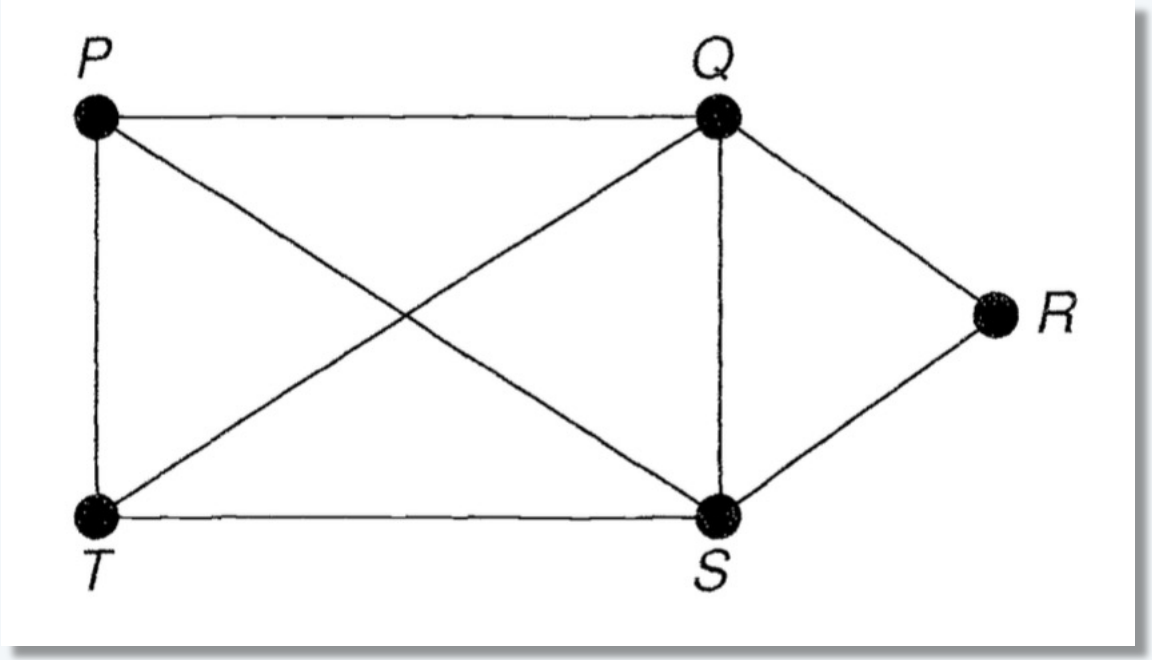
DM587/DM579 –
Scientific Programming
Linear Algebra with Applications

The Graph- (and the Subgraph)-Isomorphism Problem, the Ullmann Algorithm, and Applications in Chemistry

Slides by Daniel Merkle

daniel@imada.sdu.dk

What is a graph?

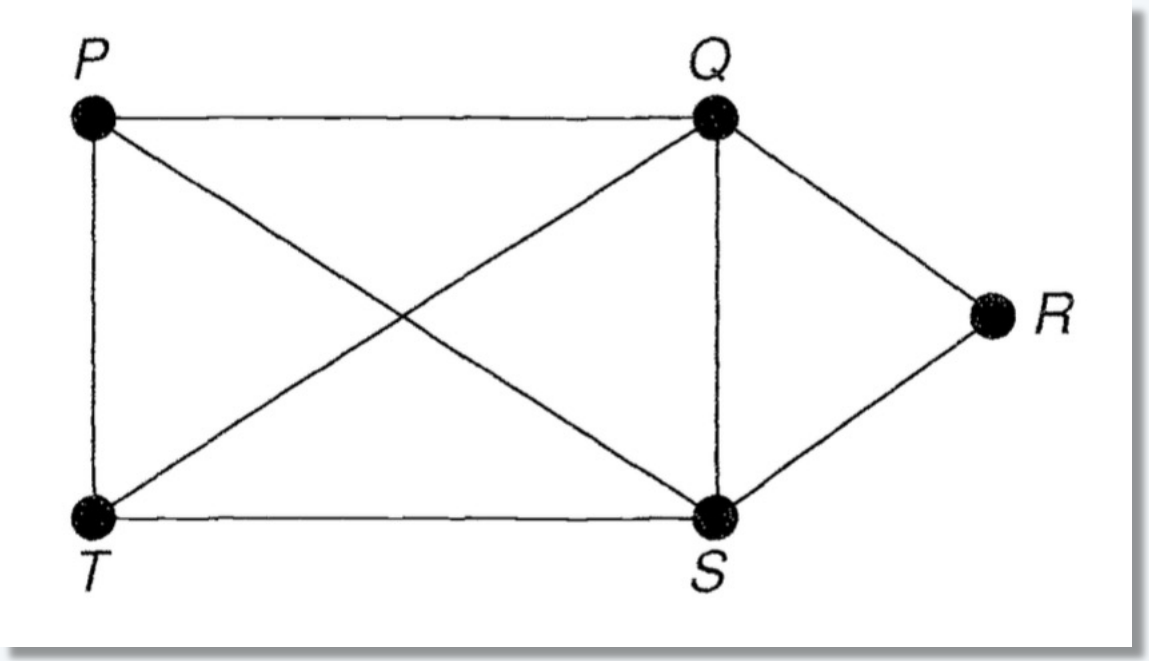


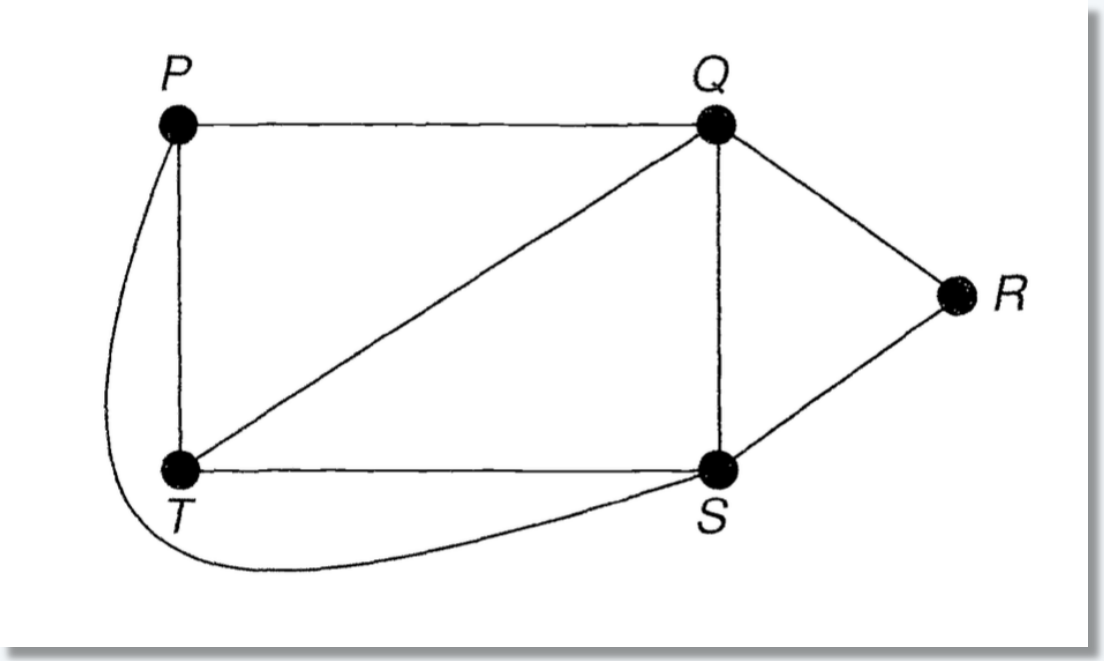
Vertices: P, Q, R, S, T

Edges: all the lines

Degree of a vertex: number of edges with that vertex as an end-point

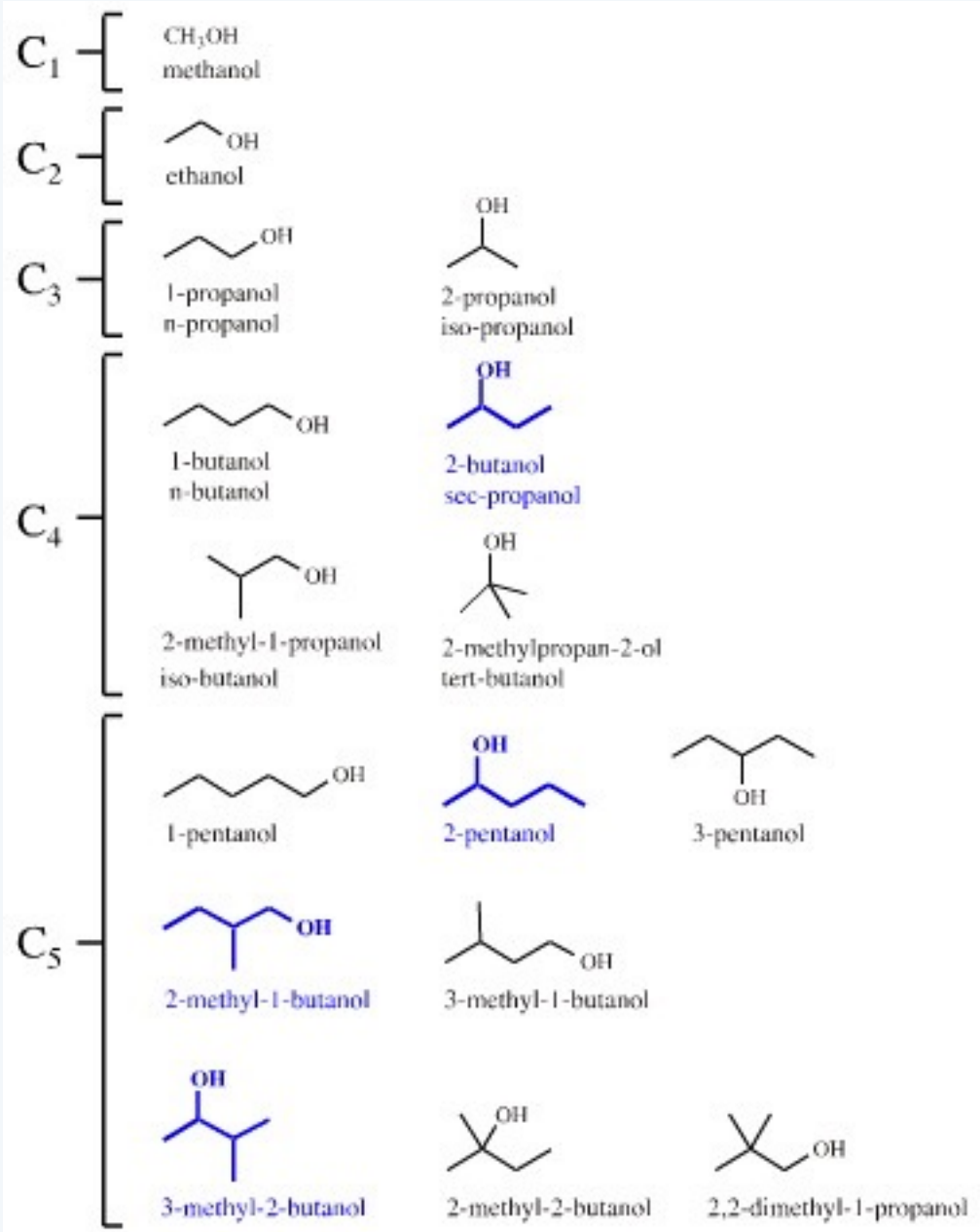
Two different graphs? No!





