# Grade 9/10 Math Circles

## November 8, 2023

## Graph Theory

## Warm-up: Timetabling, BCC

**Story**

Bebras Tech offers the following evening classes:

- Computing (C),
- Geography (G),
- Language (L),
- Math (M), and
- Science (S).

Three beavers would like to sign up for these courses:

- Xavier wants to take C, L, and M;
- Yvette wants to take C, G, and S;
- Zoey wants to take L, M, and S.

Bebras Tech wants to squeeze these courses into as few evenings as possible such that:

- each course is offered on exactly one evening, and
- beavers can take at most one course per evening.

**Timetabling**

This problem comes from the 2018 Beaver Computing Contest! What is the least number of evenings needed for Bebras Tech to schedule courses?

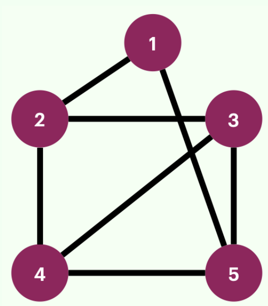
## Definitions

Let's review some of our definitions from last week!

**Definition 1.** A **graph** is a set of vertices paired with a set of unordered pairs of distinct vertices, called edges.

**Definition 2.** A vertex is a **neighbour** of another vertex if there is an edge between them.

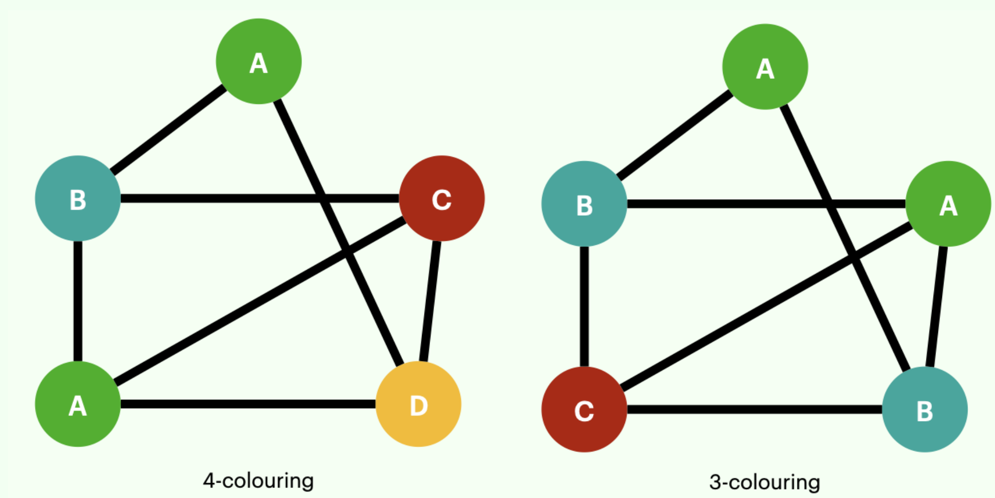
**Example: Graph and Neighbours**




In the graph, vertex 1 and vertex 2 are neighbours.
Vertex 1 and vertex 3 are not.
What are the neighbours of vertex 4?

**Definition 3.** A **k-colouring** is a way to assign one of $k$ colours to each vertex such that vertices that are neighbours have different colours.

**Example: $k$-Colouring**

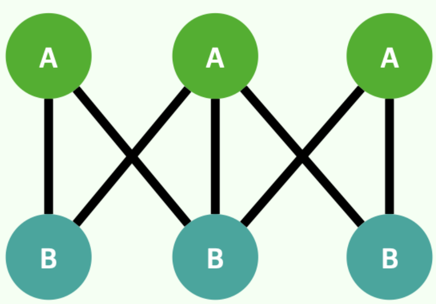


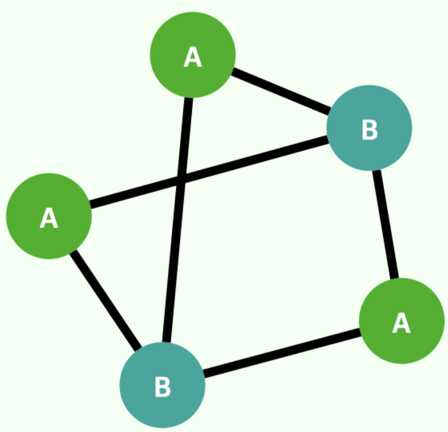
**Exercise 1**

Draw a graph then swap pages with someone else and give the graph a colouring! (Use letters or words if you don't have different coloured writing tools.)

