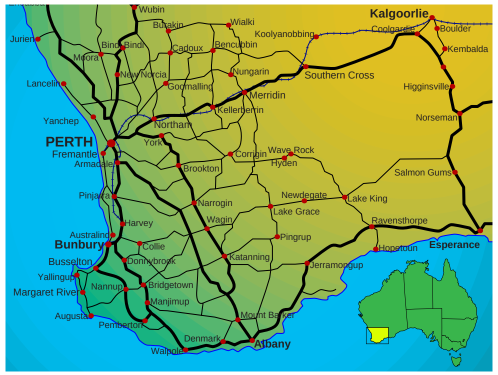
MATH1510 - Discrete Mathematics
Graphs

University of Newcastle

UoN

Road networks



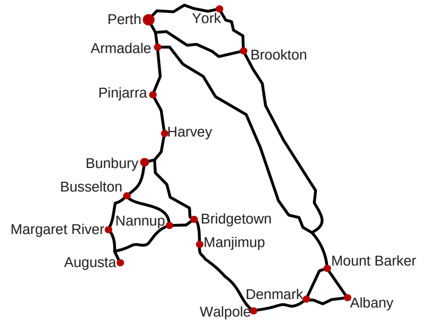
Shortest Path Problem

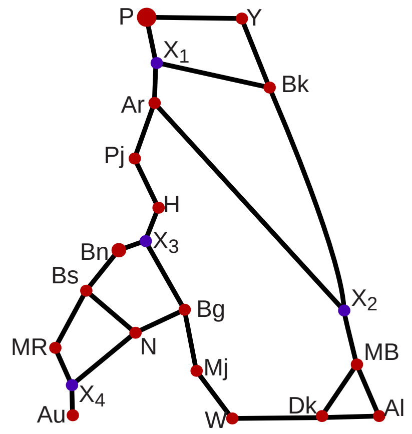
What is the shortest route from Bunbury to Albany?

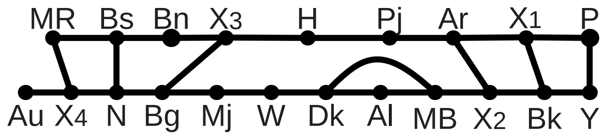
Traveling Salesman Problem

Is it possible to start in Perth, visit every city exactly once, and return to Perth?

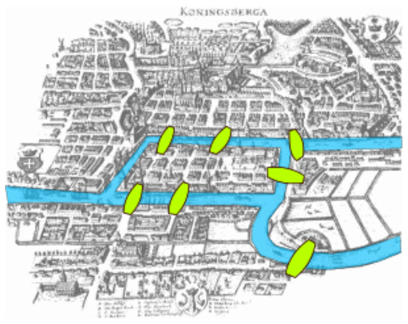
Abstract representation







The seven bridges of Königsberg



- The city of Königsberg in Prussia (now Kaliningrad, Russia) was set on both sides of the Pregel River.
- There were two large islands connected to each other and the mainland by seven bridges.

Problem

Find a walk through the city that crosses each bridge once and only once.

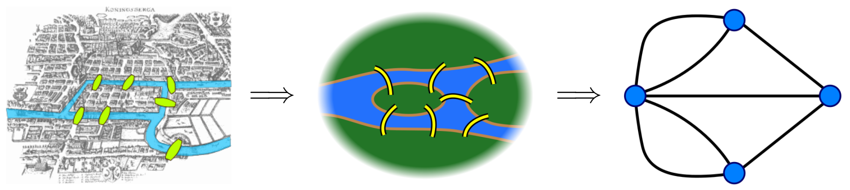
The first theorem of graph theory (Leonhard Euler, 1735)

There is no walk through the city of Königsberg that crosses each bridge once and only once.

Abstract representation

The problem corresponds to a diagram where

- Each body of land is represented by a point (call it a vertex), and
- Each bridge is represented by an arc (call it an edge).

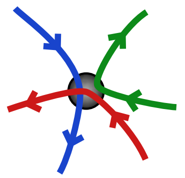


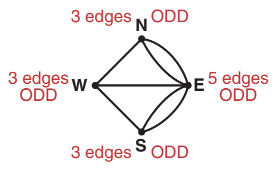
Problem reformulation

Draw the diagram in a single stroke and without retracing.