In the right graph we have removed the 'crossing' of the lines PS and QT by drawing the line PS outside the rectangle PQST.

The first scientific article using the term graph



8 cm. from the primary. *Reverse the wires in the secondary circuit, reverse the wires in the primary circuit, how you please, the mercury always moves towards the point of the capillary.*

8. Shouting or singing (excepting the above-mentioned note) produces no visible effect under the conditions mentioned in Experiments 5, 6, and 7.

9. If the secondary coil be now moved close up, so as to cover as completely as possible the primary, talking to the telephone with the ordinary voice, *i.e.* with moderate strength and at any pitch, produces a definite movement of the mercury column for each word, some sounds of course giving more movement than others, *but the movement is always towards the end of the capillary.* Singing the note mentioned in Experiments 5, 6, and 7 loudly, produces a movement too large to be measured with the electrometer.

Reversing the poles of the magnet in the telephone does not alter the results of Experiments 5, 6, 7, and 9.

On mentioning the above results to Dr. Burdon Sanderson, he suggested that the apparently anomalous behaviour of the electrometer might be accounted for, by supposing that the mercury moved *quicker* when a current passed towards the point of the capillary than when it flowed in the opposite direction ; so that if a succession of rapidly alternating currents be passed through the instrument, the mercury will always move towards the point of the capillary, the movement away from the point being masked by the sluggishness of the instrument in that direction. That this explanation is the correct one is proved by the following experiment :—The current from two Grove's cells is sent through a metal reed vibrating 100 times a second, the contact being made and broken at each vibration, the primary wire of a Du Bois Reymond's induction-coil is also included in the circuit ; on connecting the electrometer with the secondary coil placed at an appropriate distance the mercury always moves to the point of the tube whatever be the direction of the current.

F. J. M. PAGE

Physiological Laboratory, University College, London, February 2

NOTE.—On February 4 Prof. Graham Bell kindly placed at my disposal a telephone much more powerful than any of those I had previously used. On speaking to this instrument, the electrometer being in the circuit, movements of the mercury column as considerable as those in Experiment 9 were observed.—F. J. M. P.

CHEMISTRY AND ALGEBRA

IT may not be wholly without interest to some of the readers of NATURE to be made acquainted with an analogy that has recently forcibly impressed me between branches of human knowledge apparently so dissimilar as modern chemistry and modern algebra. I have found it of great utility in explaining to non-mathematicians the nature of the investigations which algebraists are at present busily at work upon to make out the so-called *Grundformen* or irreducible forms appurtenant to binary quantics taken singly or in systems, and I have also found that it may be used as an instrument of investigation in purely algebraical inquiries. So much is this the case that I hardly ever take up Dr. Frankland's exceedingly valuable "Notes for Chemical Students," which are drawn up exclusively on the basis of Kekulé's exquisite conception of *valence*, without deriving suggestions for new researches in the theory of algebraical forms. I will confine myself to a statement of the grounds of the analogy, referring those who may feel an interest in the subject and are desirous for further information about it to a memoir which I have written upon it for the new *American Journal of Pure and Applied Mathematics*, the first number of which will appear early in February.

The analogy is between atoms and *binary* quantics exclusively.

I compare every binary quantic with a chemical atom. The number of factors (or rays, as they may be regarded by an obvious geometrical interpretation) in a binary quantic is the analogue of the number of *bonds*, or the *valence*, as it is termed, of a chemical atom.

Thus a linear form may be regarded as a monad atom, a quadratic form as a duad, a cubic form as a triad, and so on.

An invariant of a system of binary quantics of various degrees is the analogue of a chemical substance composed of atoms of corresponding *valences*. The order of such invariant in each set of coefficients is the same as the number of atoms of the corresponding *valence* in the chemical compound.

A co-variant is the analogue of an (organic or inorganic) compound radical. The orders in the several sets of coefficients corresponding, as for invariants, to the respective valences of the atoms, the free valence of the compound radical then becomes identical with the degree of the co-variant in the variables.

The weight of an invariant is identical with the number of the bonds in the chemicograph of the analogous chemical substance, and the weight of the leading term (or basic differentiant) of a co-variant is the same as the number of bonds in the chemicograph of the analogous compound radical. Every invariant and covariant thus becomes expressible by a *graph* precisely identical with a Kekuléan diagram or chemicograph. But not every chemicograph is an algebraical one. I show that by an application of the algebraical law of reciprocity every algebraical graph of a given invariant will represent the constitution in terms of the roots of a quantic of a type reciprocal to that of the given invariant of an invariant belonging to that reciprocal type. I give a rule for the geometrical multiplication of graphs, *i.e.* for constructing a *graph* to the product of in- or co-variants whose separate graphs are given. I have also ventured upon a hypothesis which, whilst in nowise interfering with existing chemicographical constructions, accounts for the seeming anomaly of the isolated existence as "monad molecules" of mercury, zinc, and arsenic—and gives a rational explanation of the "mutual saturation of bonds."

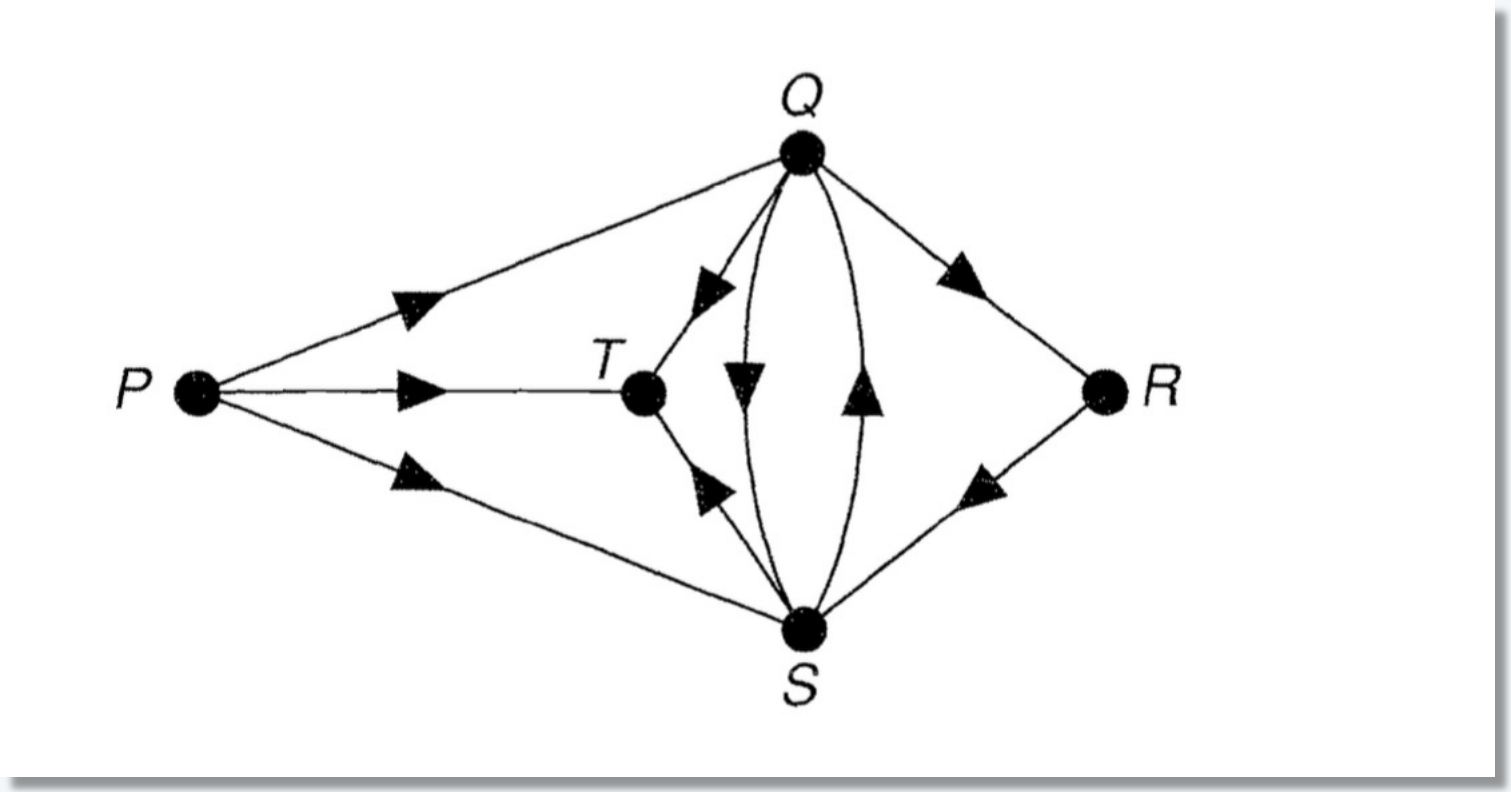
I have thus been led to see more clearly than ever I did before the existence of a common ground to the new mechanism, the new chemistry, and the new algebra. Underlying all these is the theory of pure colligation, which applies undistinguishably to the three great theories, all initiated within the last third of a century or thereabouts by Eisenstein, Kekulé, and Peaucellier.

