UNIVERSITY OF WATERLOO

Faculty of Mathematics
Waterloo, Ontario N2L 3G1

Centre for Education in
Mathematics and Computing

Grade 7/8 Math Circles

March 18/19, 2014

Graph Theory II

Review

- A **vertex** (plural: **vertices**) is a point that is usually labelled with a letter, number, or other title. (ex. A)
- An **edge** is a line that joins two vertices and is named in the format $\{-,-\}$, where " $-$ " is replaced by the vertices it is incident to. (ex. $\{A, B\}$)
- Two vertices are **adjacent** if there is an edge joining them.
- A vertex and edge are **incident** if they are joined.
- A **walk** is a sequence of vertices where each vertex is adjacent to the vertex before and after it. A walk can be written as "$vertex, vertex, vertex...$". (ex. $A \to B \to E \to A \to C$)
- A **path** is a walk that doesn't repeat vertices.
- A **cycle** is a path that begins and ends at the same vertex.
- Edges **can** be curved or straight (or any other shape).
- Edges **can** cross.
- The placement of vertices **doesn't** matter. Only the connections are important.
- Two edges **can't** connect the same vertices. (ie. There can only be a **maximum** of one edge between any two vertices.)
- An edge **can not** connect a vertex to itself.
- A graph is **planar** if it can be drawn with no edges crossing.

Handshake Lemma

The **degree** of a vertex is the number of edges incident to it (remember that this means the number of edges connected to the vertex). The **Handshake Lemma** can be stated as follows:

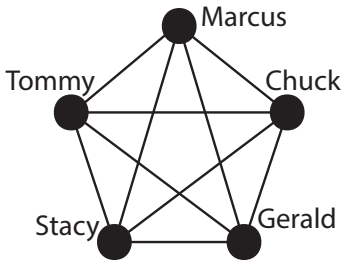
Theorem. *The sum of the degrees of all the vertices is equal to double the number of edges*

Let's think about why this is. Each edge connects to two vertices. That means that each edge is counted twice when we count the number of connections.

Exercise 1

1. Marcus, Tommy, Chuck, Gerald, and Stacy go to a party. They have never met before so they all shake hands when they get there.

   (a) Draw a graph to represent this situation. What does each vertex represent? What does each edge represent? Each vertex represents a person. Each edge represents a handshake.



2. How many times does each person shake hands? Each person shakes hands 4 times.

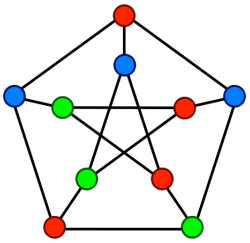
3. What is the total number of times a hand is shaken? (Note: It counts as two different hands being shaken when two people meet) There are 20 hands shaken.

4. How many handshakes take place? 10 handshakes take place.

5. How do parts (b) and (c) relate? Explain why this makes sense. The total number of times a hand is shaken is double the number of handshakes.

Colouring

A **colouring** is a way of labelling a graph. In a colouring, no two adjacent vertices can have the same colour. We can describe a graph by the minimum number of colours needed. A **k-colouring** (where k is replaced with a number) is a colouring with at most k colours. A graph is **k-colourable** if it has a k-colouring. Here is an example (from `en.wikipedia.org/wiki/Graph_coloring`):



This is the Petersen Graph with a 3-colouring. This is the minimum colouring possible for this graph.

Note: A colouring does not have to use colours. Mathematicians often use numbers instead of colours. With this system, each number represents a colour.

Exercise 3

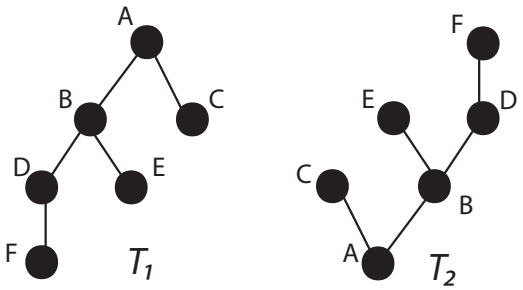
1. Is a 3-colouring also a 4-colouring? Yes, a 3-colouring is also a 4-colouring.
2. What are some applications of colouring? Colouring can be used in tournaments, maps, scheduling, and puzzles like Sudoku.

