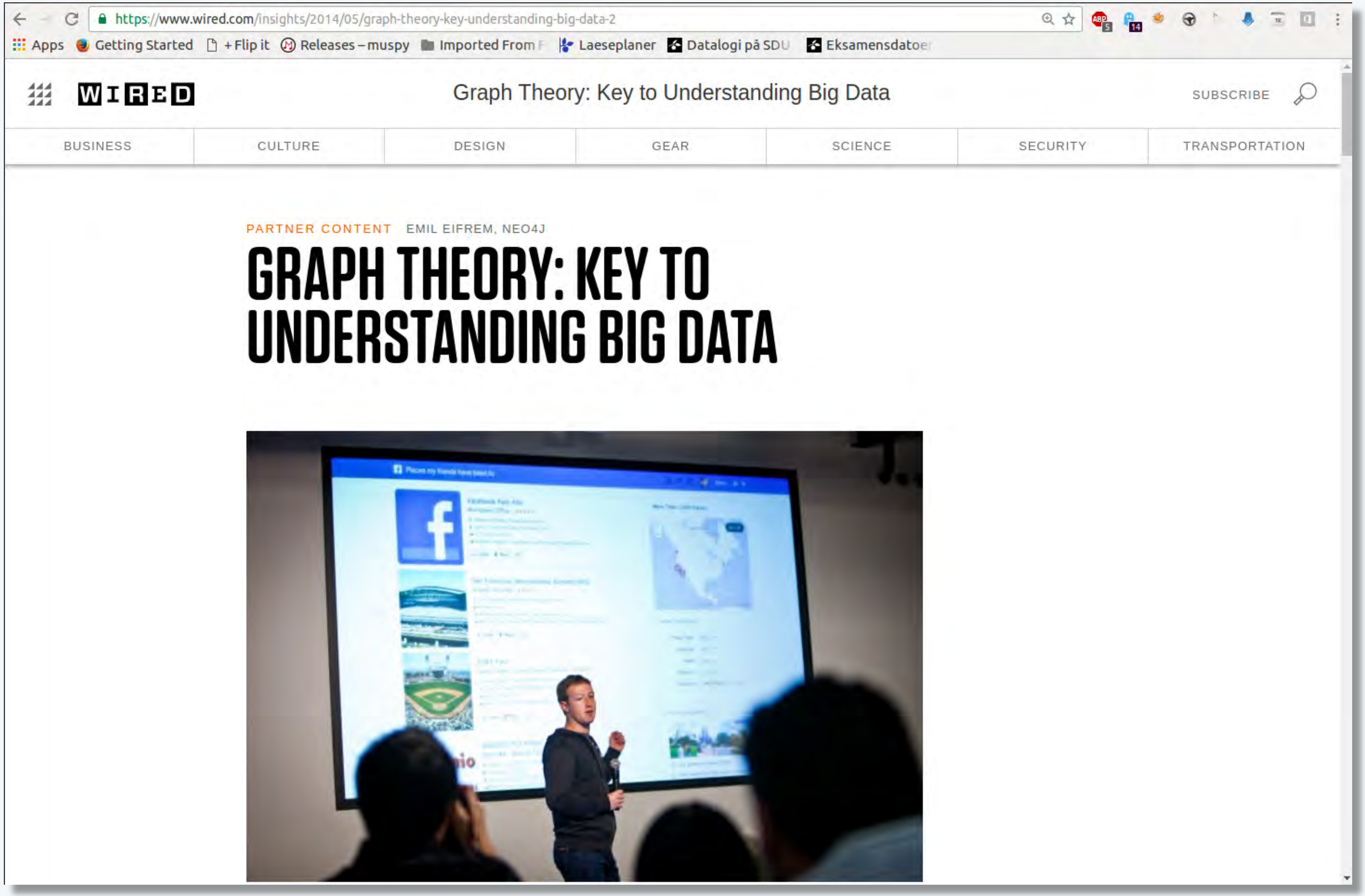
DM534 - Introduction to Computer Science, Week 48

Graph Theory

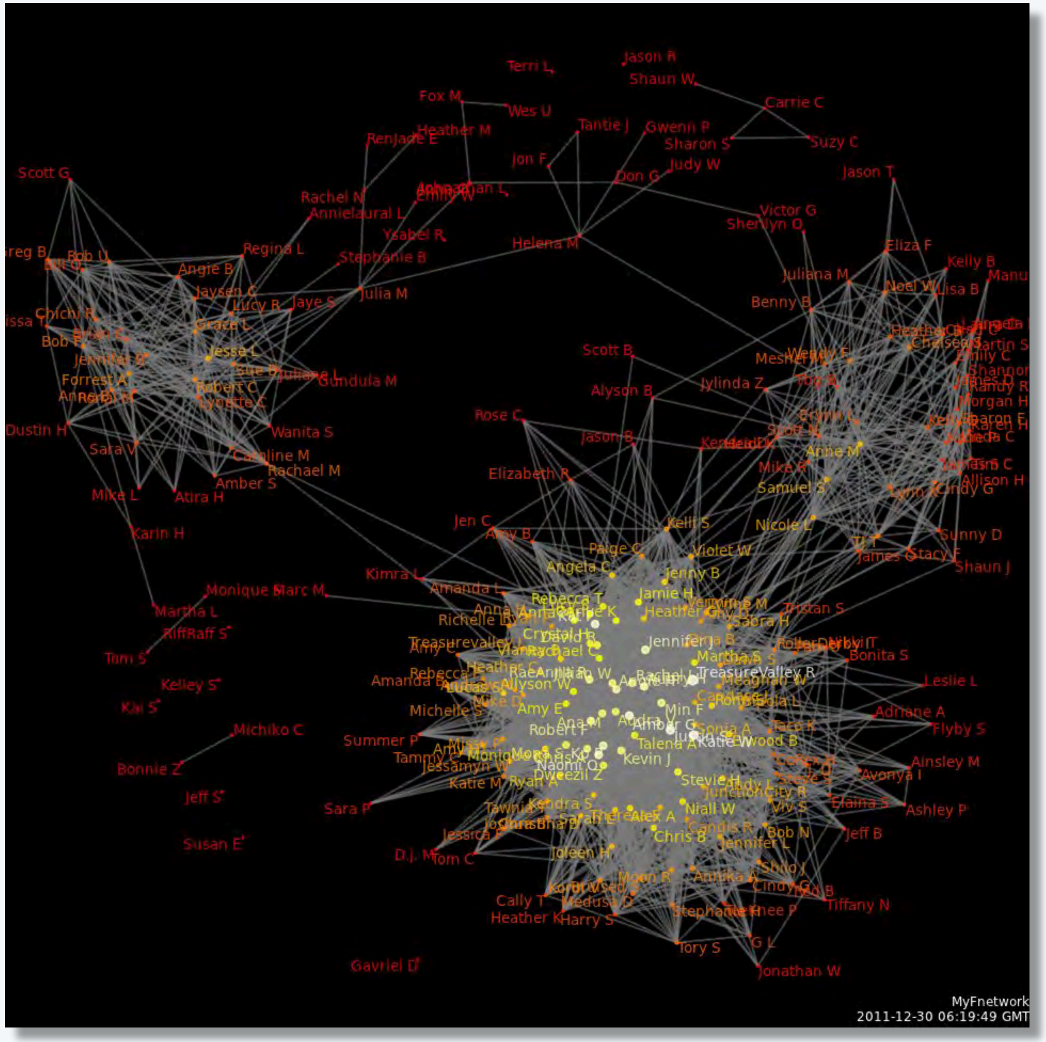
Daniel Merkle

daniel@imada.sdu.dk

Graph Theory - Motivation

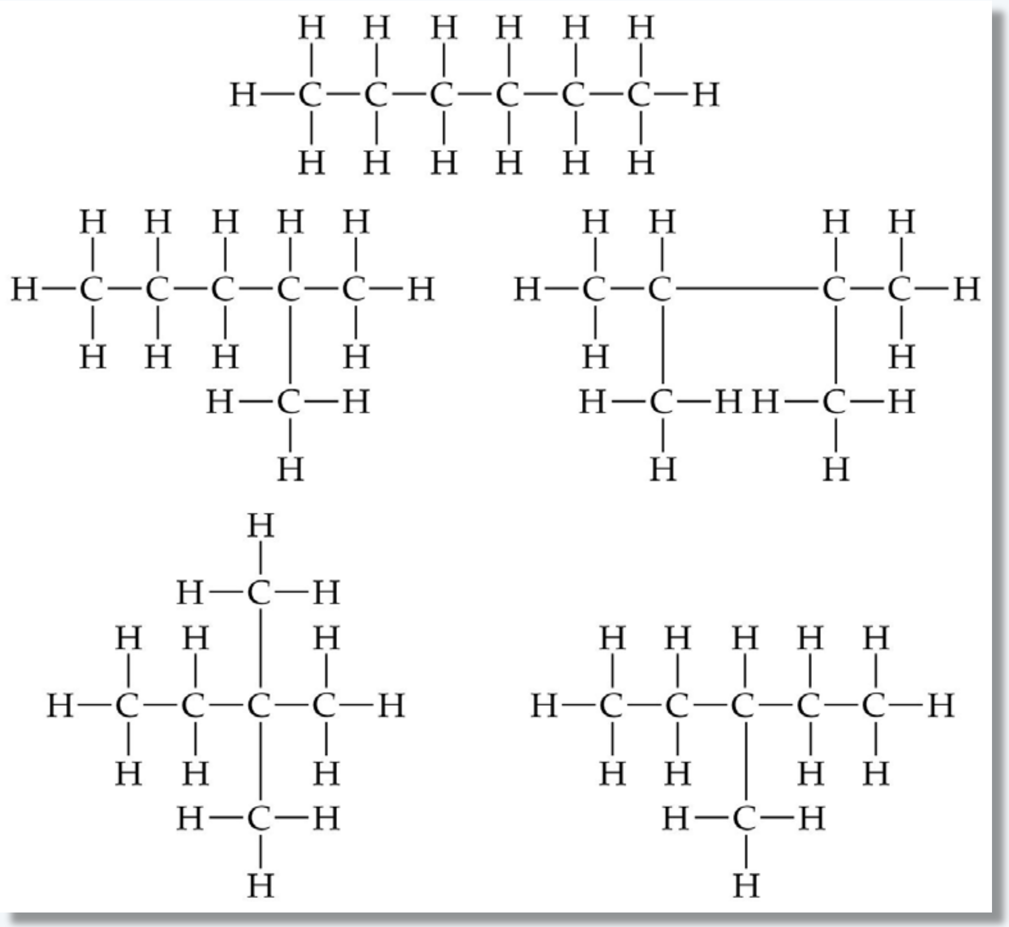


Social Networks



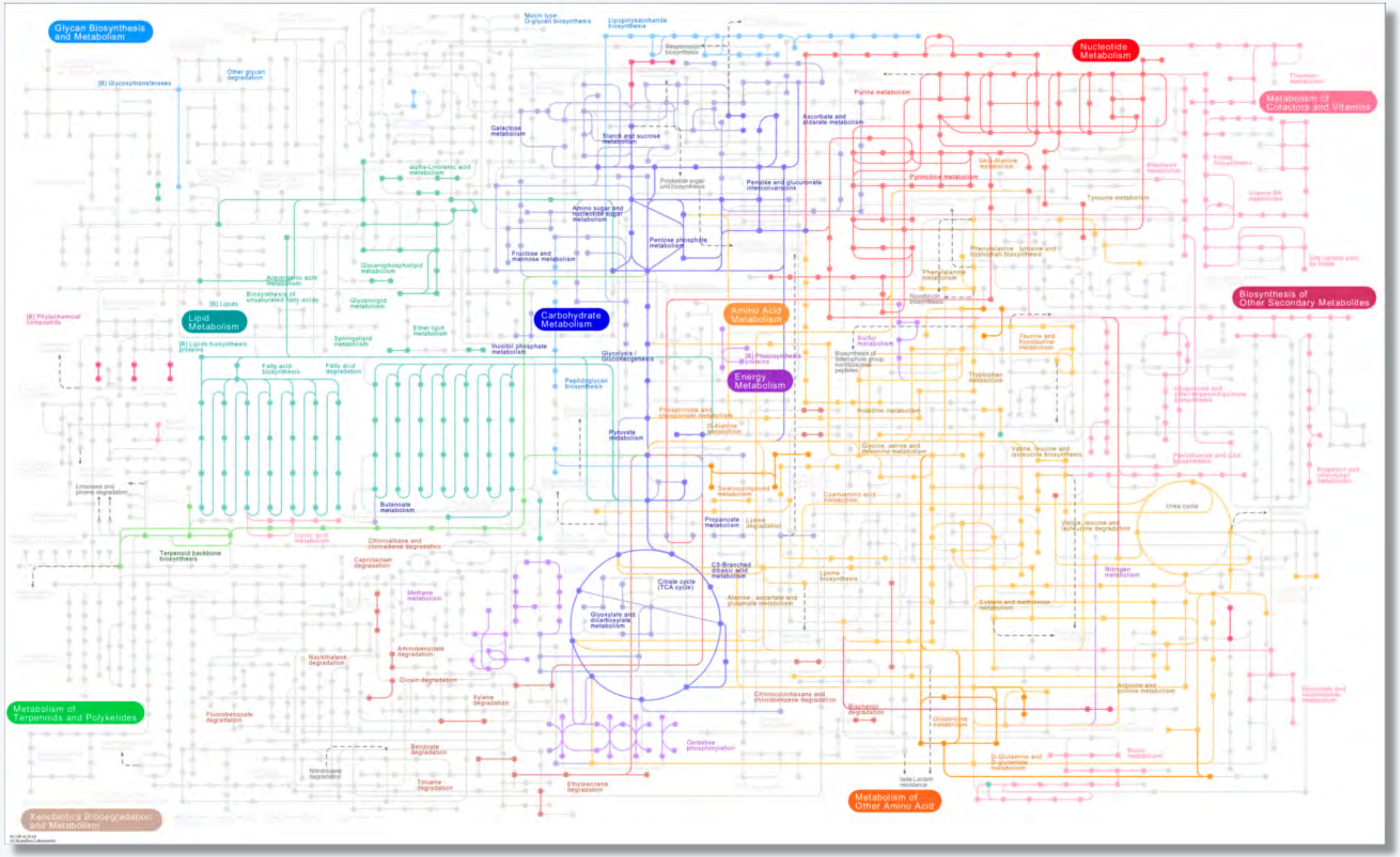
This graph might depict Facebook friendship relations, or Twitter follower relations, or…

Chemical Compounds



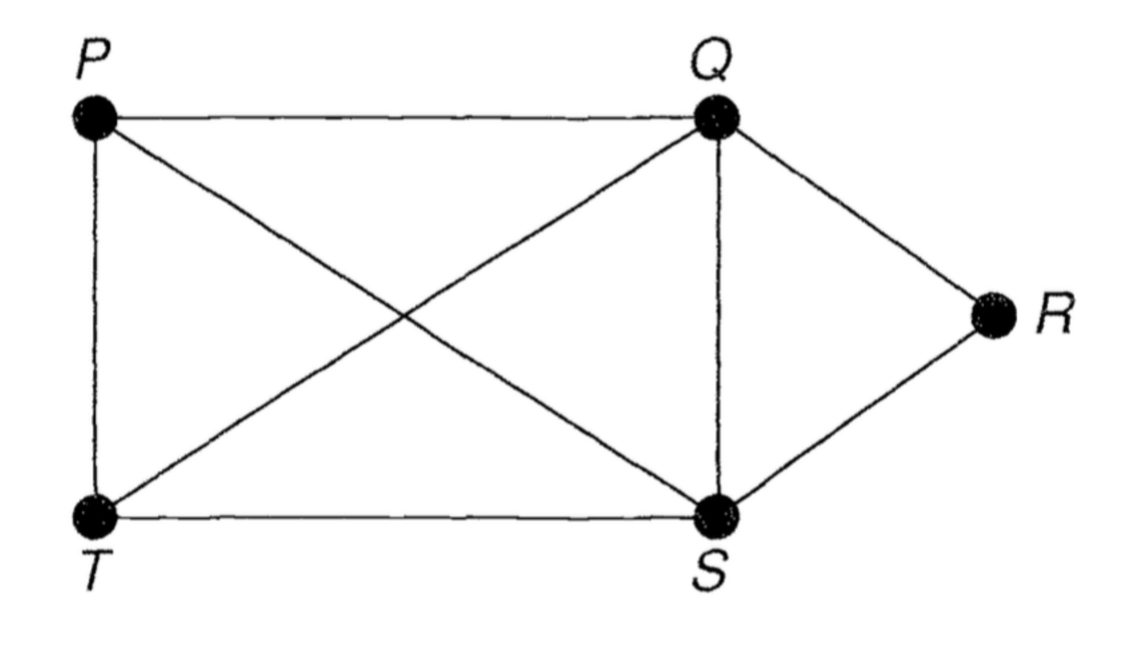
Isomers of Hexane

Metabolic Networks



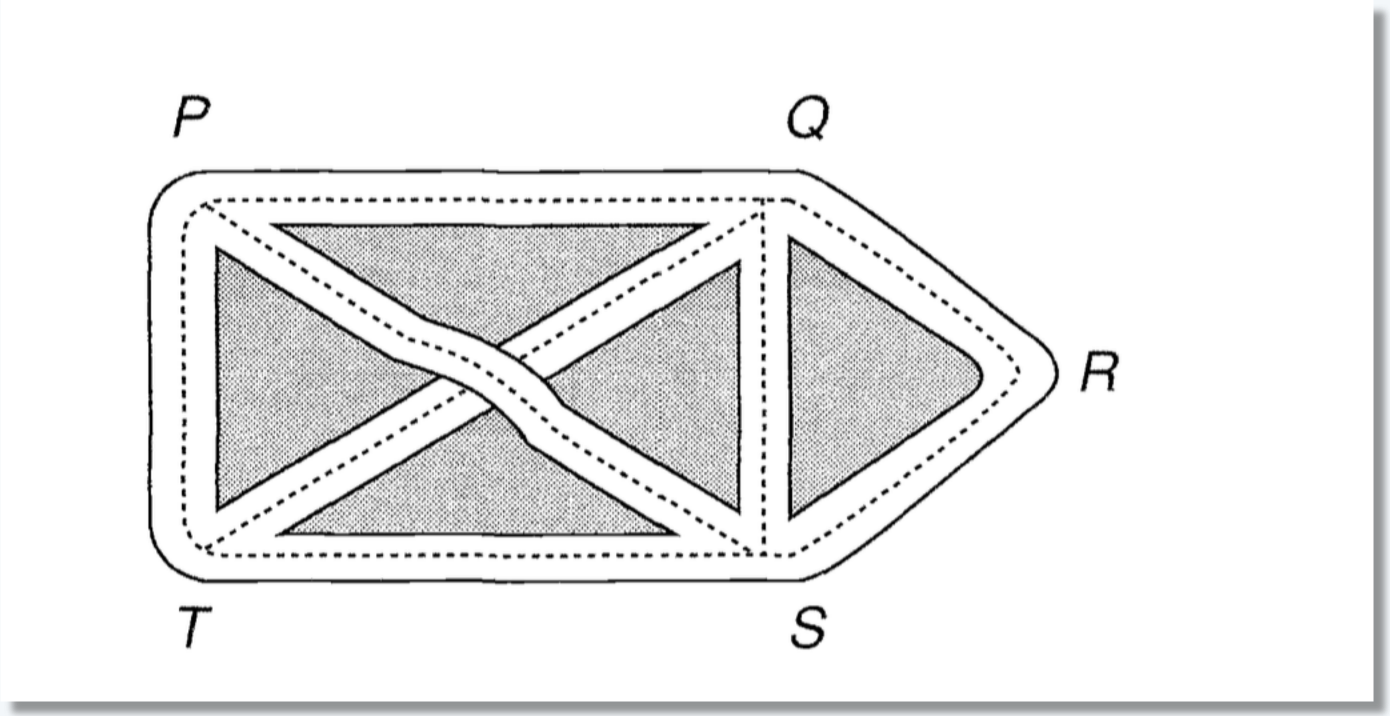
Metabolic Network of E. coli.

What is a graph?