Baltimore, January 1 J. J. SYLVESTER

PALMEN ON THE MORPHOLOGY OF THE TRACHEAL SYSTEM

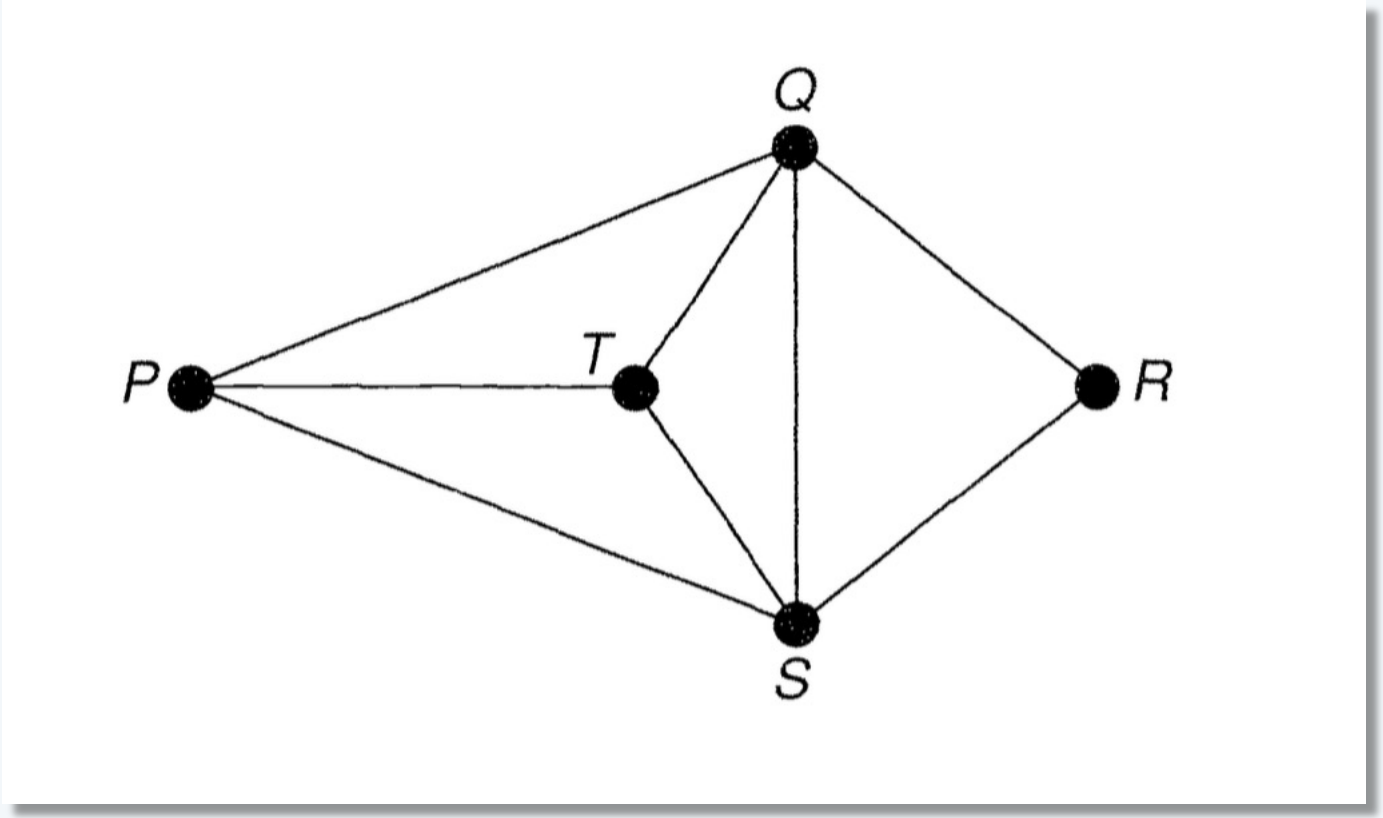
DR. PALMEN, of Helsingfors, has recently published an interesting memoir on the tracheal system of insects. He observes that although the gills of certain aquatic larvæ are attached to the skin very near to the points at which the spiracles open in the mature insects, and though spiracles and gills do not co-exist in the same segment, yet the point of attachment of the gills never exactly coincides with the position of the future spiracle. Moreover, he shows that even during the larval condition, although the spiracles are not open, the structure of the stigmatic duct is present, and indeed that it opens temporarily at each moult, to permit the inner tracheal membrane to be cast, after which it closes again. In fact, then, he urges, the gills and spiracles do not correspond exactly, either in number or in position, and there can therefore be between them no genetic connection. He concludes that the insects with open tracheæ are not derived from ancestors provided with gills,

Directed Graphs (Digraphs)



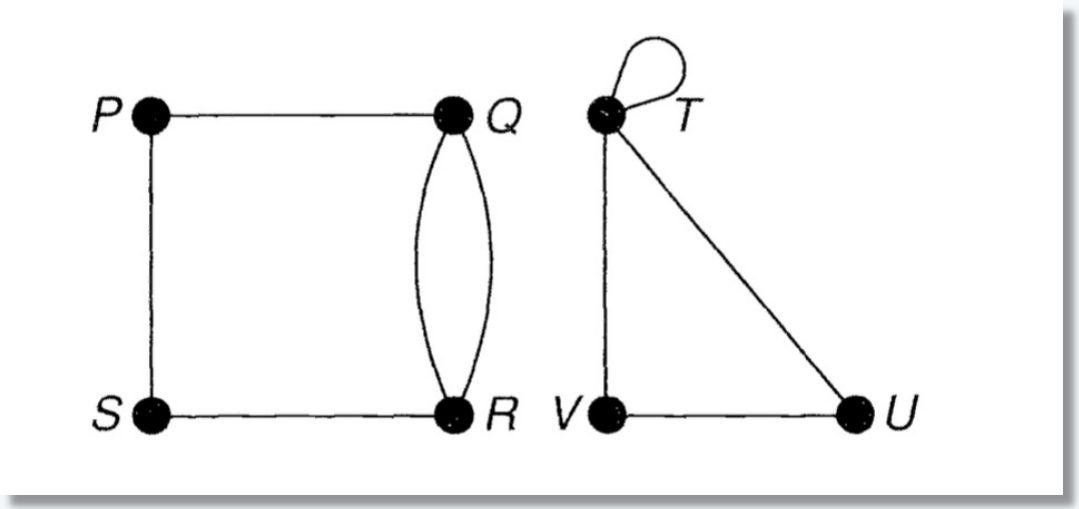
The study of **directed graphs** (or **digraphs,** as we abbreviate them) arises when making the roads into one-way streets. An example of a digraph is given above, the directions of the one-way streets being indicated by arrows. (In this example, there would be chaos at *T,* but that does not stop us from studying such situations!)

Walks, Paths, and Cycles



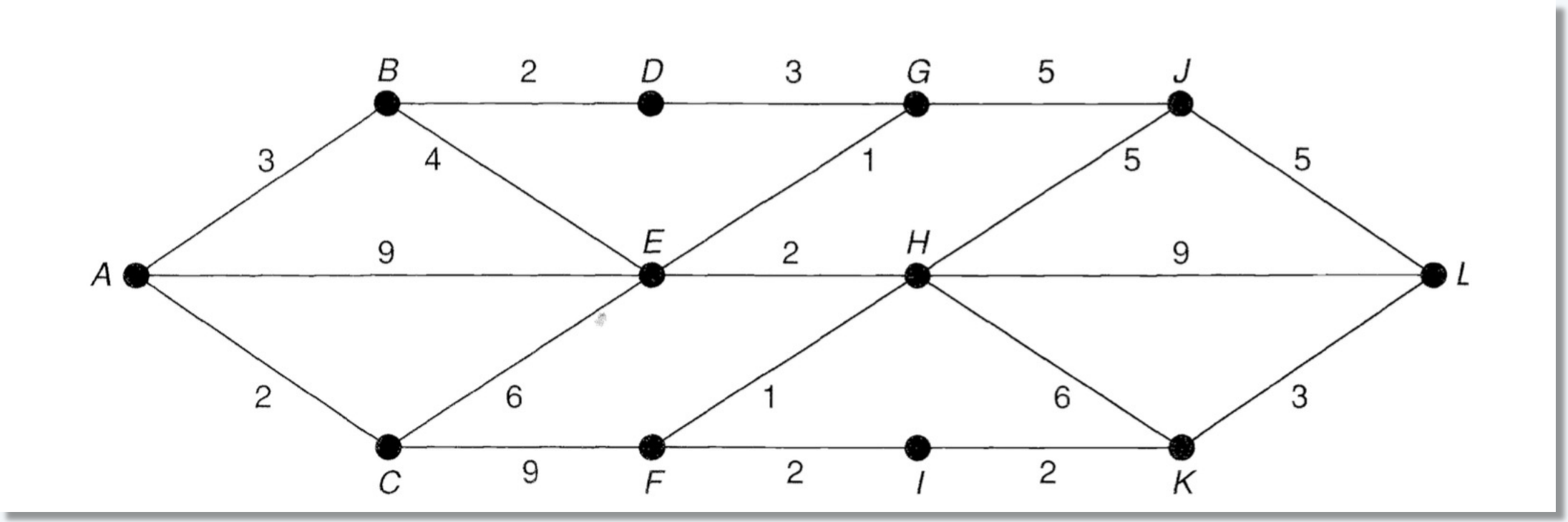
Much of graph theory involves 'walks' of various kinds. A **walk** is a 'way of getting from one vertex to another', and consists of a sequence of edges, one following after another. For example, in the above figure *P* —> *Q*—>*R* is a **walk of length 2**, and *P* —> *S* —> *Q* —> *T* —> *S* —> *R* is a walk of length 5. A walk in which no vertex appears more than once is called a **path**; for example and *P* —> *Q* —> *R* —> *S* is a path. A walk in which you end where you started, for example *Q* —> *S* —> *T* —> Q , is called a **cycle**.