Euler's argument

At each vertex, except possibly the starting and the end vertex, the walk enters the vertex the same number of times as it leaves. Since every edge is used once and only once there must be an even number of edges at every vertex except at most two.





In the Königsberg diagram there are four vertices with an odd number of edges, hence there cannot be a walk with the required property.

Definitions

- A graph is a pair $G = (V, E)$ of two sets V and E.
- The elements of V are called vertices.
- The elements of E are called edges.
- Every edge $e \in E$ is associated with an unordered pair of two vertices u and v which are called the endpoints of e.

Some equivalent terms

Graphs are also called networks; vertices are also called nodes; edges are also called arcs.

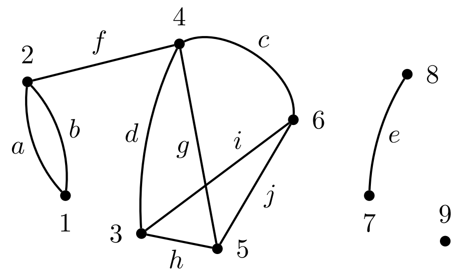
Additional structure

In applications it can be necessary to keep some more data: distances or travel times, directions, etc. Building on the basic definition of a *graph*, this is captured by concepts such as weighted graphs and directed graphs, which we will encouter later in the course.

Diagrams of graphs

To draw a graph we use

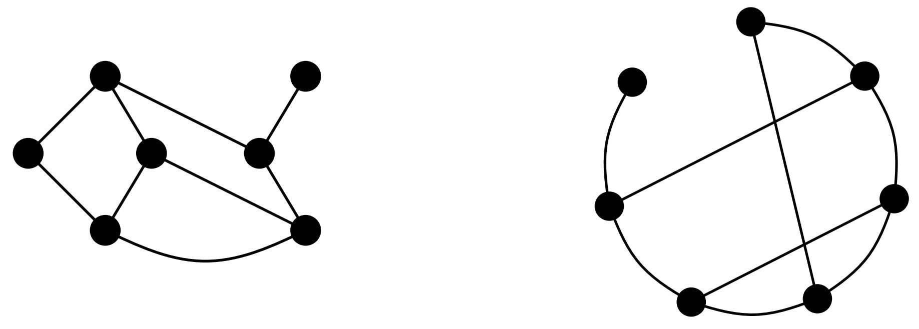
- symbols such as dots/circles for vertices,
- and lines for edges.



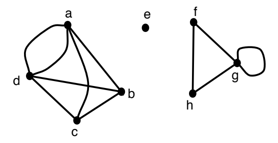
- vertex set $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- edge set $E = \{a, b, c, d, e, f, g, h, i, j\}$
- Edges a and b are associated with vertices 1 and 2; edge g is associated with vertices 4 and 5.

Diagrams of graphs

One graph can be represented by many different diagrams.

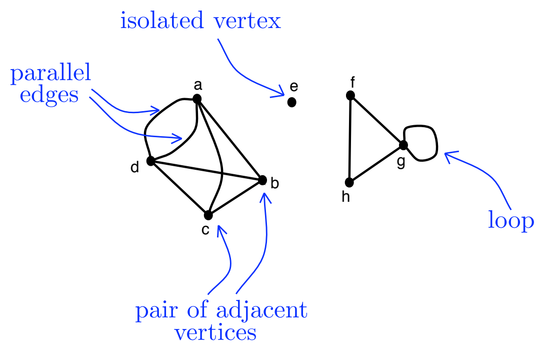


Some Graph Terminology



- An edge connects its endpoints.
- An edge is incident with its endpoints.
- A vertex with no incident edges is isolated.
- An edge with both endpoints the same is called a loop.
- The degree of a vertex is the number of edges incident with it, with loops counted twice.
- Two edges may connect the same pair of endpoints, in which case they are said to be parallel.
- Two vertices are adjacent if they are connected by an edge.
- Two edges are adjacent if they share an endpoint.