Theorem. *Every planar graph is 5-colourable.*

Theorem. *Every planar graph has at least one vertex with a degree of 5 or less.*

Theorem. *Every planar graph is 6-colourable.* (You will prove this in the problem set.)

Trees



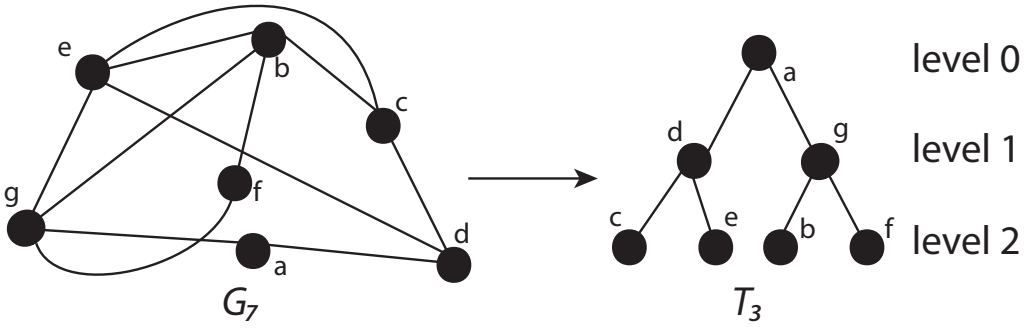
A **tree** is a graph that doesn't contain any cycles (and is connected). For example, T_1 and T_2 are trees. Although edges don't have any direction, think of trees as starting from the top (or bottom depending on how it is drawn) and going down (or up). T_1 starts from the top while T_2 starts from the bottom. We will focus on trees starting from the top like T_1. Trees usually start with one vertex and branch out. The vertex it starts with is called the **root**. In T_1, A is the root. Notice that any vertex can be the root depending on how it is drawn. The vertices at the bottom of the tree that only have one edge incident to them are called **leaves**. A group of trees is called a **forest**.

Breadth-First Search Trees (BFST)

A **Breadth-First Search Tree** or **BFST** is a way to make a tree from any graph. This allows us to easily find vertices and shortest paths.

Important: A BFST can not replace a graph since there are edges missing.

T_3 is an example of a BFST created from G_7.



In BFST's, each row of vertices is on a different **level**. The root of the tree is on level 0. Mathematically, we write $level(a) = 0$.
First we will look at the full BFST algorithm. We will look at a short-cut after.

BFST Algorithm:

1. Draw the first vertex and put it in the **queue**. (You will usually be told which vertex to start with).

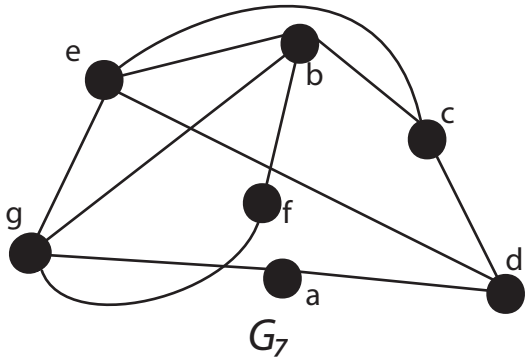
   - The **queue** is a line-up for the vertices. Always add vertices to the back of the queue.

2. Draw the vertices that are connected to it. To make it easier to compare answers, we will write the vertices in order. Put each vertex in the queue as you draw it. Remove the vertex you started with from the queue.

3. Draw the vertices that are connected to the next vertex in the queue. **Do not** draw or connect to any vertices that are already drawn. Remember to add every vertex to the queue as you draw it.