**Definition 4.** A graph is **bipartite** if it is 2-colourable.

**Example: Bipartite**



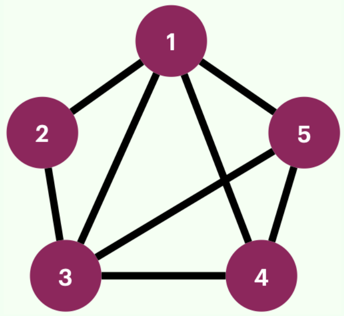




**Definition 5.** A **walk** is a sequence of vertices and edges that lead from one vertex to another.

**Definition 6.** A **path** is a walk where we do not return to the same vertex twice (except possibly the last vertex).

**Definition 7.** A **cycle** is a path which starts and ends at the same vertex.

**Example: Walks, Paths and Cycles**
For:




$1 \to 2 \to 3 \to 1 \to 4 \to 5$ is a walk 1 to 5.

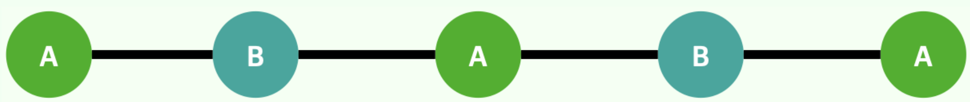
$1 \to 2 \to 3 \to 5$ is a path from 1 to 5.

$1 \to 3 \to 1 \to 5$ is not a path.

$1 \to 3 \to 5 \to 1$ is a cycle.

**Theorem 8.** *A graph which is just a path is always 2-colourable.*

**Example: 2-colouring**




**Exercise 2**

When is a graph 1-colourable? When is a graph 2-colourable?

Optimal graph colouring is "NP-complete" - there is no quick way to find a colouring with a minimum number of colours.

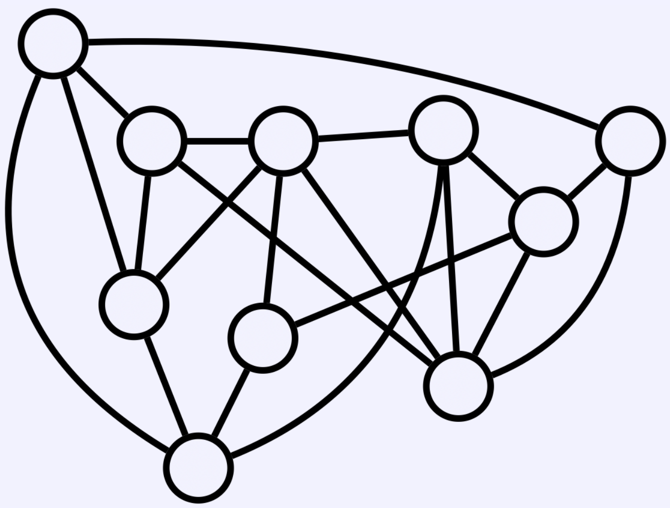
But! We can find a colouring.

## Greedy Colouring Algorithm

1. Order the potential colours
2. Pick a vertex that hasn't been given a colour
3. Colour that vertex with the first valid colour
4. Repeat steps 2-3 until all vertices have a colour

**Exercise 3**

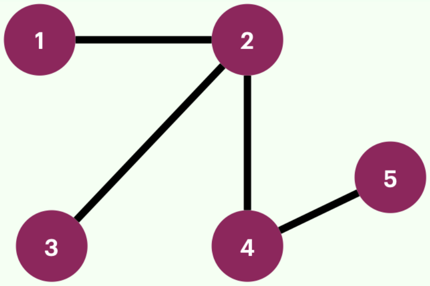
Colour the following graph using the greedy colouring algorithm.



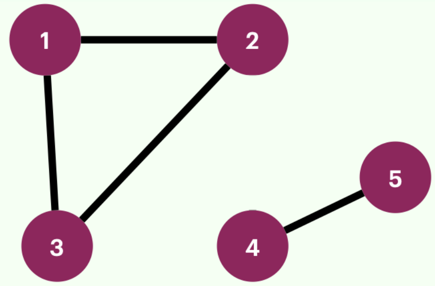
**Definition 9.** A **tree** is a graph that is **connected** (all vertices have paths to each other) and has no cycles.

**Example: Tree**

Tree:

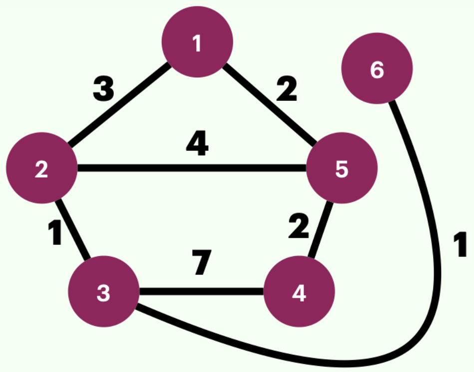



Not a Tree:



**Definition 10.** A **weighted graph** is a graph where each of the edges has a weight or cost assigned.

**Example: Weighted Graph**




How much does it cost to travel from vertex 1 to vertex 6?