Some graph concepts illustrated



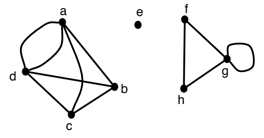
Notation for Vertices and Edges

- We often write $V(G)$ for vertices and $E(G)$ for edges when we want to mention the graph G explicitly.
- There are various different notations used for the unordered pairs of vertices $u, v \in V$ associated with an edge $e \in E$, including
  - (u, v) or (v, u) used by our textbook, even though we mean an **unordered pair**, not an ordered pair
  - uv or vu
  - $\{u, v\}$ or $\{v, u\}$
- In the special case where there is only one edge, e, between a pair of vertices u and v, we write any of

$$e = (u, v), \quad e = (v, u), \quad e = uv, \quad e = vu, \quad e = \{u, v\}, \quad e = \{v, u\}.$$

Ways of representing graphs:

The graph G :

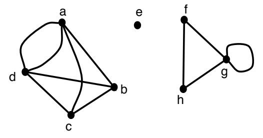


has **vertex adjacency listing**:

Vertex	Adjacent to:
a	b, c, d, d
b	a, c, d
c	a, b, d
d	a, a, b, c
e	
f	g, h
g	f, g, h
h	f, g

Ways of representing graphs:

The graph



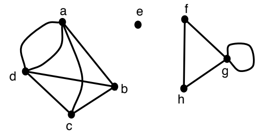
has **adjacency matrix:**

$$\begin{array}{c|cccccccc} & a & b & c & d & e & f & g & h \\ \hline a & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ b & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ c & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ d & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ g & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ h & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

The $(i, j)^{\text{th}}$ entry is the number of edges between vertices i and j.

Ways of representing graphs:

The graph



has **incidence matrix:**

$$\begin{array}{c|ccccccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} \\ \hline a & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

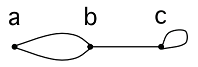
The $(i, j)^{\text{th}}$ entry is '1' if the i^{th} vertex is incident with the j^{th} edge, otherwise '0'.

Three of the following represent the same graph. One is different: which one?

A Vertex adjacency listing

Vertex	Adjacent to
a	b, b
b	a, a, c
c	b, c

B



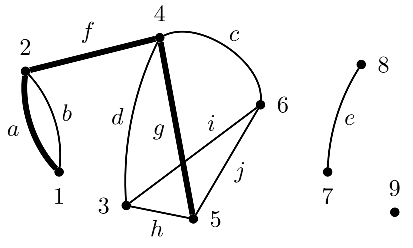
C Incidence matrix $M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$

D Adjacency matrix $A = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

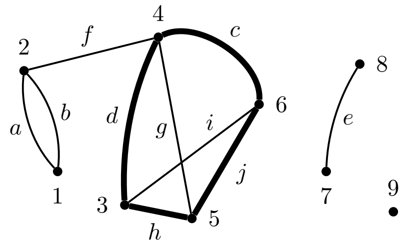
Paths and cycles

Many problems (such as the Königsberg bridge problem) require us to move through the graph from vertex to vertex in some way. If we start at vertex v_1, move along an edge to vertex v_2, then along another edge to v_3, etc. and arrive at vertex v_n, we have described a path from v_1 to v_n. If we end up where we started we get a cycle or circuit.

Example

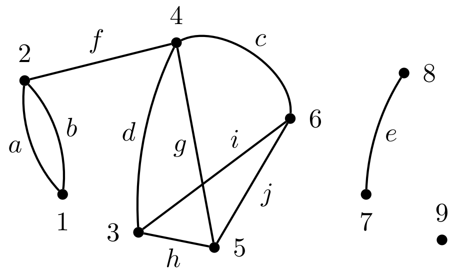


$(1, a, 2, f, 4, g, 5)$ is a path from vertex 1 to vertex 5.



$(3, d, 4, c, 6, j, 5, h, 3)$ is a cycle.

How many paths visiting no vertex more than once are there from 1 to 5?



- A 4
- B 7
- C 8
- D 10

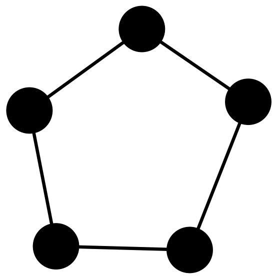
Length, distance and diameter

Paths can be used to define *distances* in a graph

length	of the path is the number of edges in it.
distance	between two vertices is the length of the shortest path between them.
diameter	of the graph is the maximum distance between any two vertices.

Note: The definition of distance and diameter assume that there is *some path* between the vertices in question. We will examine connectivity later.