4. Continue doing this until there are no vertices left in the queue.

Example of BFST Algorithm



Create a BFST from G_7 starting at a and queuing the vertices in alphabetical order.

1. Draw vertex a and add it to the queue. Queue: a

2. g and d are adjacent to a. Since d comes before g in the alphabet, draw vertex d and add it to the queue. Draw vertex g and add it to the queue. Remove a from the queue. Queue: d g

3. Since d is next in the queue, look at the vertices adjacent to it. d is adjacent to a, e, and c. a is already in the tree, so we can ignore it. c comes before e in the alphabet so draw c, add it to the queue, draw e and add it to the queue. Remove d from the queue. Queue: g c e

4. g is adjacent to e, b, a, and f, but a and e are already in the tree. Draw b and f, add them to the queue, and remove g from the queue. Queue: c e b f

5. c is adjacent to b, e, and d. All these vertices are already in the tree, so we remove c from the queue without adding anything to the tree or the queue. Queue: e b f

6. e is adjacent to c, b, g and d. All these vertices are already in the tree, so we remove e from the queue without adding anything to the tree or the queue. Queue: b f

7. b is adjacent to c, e, f and g. All these vertices are already in the tree, so we remove b from the queue without adding anything to the tree or the queue. Queue: f

8. f is adjacent to g, and b. All these vertices are already in the tree, so we remove f from the queue without adding anything to the tree or the queue. Queue: empty

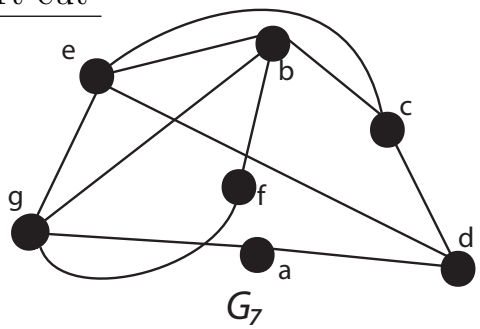
9. The queue is empty so the tree is complete.

BFST Short-cut:

1. Draw the first vertex. (You will usually be told which vertex to start with).
2. Draw the vertices that are connected to it. To make it easier to compare answers, we will write the vertices in order.
3. Draw the vertices that are connected to the first vertex. **Do not** draw or connect to any vertices that are already drawn.
4. Continue doing this until you can not add any more vertices.

The trick with the short-cut is to move from left to right on one level. Once you get to the end of a level, move to the left-most vertex in the next level.

Example of BFST Short-cut



G_7

Create a BFST from G_7 starting at a and queuing the vertices in alphabetical order.

1. Draw vertex a.
2. a is adjacent to g and d, so add them to the tree.
3. Start with d since it comes first in the alphabet. d is adjacent to a, e, and c. a is already in the tree, so add e and c to the tree.
4. Moving right from d, but staying on the same level, we look at vertex g. g is adjacent to e, b, a, and f, but a and e are already in the tree. Draw b and f.
5. Since there are no more vertices on this level, we move to the left-most vertex in the next level: c. c is adjacent to e, b, and d. All these vertices are in the tree so move on to the next vertex.
6. e is adjacent to c, b, g and d. All these vertices are already in the tree, so move on to the next vertex.
7. b is adjacent to c, e, f and g. All these vertices are already in the tree, so move on to the next vertex.
8. f is adjacent to g, and b. All these vertices are already in the tree, so move on to the next vertex.
9. There are no more vertices in this level and there are no more levels, so the tree is complete.

Matchings

A **matching** is a set of edges in a graph where none of the edges have a vertex in common. A matching is shown on a graph by thickened edges. A **maximum matching** is a matching that contains the most possible edges. Matchings are commonly used for bipartite graphs where the goal is to assign or match vertices in one group to vertices in the other group. Remember that a bipartite graph has two groups of vertices that are not adjacent to any other vertices in the same group. Matchings can be used for scheduling problems, job assignments, and many other situations. In these situations vertices in one group represent people, and the vertices in the other group represent spaces to be filled (ex. timeslots or jobs).

