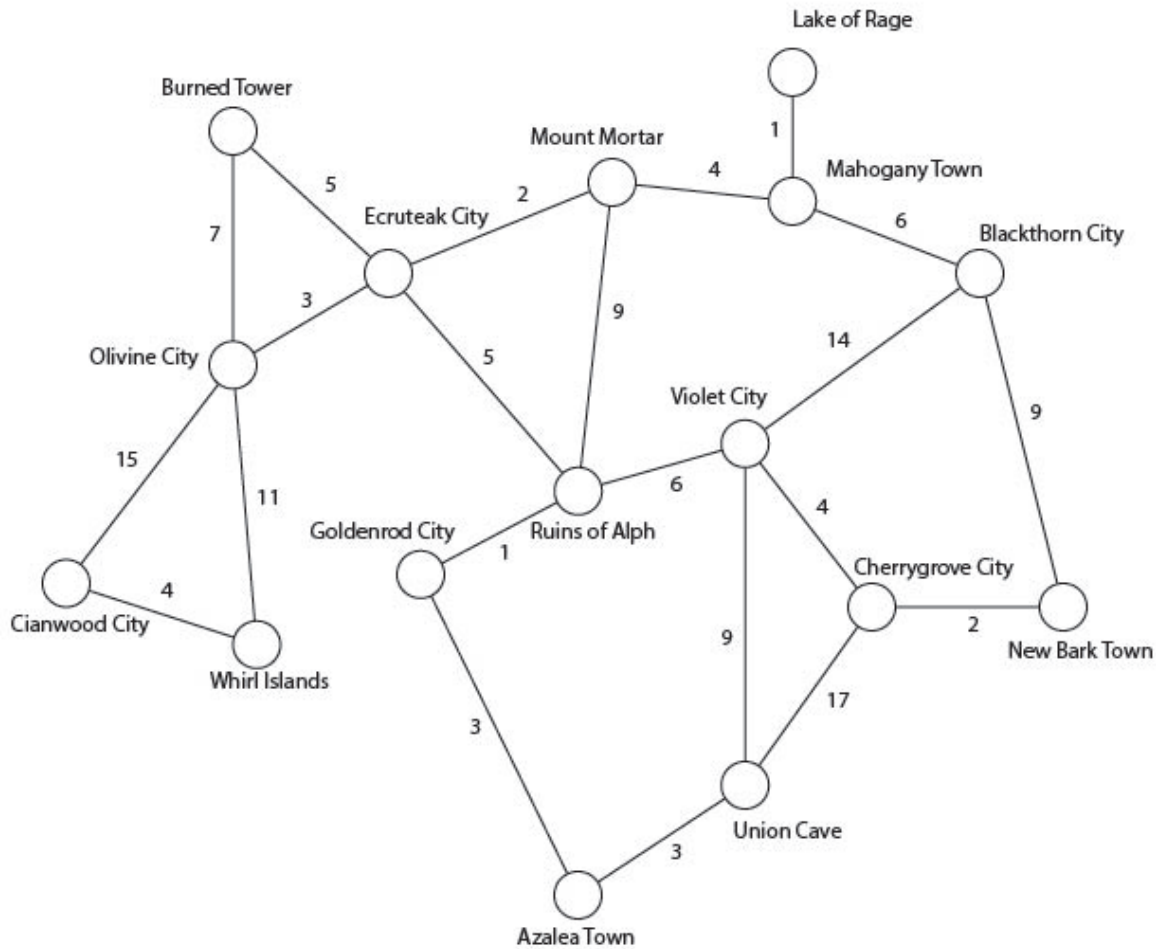
3. Determine whether the following graph is bipartite. If it is, find the groups A and B.



4. Colour the regions (including the outer portion) of this map using the fewest colours possible.



5. Using Prim's Algorithm, find a minimum spanning tree for the following railroad network. Use New Bark Town as your starting vertex.



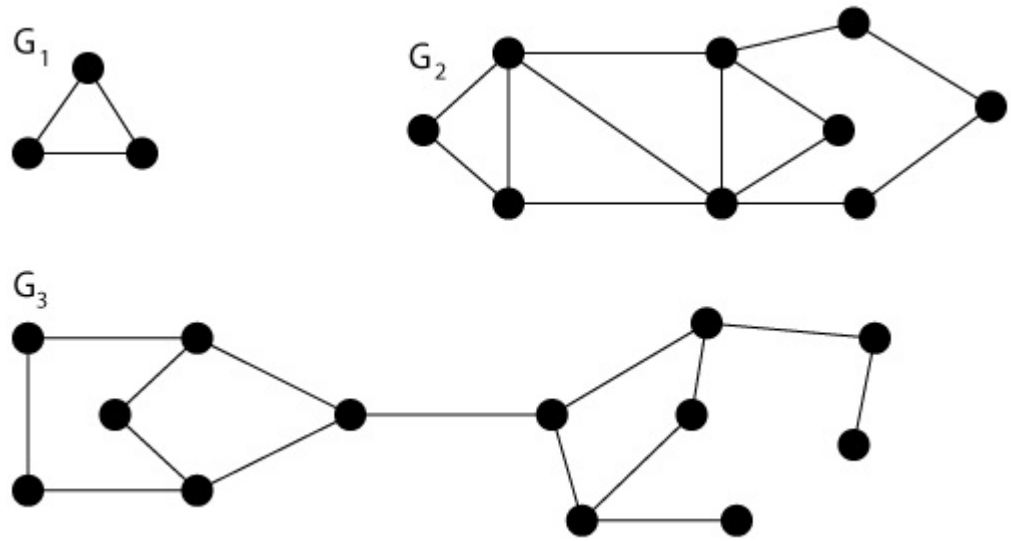
Challenge Problems

Matchings

Let $G = (V, E)$ be a graph. Let M be a subset of the edges of G. M is a **matching** of G if no two edges of M share an endpoint. For example, in the graphs below, the red lines are a matching, since no two red edges share an end point, but the blue edges do not form a matching.

The **size** of a matching is just the number of edges in it. Our goal is often to find a matching of maximum size. That is, we cannot make a larger matching.

1. Find a maximum matching in the following graphs:

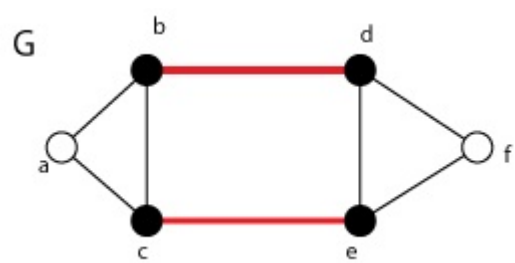


While it may be simple to check that a matching has maximum size on a small graph, what should we do on larger graphs?

To do this we introduce the concept of *saturation*. Given a graph G and a matching M, we call a vertex v **saturated** if it is the endpoint of one of the edges in M. If v is not an endpoint of an edge in M, then we call v **unsaturated**.

We want to know if the matching M is of maximum size. To do this, we want to find an *augmenting path*. A path P is an augmenting path if it begins and ends at unsaturated vertices, and the edges in the path alternate between edges not in the matching, and edges in the matching.

For example, in the graph below the red edges are the matching M. The black vertices are saturated, and the white vertices are unsaturated.
The path $P = \{\{a, b\}, \{b, d\} \{d, f\}\}$ is an augmenting path.



If an augmenting path exists, then M is not a maximum matching.

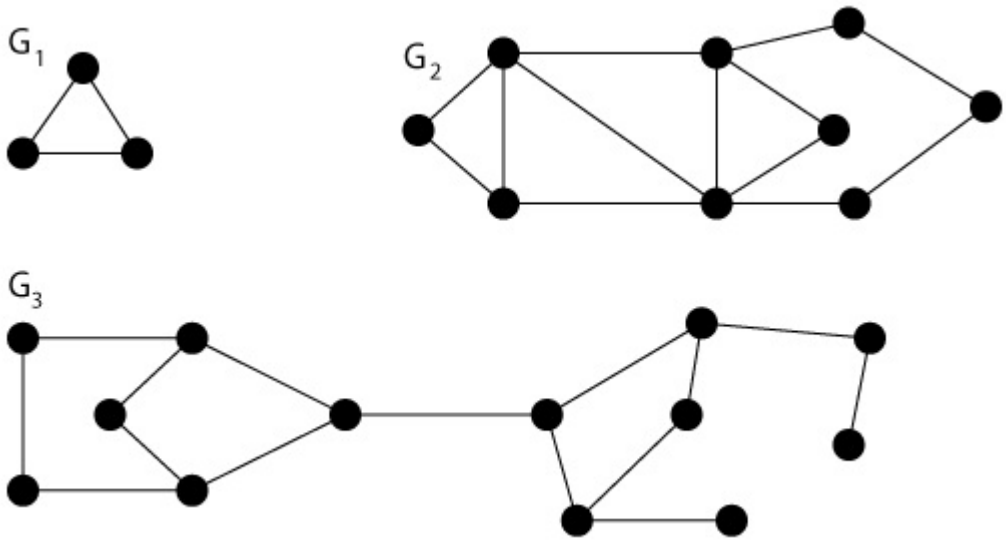
2. Why is M not a maximum matching if an augmenting path exists? How can we create a new, larger matching?

3. Using your knowledge of augmenting paths, confirm that your matchings from question 1 are maximium.

   Vertex Covers
   A vertex cover is a set C of vertices such that every edge in the graph has an endpoint in C. For example, the green vertices in the graph below are a vertex cover.

   Our goal is find a vertex cover with the smallest number of vertices.

4. Find a vertex cover with the smallest number of vertices in the following graphs



5. What do you notice about the size of the vertex covers you found and the size of the matchings you found earlier? Is one always bigger?

6. Explain why the size of a vertex cover will always be larger than the size of a matching.

7. What can we say about the a vertex cover and a matching that have the same size?