Connectedness



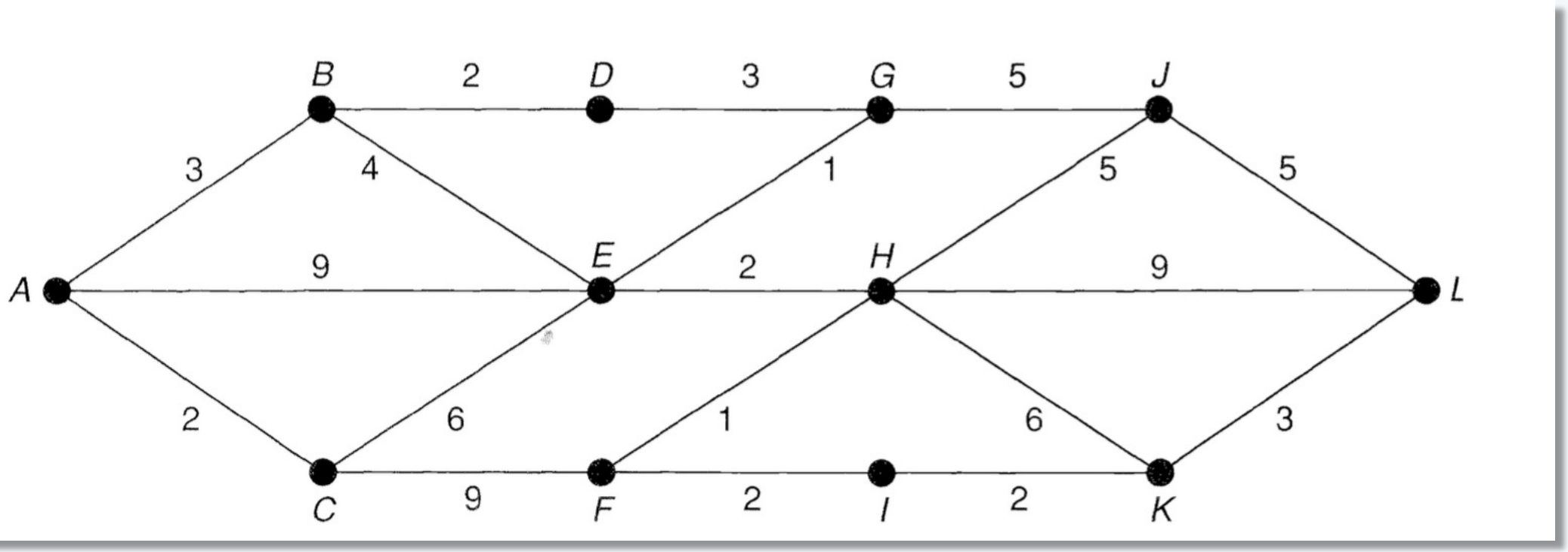
Some graphs are in two or more parts. A graph that is in one piece, so that **any two vertices are connected by a path**, is a **connected graph**; a graph in more than one piece is a **disconnected graph.**

Weighted Graphs



Consider the above graph: it is a connected graph in which a non-negative number is assigned to each edge. Such a graph is called a **weighted graph,** and the number assigned to each edge *e* is the **weight** of *e,* denoted by *w(e)*.

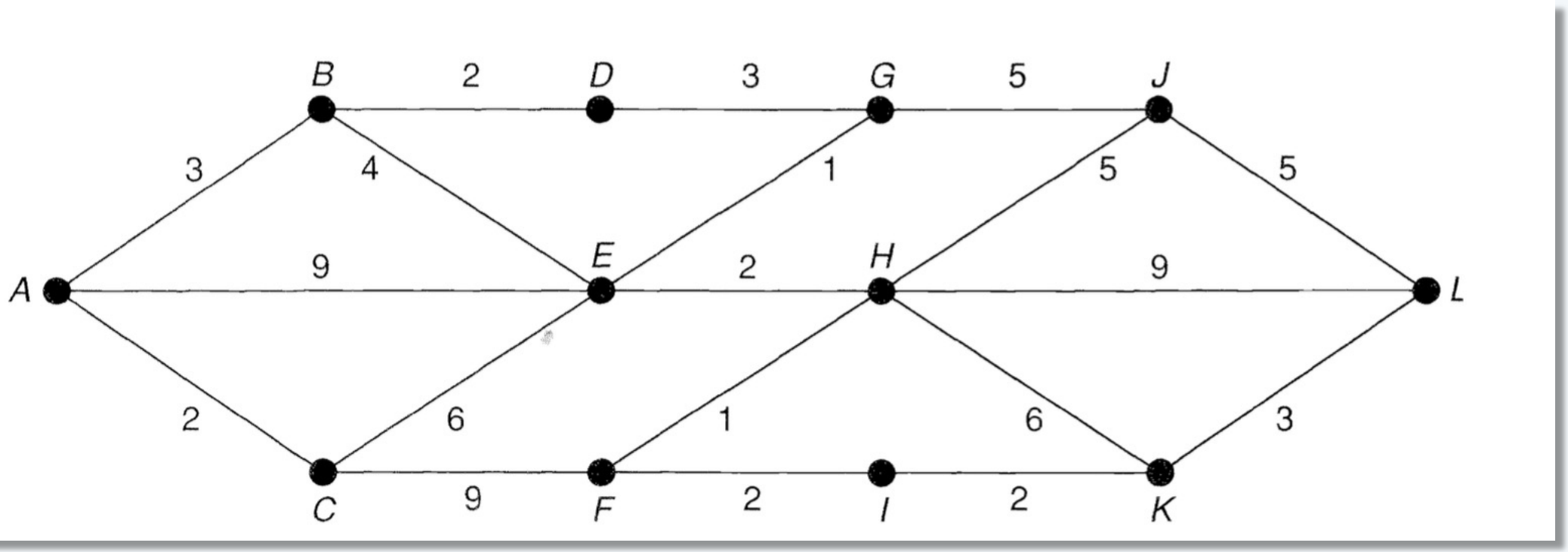
Shortest Path (between *one* pair of vertices)



What is the **length of the shortest path (=distance)** from A to L?

The problem is to find a path from A to L with minimum total weight. This problem is called the **Shortest Path Problem**. Note that, if we have a weighted graph in which each edge has weight 1, then the problem reduces to that of finding the number of edges in the shortest path from A to L.

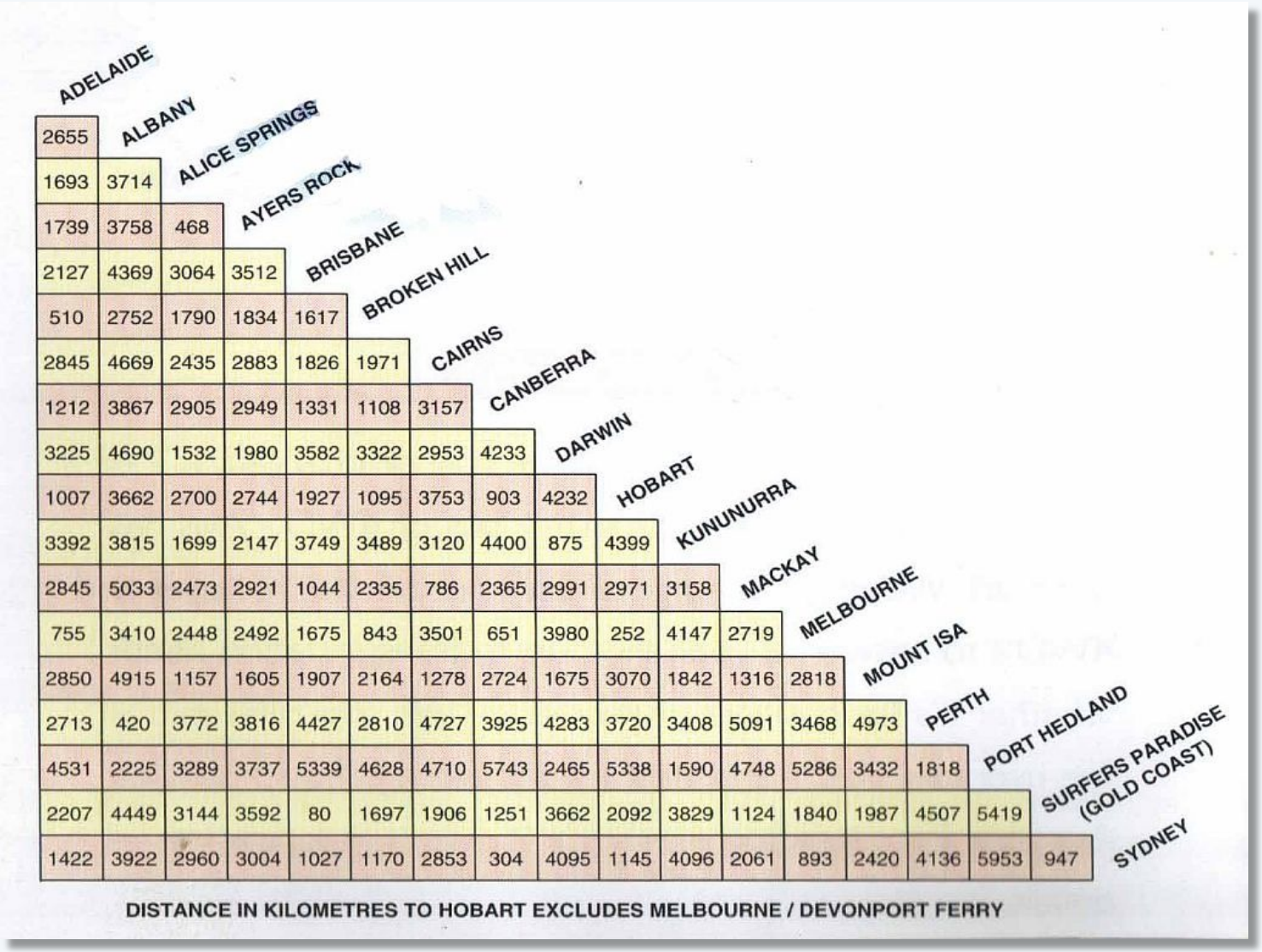
All-Pairs Shortest Path



What is the length of the shortest path (=distances) from **any vertex to any vertex**?

