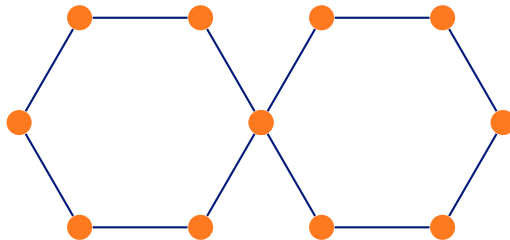


Graphs, Groups, and Isomorphisms

ian.mcloughlin@atu.ie

February 13, 2023



Graphs provide a way of working with and visualizing connections between objects. They help us understand symmetry, relationships, and algorithms.

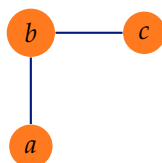
References

We will use Norman Biggs' *Discrete Mathematics*,¹ Biggs' *Algebraic Graph Theory*,² and Michael Sipser's *Introduction to the Theory of Computation*.³ Another good resource is the Open Logic Text.⁴ On the practical side, I recommend The Python Tutorial,⁵ The Python Software Foundation's official tutorial for Python.

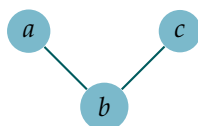
Graphs

A graph G is a 2-tuple (N, E) where N is a finite set and E is a set of 2-subsets of N . A 2-subset is a subset with two elements. The elements of N are called nodes or vertices. The elements of E are called edges.

Suppose $N = \{a, b, c\}$ and $E = \{\{a, b\}, \{b, c\}\}$. We can represent this graph using a picture, as below.



Note that the graph says nothing about colours or locations of nodes. That is just part of the picture. We can draw the same graph differently if we wish.



We have to be careful not to mistake the picture for the graph.

IN PYTHON, one way to represent a graph is to use sets. Unfortunately, sets in Python cannot contain other sets. Sets in Python are

¹ Biggs, Norman L. *Discrete Mathematics*. revised Edition. Oxford Science Publ., 1989.

² Norman Biggs. *Algebraic Graph Theory*. 2nd ed. Cambridge Mathematical Library. Cambridge University Press, 1974. DOI: 10.1017/CB09780511608704.

³ Sipser, Michael. *Introduction to the Theory of Computation*. Third. Boston, MA: Course Technology, 2013. ISBN: 113318779X.

⁴ Open Logic Project. *Open Logic Project Builds*. Dec. 21, 2022. URL: <https://builds.openlogicproject.org/> (visited on 01/23/2023).

⁵ The Python Tutorial — Python 3.11.1 documentation. Jan. 22, 2023. URL: <https://docs.python.org/3/tutorial/> (visited on 01/22/2023).

mutable and *unhashable*. There is a similar, immutable type called `frozenset` that we can use.

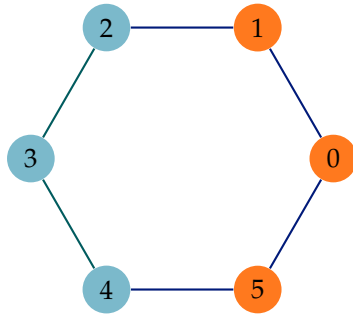
```
N = {'a', 'b', 'c'}
# TypeError: unhashable type: 'set':
# E = {{ 'a', 'b'}, { 'b', 'c'}}
E = {frozenset({'a', 'b'}), frozenset({'b', 'c'})}
```

In any case, we usually use other, more efficient data structures to represent sets, such as adjacency matrices. More on those later.

Subgraphs

A subgraph is a graph fully contained in another graph. We say $G' = (N', E')$ is a subgraph of $G = (N, E)$ when N' is a subset of N and⁶ E' only contains edges containing elements of N' .

Consider the following picture of the *cycle graph* C_6 . It contains as subgraphs several copies of the graph above. Some of these copies overlap. One copy is depicted with teal nodes. Remember the colours do not matter to the graph. The graph does not care how it is drawn so long as the nodes and edges are correct.



⁶ This last condition just ensures that E' is a set of 2-subsets of N' .

Adjacency Matrices

A common way to represent a graph is with an adjacency matrix. Two nodes a and b of a graph $G = (N, E)$ are *adjacent* if $\{a, b\}$ is an edge of the graph — that is, it is in E . To form an adjacency matrix for a graph, we must create an ordering of N . Remember N is a set, so the elements do not come in any order.

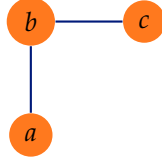
We can fix an ordering of N by creating a tuple of length $|N|$ ⁷ where every element of N appears exactly once. Then the tuple defines an order on N . With that order we can create the adjacency matrix of the graph with respect to it.

The *adjacency matrix* of the graph $G = (N, E)$ according to the ordering $(n_1, n_2, \dots, n_{|N|})$ of N is the matrix A with entry a_{ij} in row i and column j given by the following formula where i and j range over 1 to $|N|$.

⁷ The notation $|S|$ means the number of elements in the set S .

$$a_{ij} = \begin{cases} 1 & \text{if } \{n_i, n_j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

For example, consider the graph again where $N = \{a, b, c\}$ and $E = \{\{a, b\}, \{b, c\}\}$.



We can fix N in any order we like — let us pick (a, b, c) . Then the adjacency matrix is as follows.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

We can more-or-less re-create a graph from an adjacency matrix. An adjacency matrix does not tell us the elements the node set N contains. However, it does tell us exactly how many elements it contains and how they are connected with edges. So, we can draw the graph from the adjacency matrix. You might try drawing the graph represented by the following adjacency matrix.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

It is worth considering what matrices are possible adjacency matrices of graphs. We have specified that each entry must be either 0 or 1. The matrix must also be square.

Note that it is impossible for a graph as we have defined it to have looped edge. The edges must be 2-subsets and sets cannot contain the same element more than once. So when a is a node, $\{a, a\}$ cannot be an edge because it only contains one element, despite how we have written it. This leads to the fact that an adjacency matrix has zeroes down the main diagonal.

It is also symmetric along that diagonal — a_{ij} is equal to a_{ji} . That is because any edge $\{a, b\}$ is the same edge as $\{b, a\}$.