What is the diameter of this graph?



A 1

B 2

C 3

D 5

Similarity graphs

Another common type of graph is where vertices represent entities that are connected when they are in some way similar. For instance, this concept is applied

- in molecular biology, where the vertices are DNA sequences, edges represent similarity, and the resulting graphs can be used to study relationships between species, or
- in handwriting recognition, where the vertices are strings of letters, and distances in a similarity graph can be used to match an input string scanned by a computer to the entries of some dictionary.

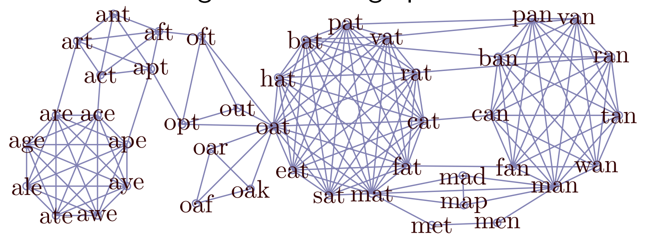
Example: word puzzle

A common word puzzle is to transform one word into another, one letter at a time, via intermediate words. For example, dog to cat like this:

$$\text{dog} \to \text{cog} \to \text{cot} \to \text{cat} \; .$$

We use a graph with a vertex for each word and an edge where two words differ in one letter. A solution to the puzzle is a *path* between the two vertices.

The diagram below is a fragment of this graph



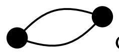
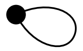
So for APE $\to$ MAN, we can see that there are many solutions. However, we observe that there is a best solution of 5 steps:

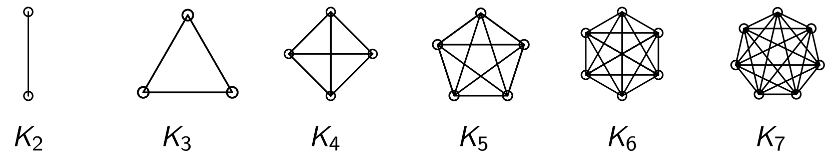
$$\text{ape} \to \text{apt} \to \text{opt} \to \text{oat} \to \text{mat} \to \text{man}$$

and so we can say dist(APE,MAN) $= 5$.

Simple graphs

The "word puzzle" graph shows certain features.

- There are no *parallel edges*:  or *loops*: 
- We define a simple graph to be one without loops or parallel edges.
- Simple graphs are nice — to identify an edge in a simple graph, it is enough to say something like "the edge from vertex x to vertex y" or "the edge (x, y)"
- The "word puzzle" graph has many *subgraphs* where a number of vertices are all adjacent to each other. That is, it has many subgraphs that are copies of a complete (simple) graph K_n:



Trees

Graphs are used to represent data structures, with vertices representing data objects and edges representing links between them.

For instance, a simple file-system on a storage device has a single "root folder" and subfolders and files can be seen to be descendent from it, forming a tree.

When data is stored in a tree-based structure, there are algorithms for manipulating and maintaining data, which can be visualised as algorithms that manipulate the tree.

Binary search trees

Binary search trees are constructed by storing data in a tree where data objects are inserted one at a time into the tree in a way the represents an ordering of the data.

In a BST, the first object inserted is at the root.

Thereafter, objects are inserted by either descending left or right depending on whether the object should go before or after the "current" vertex.

Once an appropriate space is found in the tree, the data object is added there.

The depth of a BST is the maximum distance between any vertex and the root.

Data stored in a BST is easily searched. We can write an algorithm for it.

```
function FindInTree(tree $T$, data $X$) {
  Set *here* to the root of $X$
  Repeat {
    **If** $X =$ *here*, **then stop**:  data $X$ is found
    **If** $X <$ *here*, **then**
      **if** leftchild of *here* exists, **then**
        set *here* to leftchild of *here*
      **else**
        **stop**, $X$ not found
    **If** $X >$ *here*, **then**
      **if** rightchild of *here* exists, **then**
        set *here* to rightchild of *here*
      **else**
        **stop**, $X$ not found
  }
}
```

Does it scale?

For small trees, finding objects using this algorithm is quick.

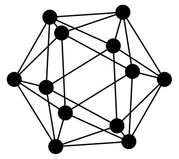
However, for larger BSTs, experiments show that the efficiency of the searching and insertion algorithms depend on the tree's shape.

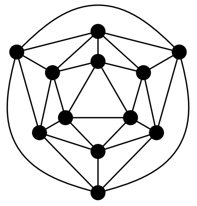
However, once the tree has hundreds, thousands or millions of vertices we have to ask whether the time taken to access data in the tree is still relatively quick.

If we have a poor strategy for maintaining the tree, then the overall storage strategy will not scale.

Skeletons of polyhedra

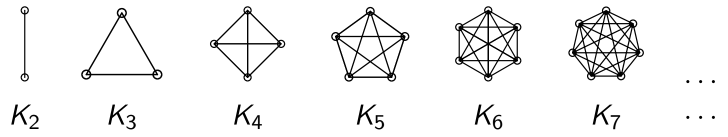
The vertices and edges of a polyhedron form a graph. One useful property is that with a convex polyhedron such as the regular icosahedron, the graph can be laid flat in the plane.





Complete graph

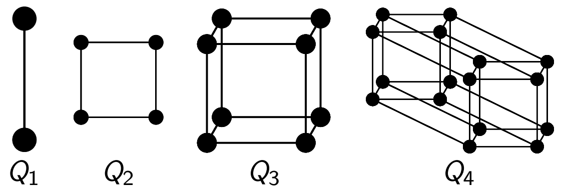
The Complete simple graph on n vertices, written K_n, is a simple graph with n vertices and an edge between every possible pair of vertices. We often label the vertices $1, 2, \ldots, n$ and then the edges are $(1, 2)$, $(1, 3)$, …, $(1, n)$, $(2, 3)$, …, $(2, n)$, $(3, 4)$, …, $(n-1, n)$.



n-cube

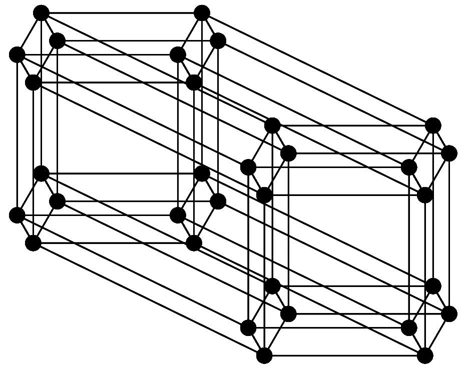
The n-cube is a model for efficient connection of processors in certain parallel computing architectures, as used in the Connection machine.

vertices	processors
edges	direct connection



n-cubes can be characterised via a geometric algorithm or symbolically via binary strings.

What is the diameter of the 5-cube?



A 32

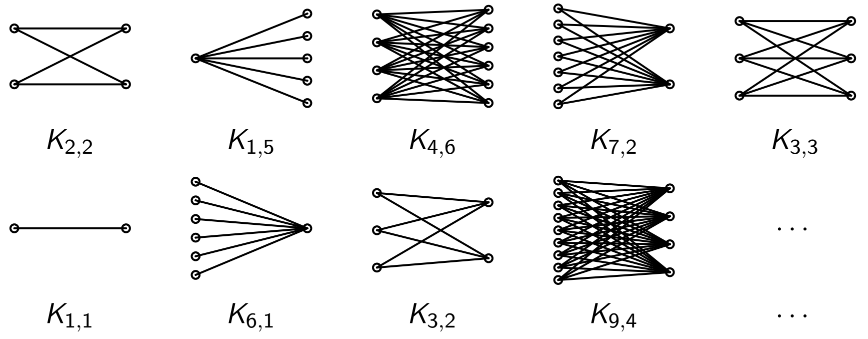
C 5

B 10

D 25

Complete bipartite graph

The Complete bipartite graph on m and n vertices, denoted $K_{m,n}$ is a simple graph with $m + n$ vertices and an edge between every pair of vertices (i, j) where i is in the first m vertices and j is in the last n vertices.



Summary

Graphs. Used to model a system involving objects and connections between them.

Paths. Arise in many applications, and are used to define other concepts: distance, diameter.

Simple graphs. Arise in some systems and are simpler to work with. For instance, edges can be identified with sets of two vertices.

Trees. Special type of graphs. They are important for organising large amounts of data.

Standard graphs. Examples such as polyhedra, K_n, Q_n, $K_{m,n}$.