This problem is called the **All-Pairs Shortest Path Problem**

All-Pairs Shortest Path : A Solution for Some Cities in Australia



ADELAIDE	ALBANY	ALICE SPRINGS	AYERS ROCK	BRISBANE	BROKEN HILL	CAIRNS	CANBERRA	DARWIN	HOBART	KUNUNURRA	MACKAY	MELBOURNE	MOUNT ISA	PERTH	PORT HEDLAND	SURFERS PARADISE (GOLD COAST)	SYDNEY
2655																	
1693	3714																
1739	3758	468															
2127	4369	3064	3512														
510	2752	1790	1834	1617													
2845	4669	2435	2883	1826	1971												
1212	3867	2905	2949	1331	1108	3157											
3225	4690	1532	1980	3582	3322	2953	4233										
1007	3662	2700	2744	1927	1095	3753	903	4232									
3392	3815	1699	2147	3749	3489	3120	4400	875	4399								
2845	5033	2473	2921	1044	2335	786	2365	2991	2971	3158							
755	3410	2448	2492	1675	843	3501	651	3980	252	4147	2719						
2850	4915	1157	1605	1907	2164	1278	2724	1675	3070	1842	1316	2818					
2713	420	3772	3816	4427	2810	4727	3925	4283	3720	3408	5091	3468	4973				
4531	2225	3289	3737	5339	4628	4710	5743	2465	5338	1590	4748	5286	3432	1818			
2207	4449	3144	3592	80	1697	1906	1251	3662	2092	3829	1124	1840	1987	4507	5419		
1422	3922	2960	3004	1027	1170	2853	304	4095	1145	4096	2061	893	2420	4136	5953	947	

DISTANCE IN KILOMETRES TO HOBART EXCLUDES MELBOURNE / DEVONPORT FERRY

Real World vs Model vs Representation vs Implementation

The Real World

That messy thing we are trying to study (with computers).

Model

Mathematical object in some class M.

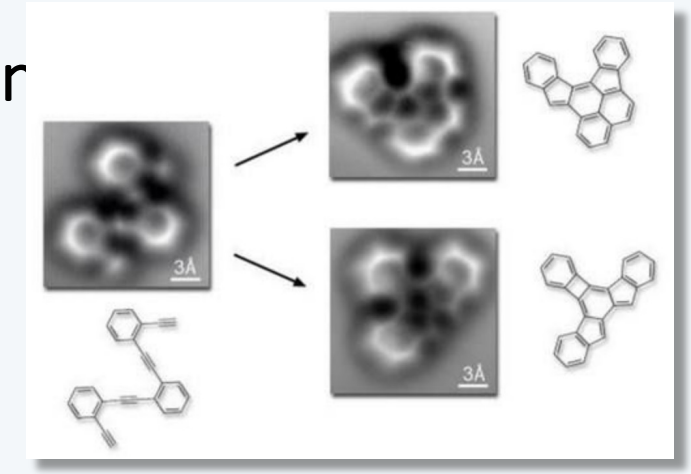
Representation

An object of an abstract data type R used to store the model.

Implementation

An object of a concrete type used to store the model.

Any object from the real world might have different models.
Any model might have several representations (exact).
And representation might have different implementations (exact).

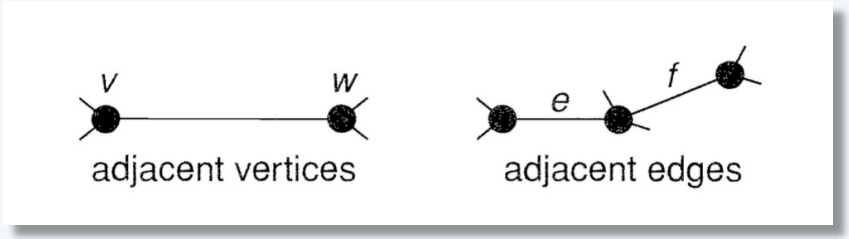


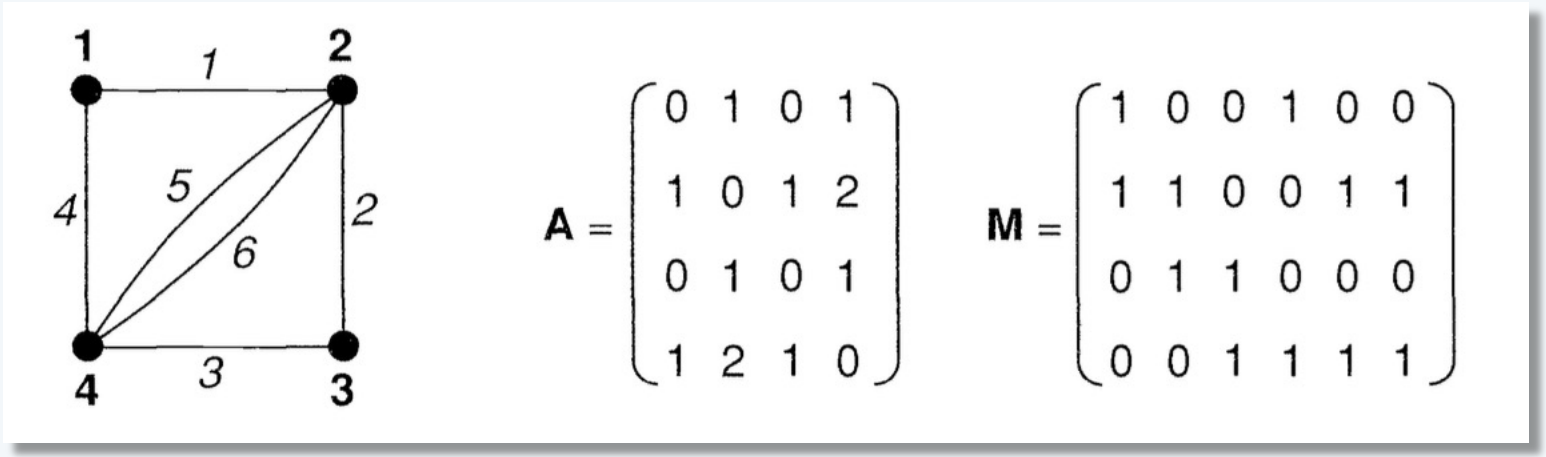
$$G = (V, E)$$

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

```
>>> import scipy as sp
>>> G = nx.Graph([(1,1)])
>>> A = nx.adjacency_matrix(G)
```

Matrix Representations for Graphs



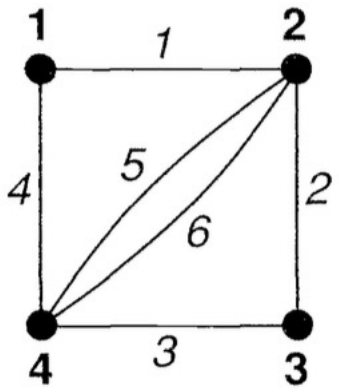


$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

If *G* is a graph with vertices labelled {1, 2, ...}, its **adjacency matrix A** is the *n* x *n* matrix whose ij-th entry is the number of edges joining vertex ***i*** and vertex ***j***. Two nodes ***i*** and ***j*** are adjacent if the ij-th entry in the adjcacency matrix is larger than 0.

If, in addition to the vertices, the edges are labelled {1, 2,..., *m}*, its **incidence matrix M** is the *n* x *m* matrix whose *ij-th* entry is 1 if **vertex i is incident to edge j** and 0 otherwise. The figure above shows a labelled graph *G* with its adjacency and incidence matrices.

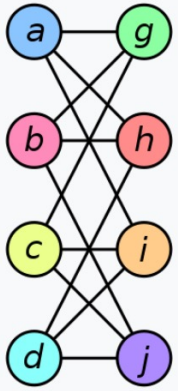
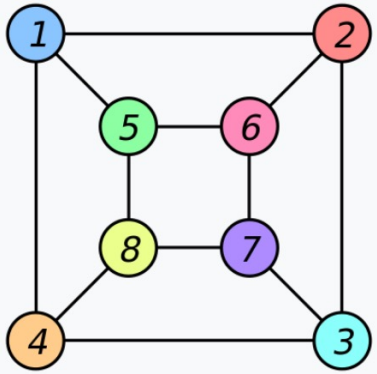
Adjacency Matrix for Weighted Graphs



$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Given a weighted graph G, the **adjacency matrix A** is the matrix whose ij-th entry is the weight of the edge between vertex ***i*** and vertex ***j***.

Graph Isomorphism

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

Are these 2 graphs isomorphic? YES!