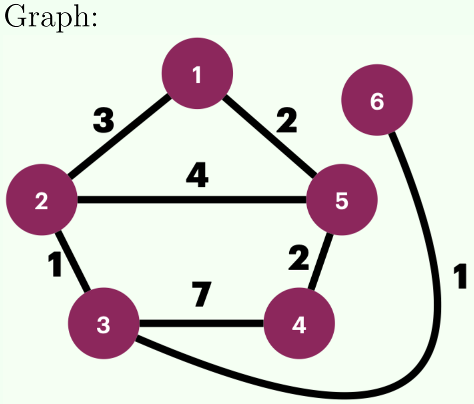
$1 \to 2 \to 3 \to 6$

$3 + 1 + 1 = 5$

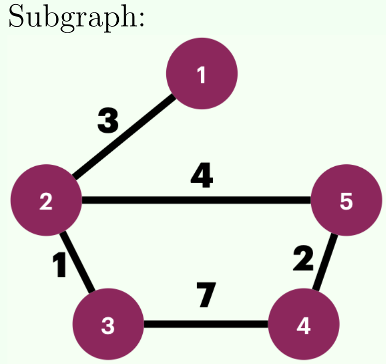
**Definition 11.** A **subgraph** is a graph whose edges and vertices are all part of a possibly larger graph.

**Example: Subgraph**

Graph:




Subgraph:



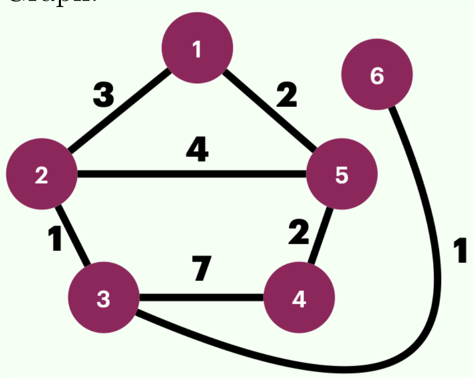

**Definition 12.** A **spanning tree** is a subgraph with the following properties:

- it is a tree
- all vertices from the original graph must be included

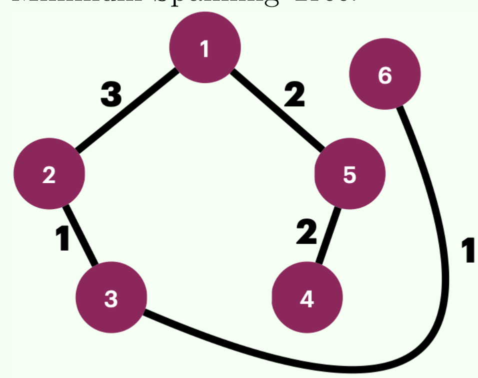
**Definition 13.** A **minimum spanning tree (MST)** is a spanning tree of lowest cost.

**Example: MST**

Graph:



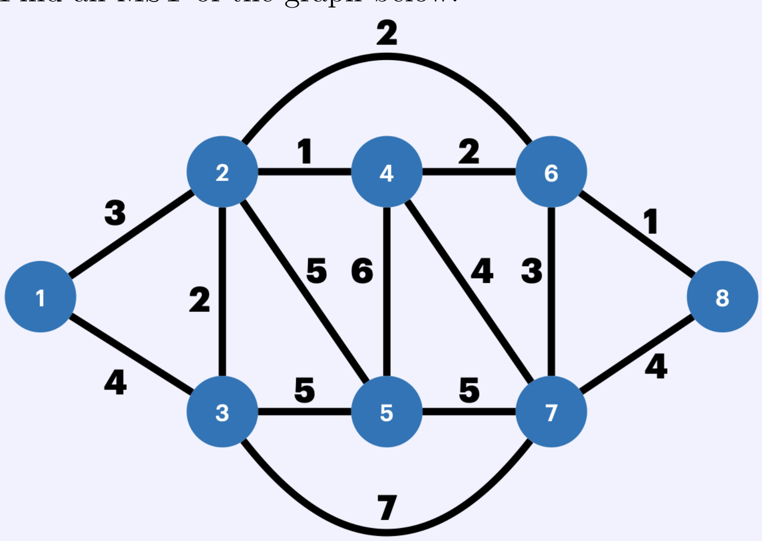

Minimum Spanning Tree:




MST with cost $3 + 1 + 2 + 2 + 1 = 9$

**Exercise 4**

Find an MST of the graph below:



## Prim's Algorithm

Prim's Algorithm is a way of finding an MST of a graph.

1. Pick a vertex in your graph - it is the start of our spanning tree
2. Consider all edges that are connected to exactly one vertex in the current tree
3. Pick the edge with the smallest weight and add it, along with the new vertex, to the tree
4. Repeat steps 2-3 until all vertices are in the tree

**Exercise 5**

Use Prim's Algorithm to find a different MST of the graph below: