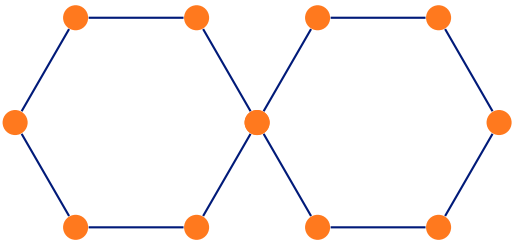
Graphs, Groups, and Isomorphisms

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.

[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/CBO9780511608704.

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

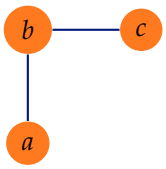
[4] Open Logic Project. *Open Logic Project Builds*. Dec. 21, 2022. URL: https://builds.openlogicproject.org/ (visited on 01/23/2023).

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

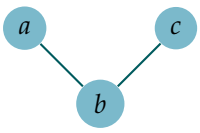
Graphs

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

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'})}
```

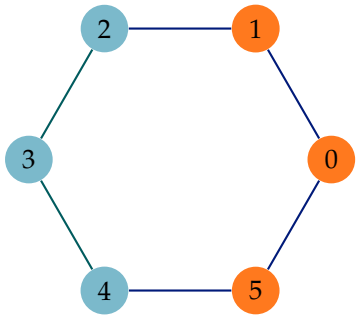
In any case, we usually use other, more efficient data structures to represent sets, such as adjacency matrices. More on those later.

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 and[6] E' only contains edges containing elements of N'.

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



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 its node set N. Remember N is a set, so the elements do not come in any order.

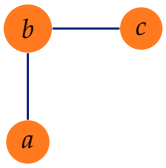
We can fix an ordering of N by creating a tuple of length $|N|$[7] where every element of N appears exactly once. Then the tuple defines an order on N. With that order we can create the adjacency matrix of the graph with respect to it.

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

The *adjacency matrix* of the graph $G = (N, E)$ according to the ordering $(n_1, n_2, \ldots, n_{|N|})$ 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 $|N|$.

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

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. What other properties must it have? Can you list a sufficient list of such properties?

Note that it is impossible for a graph as we have defined it to have looped edges. A looped edge connects a node to itself. 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.

An adjacency matrix 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 $\{b, a\}$.

Digraphs and Multigraphs

We can consider defining graphs differently if we want loops and directed edges. Both concepts can be covered by changing the definition of the edge set to contain pairs[8] Then we can have a looped edge

[8] Pairs are 2-tuples where we use parentheses notation. For example, (a, b) is a pair where a is the first element and b the second.

(a, a) connecting node a to itself. We also then distinguish the edge (a, b) from the edge (b, a), as two edges connecting the same nodes but with opposite direction[9].

When graphs are redefined in this way we usually call them *digraphs*[10]. We redefine the adjacency as you would expect. The matrix entry a_{ij} is 1 if there is an edge from node i to node j. In that case, looped edges place a 1 in the main diagonal and the matrix might not be symmetric[11].

Digraphs, similar to our original definition of graphs, do not allow for more than one edge between the same node(s). When we want repeated edges, we define the notion of a *multiset*, which is a set that can contain multiple copies of the same object. In that case we might call the graphs *multigraphs* or even *multidigraphs*. While digraphs and multigraphs have many applications, we will focus solely on graphs under our original definition as they have the broadest applications.

[9] Typically we say the direction of the edge (a, b) is *from a to b*.

[10] We will always use the term digraph from now on, and reserve the word graph for our original definition.

[11] Symmetric matrices are mirrored along the main diagonal. The matrix transpose is equal to the matrix.

Isomorphisms

Permutations

Automorphisms