[advanced comment]
Computational Complexity:
Probably "quasi-polynomial"

https://www.youtube.com/watch?v=qYIhA3O9Nz0

(to be confirmed)

Graph Isomorphism
Two graphs *G* and *H* are isomorphic if there is a one-one correspondence between the vertices of *G* and those of H such that structure (i.e., adjacency) is preserved (i.e., the number of edges joining any two vertices of *G* is equal to the number of edges joining the corresponding vertices of H.)

Checking Isomorphism always requires to compare representations of two graphs!

Note: we assume here unlabeled undirected graphs only. The vertices of a graph still have to be identified.

Permutation Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

In mathematics a **permutation matrix** is a square binary matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere. Each such matrix, say *P*, represents a permutation of *m* elements and, when used to multiply another matrix, say *B*, results in permuting the rows (when pre-multiplying, i.e., *PB*) or columns (when post-multiplying, *BP*) of the matrix *B*.

Multiplying a Matrix with a Permutation Matrix P (from the right)

column 2 becomes column 4

$$\begin{pmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 11 & 21 & 6 & 16 \\ 2 & 12 & 22 & 7 & 17 \\ 3 & 13 & 23 & 8 & 18 \\ 4 & 14 & 24 & 9 & 19 \\ 5 & 15 & 25 & 10 & 20 \end{pmatrix}$$

Let B be a matrix and let P be a permutation matrix:

The multiplication BP (multiplying the permutation matrix **from the right**) **permutes the columns** of B

$$1 \leftarrow 1$$
$$4 \leftarrow 2$$
$$2 \leftarrow 3$$
$$5 \leftarrow 4$$
$$3 \leftarrow 5$$

Multiplying a Matrix with a Permutation Matrix P (from the left)

row 4 becomes row 2

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{1} & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 11 & 16 & 21 \\ 4 & 9 & 14 & 19 & 24 \\ 2 & 7 & 12 & 17 & 22 \\ 5 & 10 & 15 & 20 & 25 \\ 3 & 8 & 13 & 18 & 23 \end{pmatrix}$$

Let B be a matrix and let P be a permutation matrix:

The multiplication PB (multiplying the permutation matrix **from the left**) **permutes the rows** of B

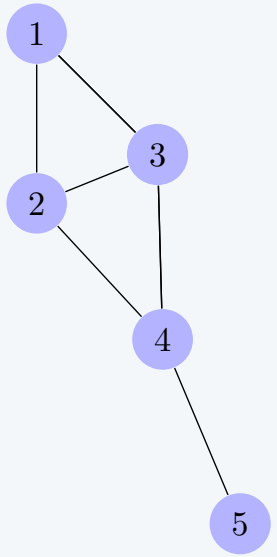
$$1 \to 1$$
$$4 \to 2$$
$$2 \to 3$$
$$5 \to 4$$
$$3 \to 5$$

Changing the representation of a graph via $P(PB)^T$

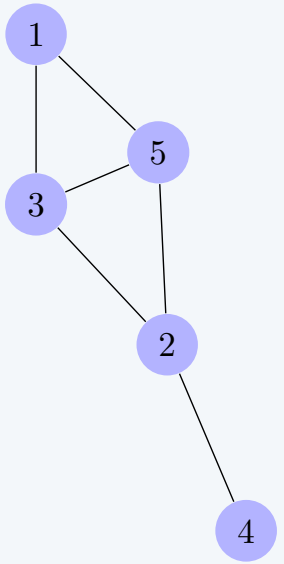
$$\underbrace{\underbrace{\begin{pmatrix}1&0&0&0&0\\0&0&0&1&0\\0&1&0&0&0\\0&0&0&0&1\\0&0&1&0&0\end{pmatrix}}_{P}\times\underbrace{\left(\underbrace{\begin{pmatrix}1&0&0&0&0\\0&0&0&1&0\\0&1&0&0&0\\0&0&0&0&1\\0&0&1&0&0\end{pmatrix}}_{P}\times\underbrace{\begin{pmatrix}0&1&1&0&0\\1&0&1&1&0\\1&1&0&1&0\\0&1&1&0&1\\0&0&0&1&0\end{pmatrix}}_{B}\right)}_{\text{row permuted } B}}_{\text{transposed row permuted } B}}_{\text{row permuted column permuted } B}=\begin{pmatrix}0&0&1&0&1\\0&0&1&1&1\\1&1&0&0&1\\0&1&0&0&0\\1&1&1&0&0\end{pmatrix}$$

$P(PB)^T$ re-labels the graph represented by adjacency matrix B

The mathematical object remains identical (the represented graphs are isomorphic), only the representation of the graph changes (the adjacency matrix representations are potentially different).



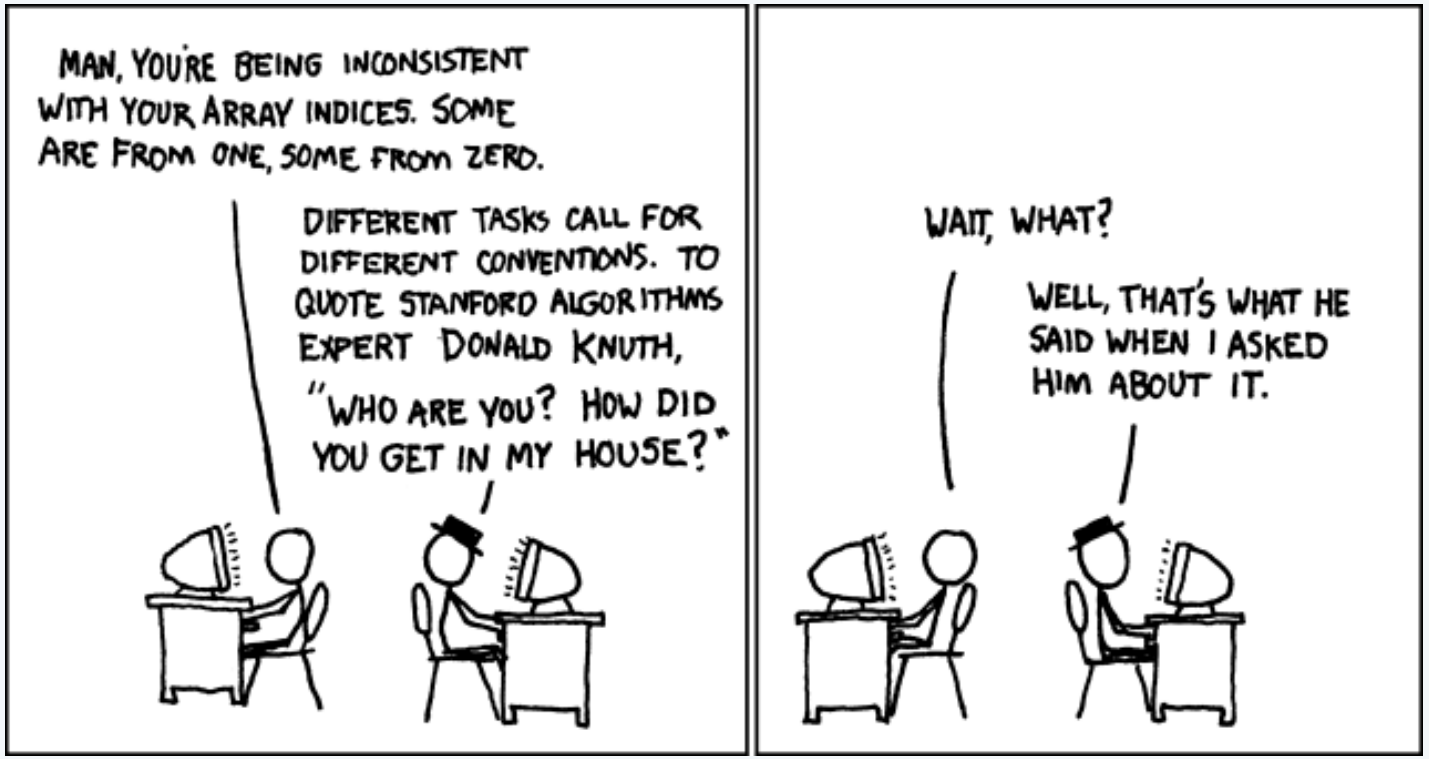
$$1 \to 1$$
$$4 \to 2$$
$$2 \to 3$$
$$5 \to 4$$
$$3 \to 5$$



Zero-Indexing

Zero-based numbering is a way of numbering in which the initial element of a sequence is assigned the index 0, rather than the index 1 as is typical in everyday non-mathematical/non-programming circumstances.

Make sure that it is clear what you mean, when you say, e.g., the “row with index 1” in a matrix.