Finding a maximum matching in a bipartite graph:

1) Arrange the vertices into two rows (and vertices are not adjacent to other vertices in the same row). Label one group "A" and the other group "B" (it does not matter which is which, but it will be easier if "A" has less unmatched vertices).

2) Make a matching. It doesn't have to be a maximum matching (it can even be just one edge), but it helps to use the largest matching you can easily find.

3) For each unmatched vertex in group A, follow the BFST algorithm with a small change: each level of the tree must alternate between a matched and unmatched edges. This means that you may only add an edge(and the incident vertex) to the tree if it is the opposite of the last edge on the tree. For example, you will always start with an unmatched edge (since the vertex is unmatched), so the next edge(s) must be matched, then unmatched, and so on. Note that you must put all edges that meet the conditions on the tree (which means some levels may have more than one edge).

4) Repeat step 3) for all unmatched vertices in group A, making sure to exclude vertices that are already on one of the trees created in step 3).

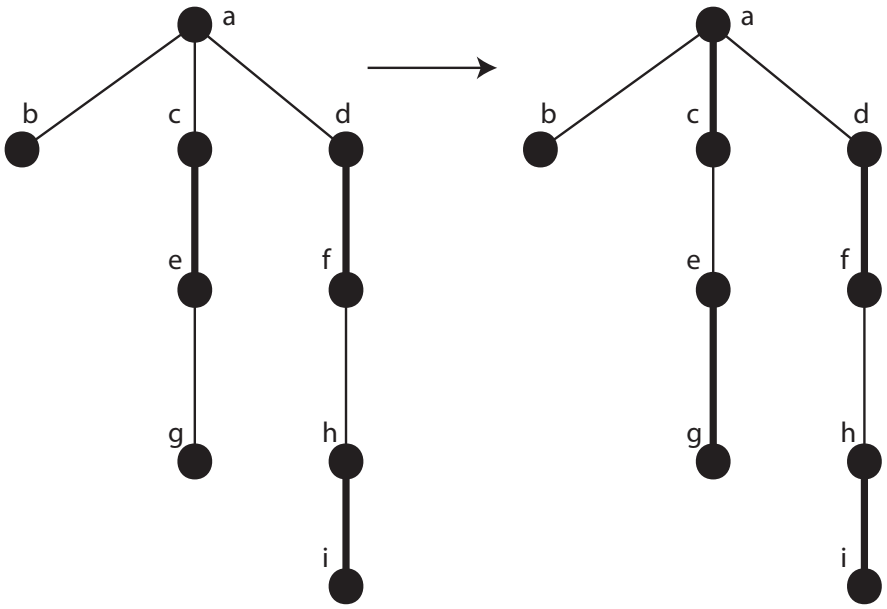
   (a) If all the roots of the trees end with a **matched** edge, stop here. The current matching is a maximum matching.

   (b) If any of the roots end with an **unmatched** edge then proceed to step 5) (ignoring the trees that end in a matched edge).

5) Choose the longest alternating path in each tree that ends with an unmatched edge.For all edges in that path(s), change matched edges to unmatched, and unmatched to matched. Notice that there is now one more matched edge (per path) than before. See the example below.

6) Draw the new matching on a copy of the original graph. The new matching contains the matched edges created in step 5) as well as the matched edges on trees that were not used in step 5), and matched edges that were not used in a tree.

7) Repeat this algorithm using the new matching until you can not proceed (ie. all the trees in step 4) end in a matched edge).



Problem Set

"*" indicates challenge question

1. There are five teams in a tournament.

   (a) Use a graph to prove that there is a way for each team to play every other team once.

      i. How many games does each team play?

      ii. What is the total number of games played?

   (b) The organizers of the tournament would like to consider the different number of games each team can play (ie. how many games each team should play). Assuming each team must play at least one game, the options are for each team to play 1 game, 2 games, 3 games, or 4 games. Assume that each team has to play the same number of games, and that teams can't play each other more than once.

      i. Is it possible to draw a graph for all the different number of games? Explain. Be sure to use specific reasons you learnt in this lesson.

      ii. Draw one graph for each number of games that is possible.

      iii. Can you fix the scenarios from part i) by removing one of the assumptions? You must still assume that each team has to play at least one game.

2. *How can you rearrange vertices to make it easier to see the minimum colouring. (Hint: think about grouping vertices according to what they are connected to)

3. Draw one graph that is 1-colourable, 5-colourable, 10-colourable, 20-colourable, 50-colourable, 100-colourable, **and** 1000-colourable.

4. Draw a graph that has a minimum colouring of

   (a) 1

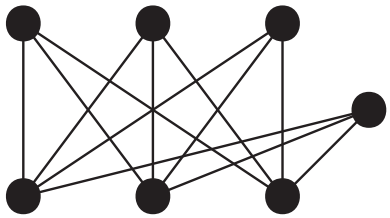
   (b) 3

   (c) 5

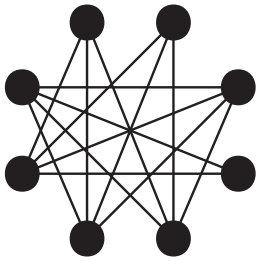
   (d) *10

5. Find the minimum colouring for each of the following graphs.

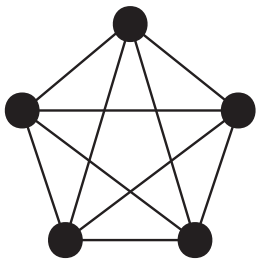
(a)



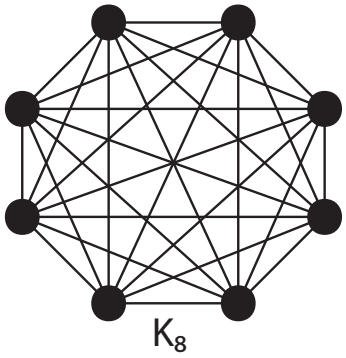
(b) Hint: Use question 2



(c)



(d) Hint: What does K_8 mean?



6. Use the map of Europe on the next page to answer the following questions. When looking at the map, be careful not to mistakes rivers for boarders.

   (a) Find the minimum colouring of the graph. (Hint: use the theorems)

   (b) **Determine a colour for each country. Watch out for countries that that share a very small border. Don't forget about the small countries (you may need to research some countries if they are not clear on the map)

   (c) Ryan wants to go to Sochi, Russia to see where the Olympics took place. He will fly to Portugal and drive to Russia. He doesn't want to waste too much time waiting at the border, so he would like to travel through the least number of countries possible. Use a BFST to find the best route (you may stop the BFST when you reach Russia).

Source: `http://commons.wikimedia.org/wiki/File:Template_europe_map_en.png`

7. Below is a map of where an airline flies (from `http://www.pacificcoastal.com/id/37/Cargojet.html`). Use it to answer the following questions. Notice that this map already has a graph on it. However, you may wish to redraw the map as a graph to make it easier to see.

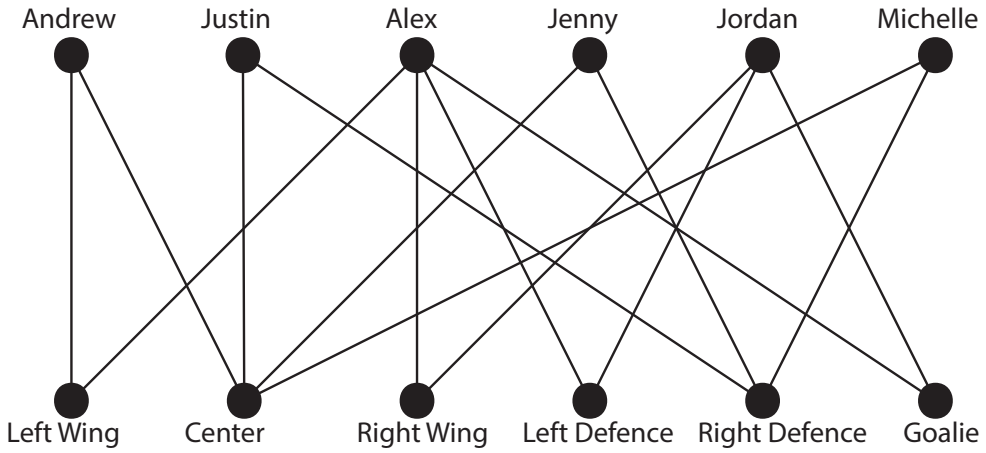
   (a) Find the minimum colouring.

      i. How many "colours" did you use?

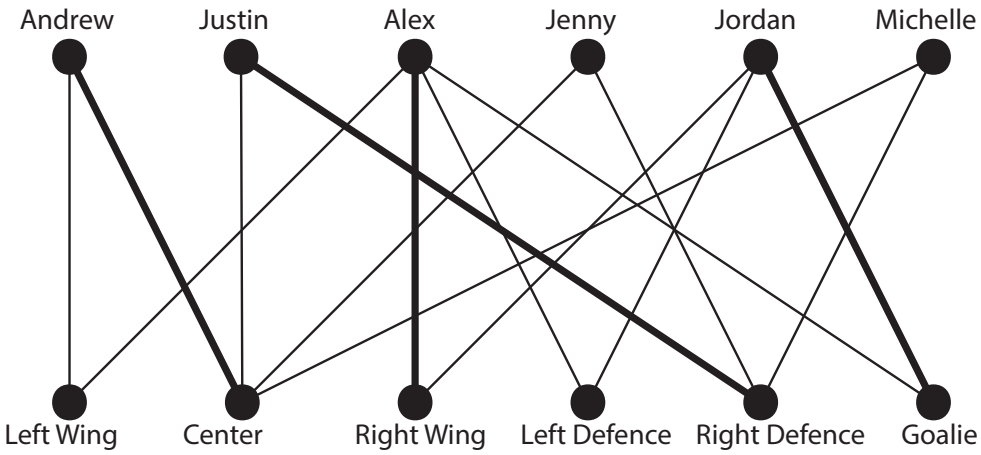
      ii. What does the colouring tell you about the flights? (Hint: think about direct flights)

   (b) Use a BFST to find the least number of flights needed to get from Calgary, Alberta to Halifax, Nova Scotia. How many flights do you need?

8. **The Edmonton Eulers hockey team has 6 spots open (3 forward, 2 defence, and 1 goalie). Below is a graph of the players and the open spots. A player is adjacent to a spot if they can play that position. All the players have an equal skill level and the coach wants to fill as many spots as possible.



(a) Starting with the matching below, use the maximum matching algorithm to find the maximum matching.



(b) What is the maximum number of spots the coach can fill?

(c) Is there a different way the coach can fill the maximum number of spots?

9. ***Prove the six-colour theorem.

10. ** Show that in a tournament with 25 people, at the end of the tournament, if x is the number of people who played an odd number of games, then x is even.

11. ** In the UW Information Game, 50 people are each given a unique piece of information. A person may call another person by one way-radio and transmit their information to the receiver, but it does not go both ways. For example, Mike can call Alice and give her his information (and thus she would have 2 pieces), but she cannot give him hers in the same call; she would have to make another call and then give it to him. However, Alice could then call Steve and give him all of her information (her original info, and the info she got from Mike). Then Steve would have 3 pieces of information. What is the minimum number of phone calls required so that everybody has all 50 pieces of information, and how do you achieve this?

12. *** Show that in a house with 25 rooms, if every room has an odd number of doors then there must be an odd number of doors alongside the outside wall of the house.