(picture from xkcd.com)

Python

Using:

- `matplotlib` for plotting
- `networkx` for graph theory

More efficient solution in Exercises (and hopefully in lab submission)

Code provided :

`graph-permutation.py`

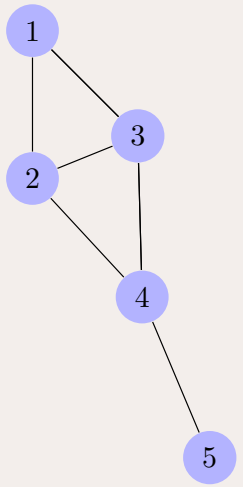
```
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx

# Note, there might be several differnent adjacency matrices for the same graph
B=np.array([[ 0,  1,  1,  0,  0],
            [ 1,  0,  1,  1,  0],
            [ 1,  1,  0,  1,  0],
            [ 0,  1,  1,  0,  1],
            [ 0,  0,  0,  1,  0]])

P=np.array([[ 1,  0,  0,  0,  0],
            [ 0,  0,  0,  1,  0],
            [ 0,  1,  0,  0,  0],
            [ 0,  0,  0,  0,  1],
            [ 0,  0,  1,  0,  0]])

# 1->1                    0->0                [[ 1,  0,  0,  0,  0],
# 4->2                    3->1                 [ 0,  0,  0,  1,  0],
# 2->3                    1->2                 [ 0,  1,  0,  0,  0],
# 5->4                    4->3                 [ 0,  0,  0,  0,  1],
# 3->5                    2->4                 [ 0,  0,  1,  0,  0]]


# for drawing the graph, only
G = nx.from_numpy_matrix(B)
plt.figure()
plt.subplot(211)

# the annoyingness of zero- and one- indexing let's rename the depicted labels
mathLabels = {0: 1, 1:2, 2:3, 3:4, 4:5}
nx.draw(G, with_labels=True, labels=mathLabels)

# As taught in the lecture P*(P*B)^T
A = np.matmul(P,np.transpose(np.matmul(P,B)))

# Since python 3.5 the following also works (note the stronger binding of
# the transpose operator
A = P@(P@B).T

# More clever (see Appendix C or the numpy documentation)
# define how rows should be permuted
# r = np.array([0, 3, 1, 4, 2])
# A = (B[np.array(r)].T)[np.array(r)]

H = nx.from_numpy_matrix(A)
plt.subplot(212)
nx.draw(H, with_labels=True, labels=mathLabels)
plt.show()
```

Brute Force Graph Isomorphism Check: $P(PB)^T$

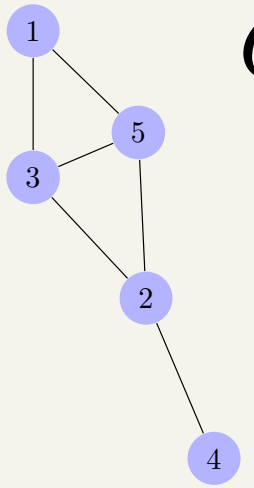
Theorem:
Two undirected graphs G_A and G_B (with adjacency matrices A and B) are isomorphic, if and only if there exists a permutation matrix P, such that $A = P\,(PB)^T$.

$1 \to 1$
$4 \to 2$
$2 \to 3$
$5 \to 4$
$3 \to 5$

G_B



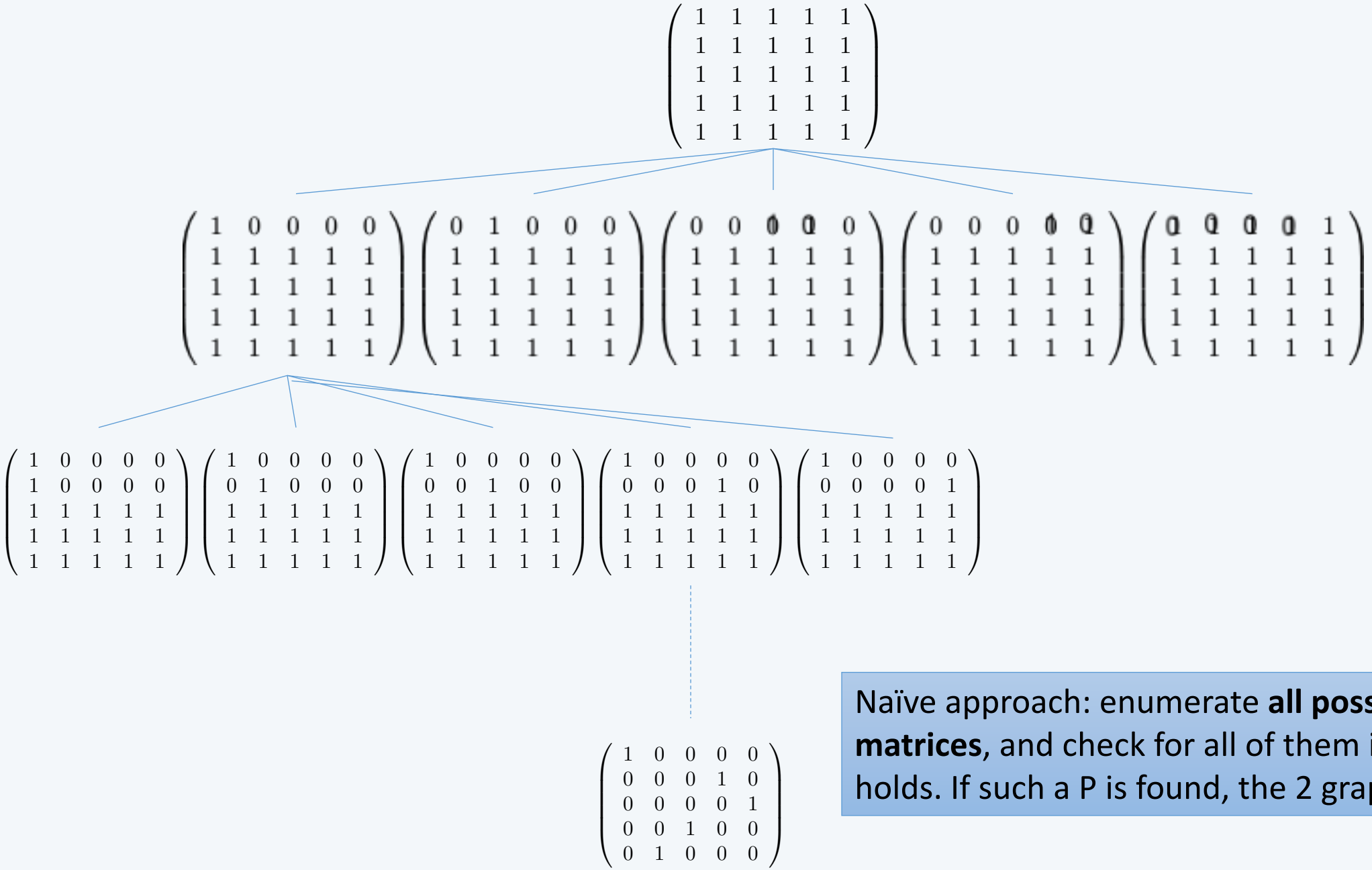
G_A



$$\begin{pmatrix} 1&0&0&0&0\\ 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&0&0&1\\ 0&0&1&0&0 \end{pmatrix} \times \left(\left(\begin{pmatrix} 1&0&0&0&0\\ 0&0&0&1&0\\ 0&1&0&0&0\\ 0&0&0&0&1\\ 0&0&1&0&0 \end{pmatrix} \times \begin{pmatrix} 0&1&1&0&0\\ 1&0&1&1&0\\ 1&1&0&1&0\\ 0&1&1&0&1\\ 0&0&0&1&0 \end{pmatrix} \right)^T = \begin{pmatrix} 0&0&1&0&1\\ 0&0&1&1&1\\ 1&1&0&0&1\\ 0&1&0&0&0\\ 1&1&1&0&0 \end{pmatrix}$$

Problem:
How to find P?

Brute Force Graph Isomorphism Check: $A = P\,(PB)^T$



Naïve approach: enumerate **all possible permutation matrices**, and check for all of them if $A = P\,(PB)^T$ holds. If such a P is found, the 2 graphs are isomorphic.

Improvement(s)

Consider node degrees!

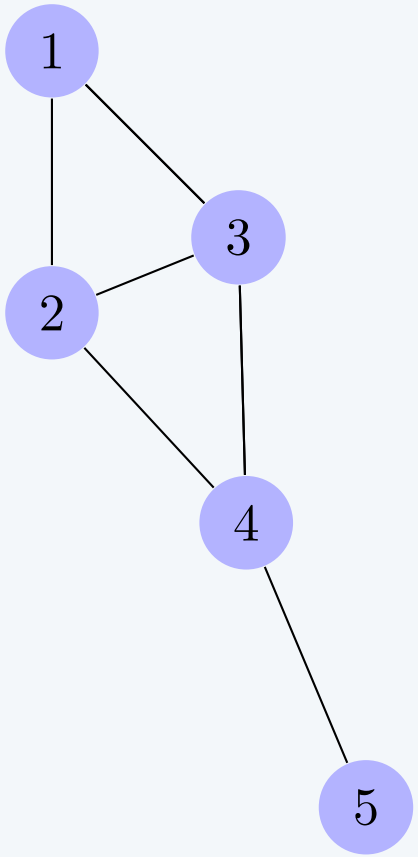
Does it make sense to consider a P which maps

- vertex 1 of G_B to vertex 2 of G_A ? NO!
- vertex 1 to vertex 1? YES!
- vertex 1 to vertex 3, 4, or 5? NO!
- vertex 2 to vertex 1 or 4? NO!
- vertex 2 to vertex 2, 3, or 5? YES!

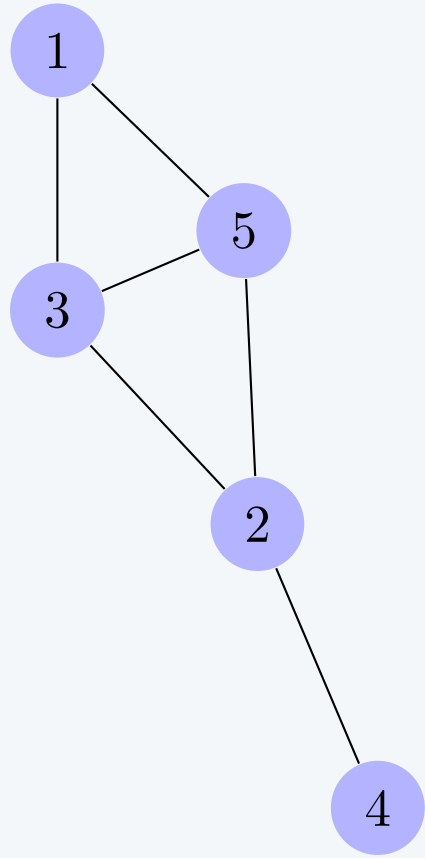
Use as initial matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

G_B



G_A



Improved Graph Isomorphism Check: $A = P\,(PB)^T$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \boxed{0 & 1 & 1 & 1 & 0} \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

No branching for first row!
Consider all (3) possibilities for row 2.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \boxed{0 & 1 & 1 & 1 & 0} \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \boxed{0 & 1 & 1 & 1 & 0} \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \boxed{0 & 1 & 1 & 1 & 0} \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Consider all (3) possibilities for row 3.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Always a good idea:
Reduce the size of the search tree

vertex 4 of B cannot be mapped to vertex 2 of A **and** to vertex 3 of A

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Significantly smaller number of possible permutation matrices!

Applications [wikipedia]

Graphs are commonly used to encode structural information in many fields, including computer vision and pattern recognition, and graph matching, i.e., identification of similarities between graphs, is an important tools in these areas. In these areas graph isomorphism problem is known as the exact graph matching.[39]

Graph matching

In cheminformatics and in mathematical chemistry, graph isomorphism testing is used to identify a chemical compound within a chemical database.[40] Also, in organic mathematical chemistry graph isomorphism testing is useful for generation of molecular graphs and for computer synthesis.

Chemical database search is an example of graphical data mining, where the graph canonization approach is often used.[41] In particular, a number of identifiers for chemical substances, such as SMILES and InChI, designed to provide a standard and human-readable way to encode molecular information and to facilitate the search for such information in databases and on the web, use canonization step in their computation, which is essentially the canonization of the graph which represents the molecule.

In electronic design automation graph isomorphism is the basis of the Layout Versus Schematic (LVS) circuit design step, which is a verification whether the electric circuits represented by a circuit schematic and an integrated circuit layout are the same.[42]

State-of-the-Art Implementation for Checking for Graph Isomorphism

State-of-the-art approaches to test if two graphs are isomorphic can compute an answer for graphs with thousands of nodes in milliseconds. If you are interested, see for example here https://dl.acm.org/doi/10.1145/3356020 for a very nice visualization accompanying this scientific paper, see https://jakobandersen.github.io/graph_canon_vis/

Advanced Comment: Find a **canonical representation**

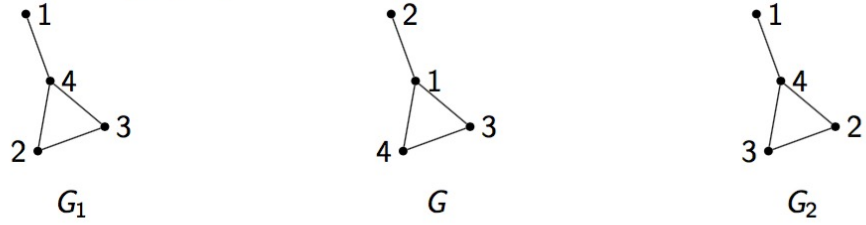
Given a representation $G \in R$ find a new representation $C(G)$, a canonical form, such that:

- It represents the same model: $C(G) \cong G$
- All canonicalized isomorphic representations are the same: $\forall G' \in R, G' \cong G: \quad C(G') \stackrel{r}{=} C(G)$

Example: Chose the "smallest" adjacency matrix representation As the canonical representation.

$G = (V, E) \quad V = \{1, 2, \ldots, n\}$

Isomorphic graphs, different representations:



Adjacency matrix representation:

	1	2	3	4
1				1
2			1	1
3		1		1
4	1	1	1	

	1	2	3	4
1		1	1	1
2	1			
3	1			1
4	1		1	

	1	2	3	4
1				1
2			1	1
3		1		1
4	1	1	1	

Subgraph Isomorphism

The formula $A = P\,(PB)^T$ can also be used to check if a graph H is a subgraph of graph G.

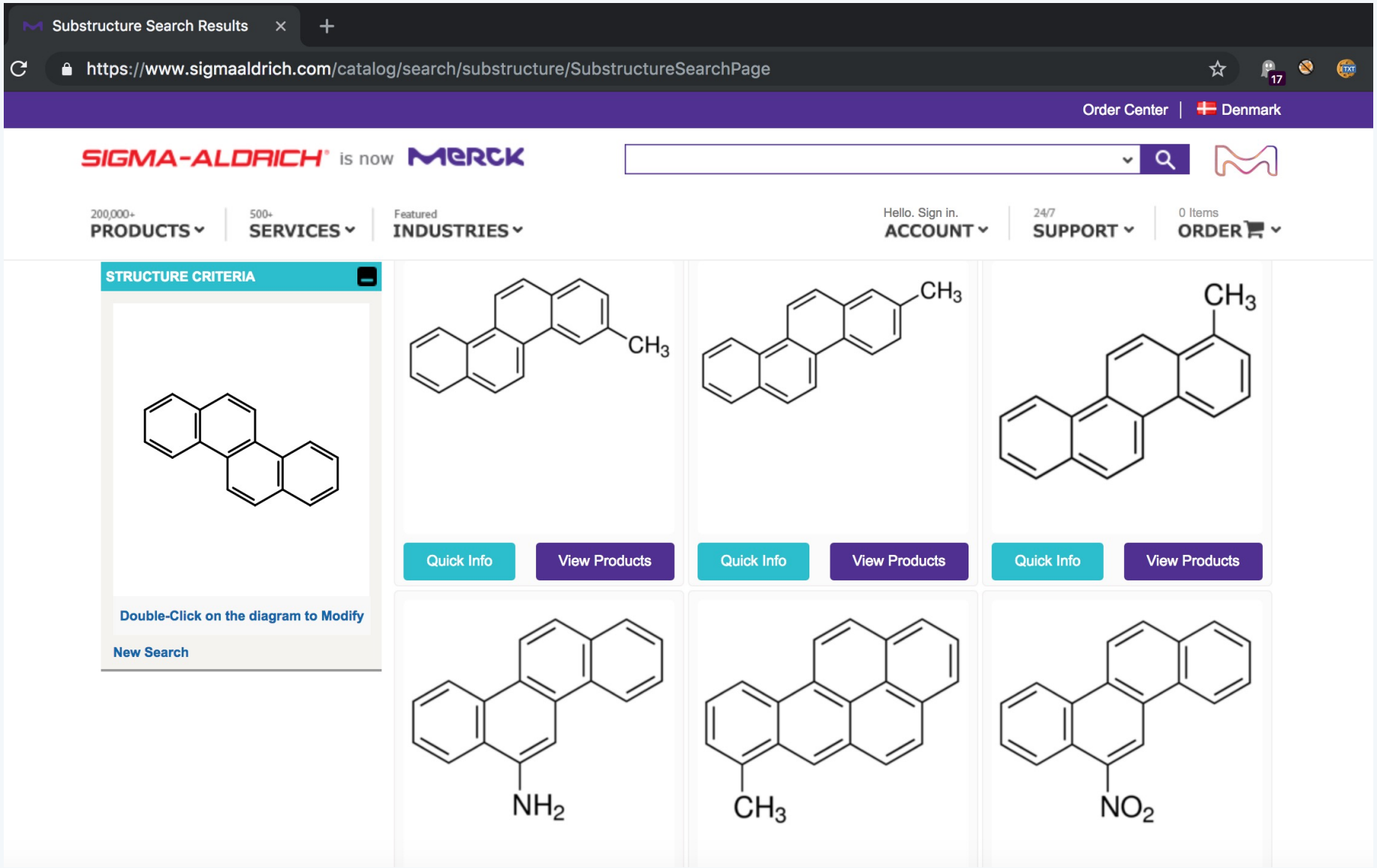
This will be discussed in the exercise session, the slides will be included in this document.

Applications

Approx. 70 mio. chemical compounds are know.

- How to find them in databases?
- How to name them?
- Where to buy them?
- How to curate databases of compounds?
- ...

See also https://math.stackexchange.com/questions/120408/what-are-the-applications-of-the-isomorphic-graphs for a nice list of the endless application scenarios of isomorphism checking.

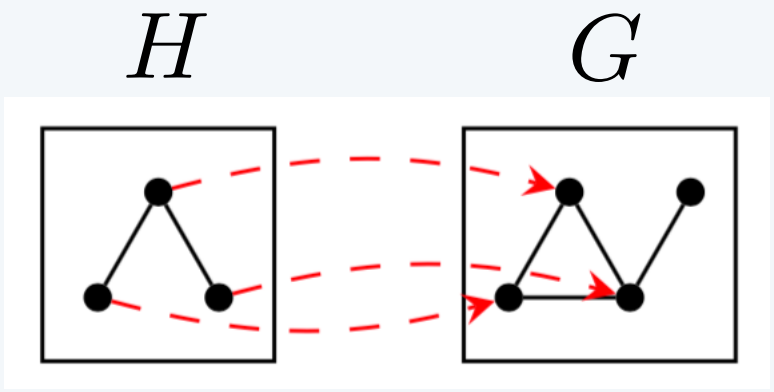


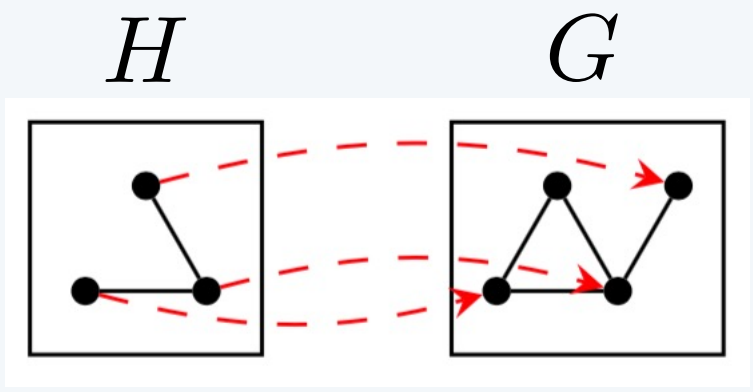
Subgraph Isomorphism (-> Exercises)

Formally, let $G = (V, E)$ be any graph, and let $S \subset V$ be any subset of vertices of G.

The subgraph H is a graph whose vertex set is S and whose edge set consists of (not necessarily all) edges in E that have both endpoints in S.

The *induced* subgraph H is a graph whose vertex set is S and whose edge set consists of *all* the edges in E that have both endpoints in S.





Obviously H is a subgraph of G (see left figure). Is it also an induced subgraph of H? Yes! (right)