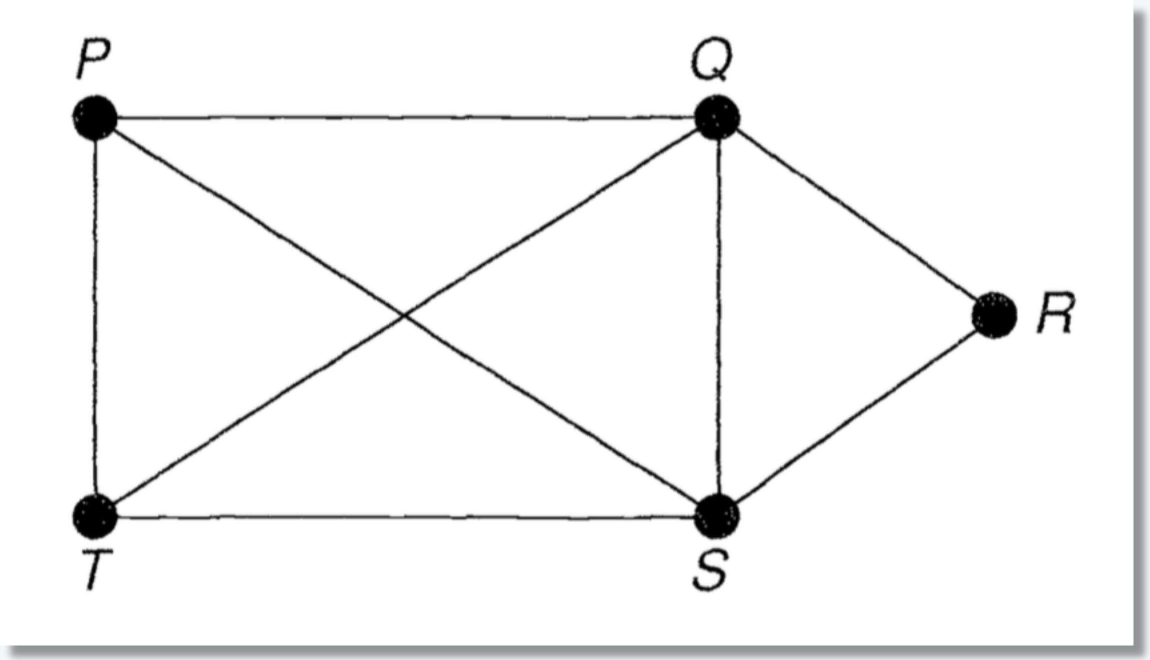
Vertices: P, Q, R, S, T
Edges: all the lines
Degree of a vertex: number of edges with that vertex as an end-point

Interpretation:



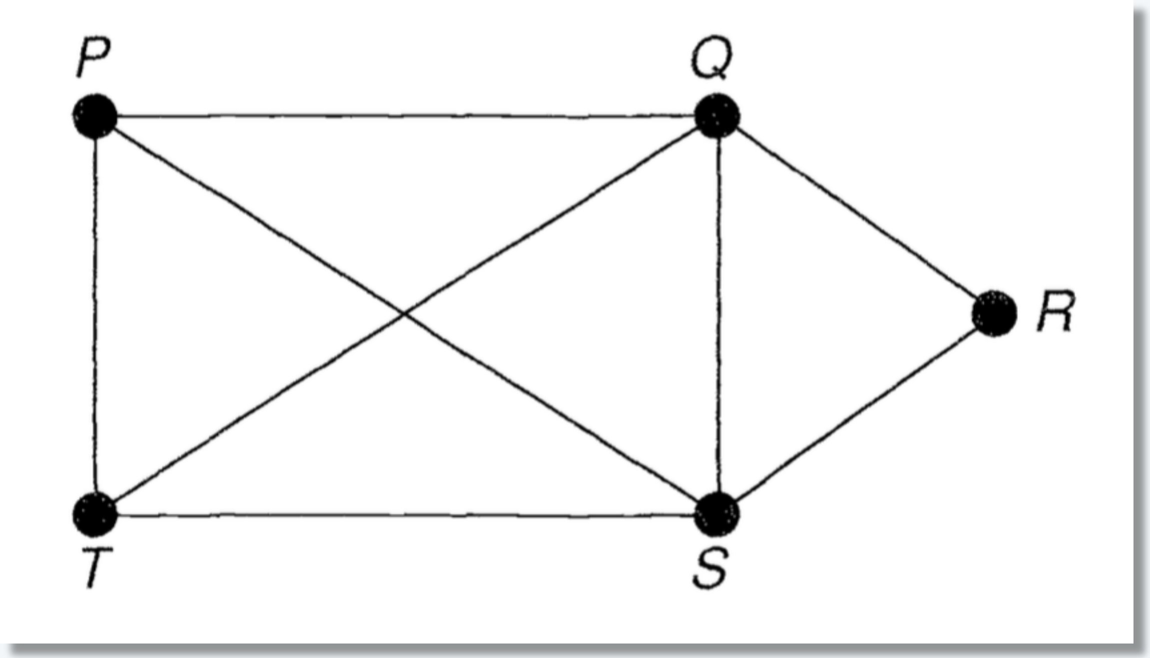
The graph from the last slide might depict this roadmap. Note that the intersection of the lines PS and QT is not a vertex, since it does not correspond to a cross-roads

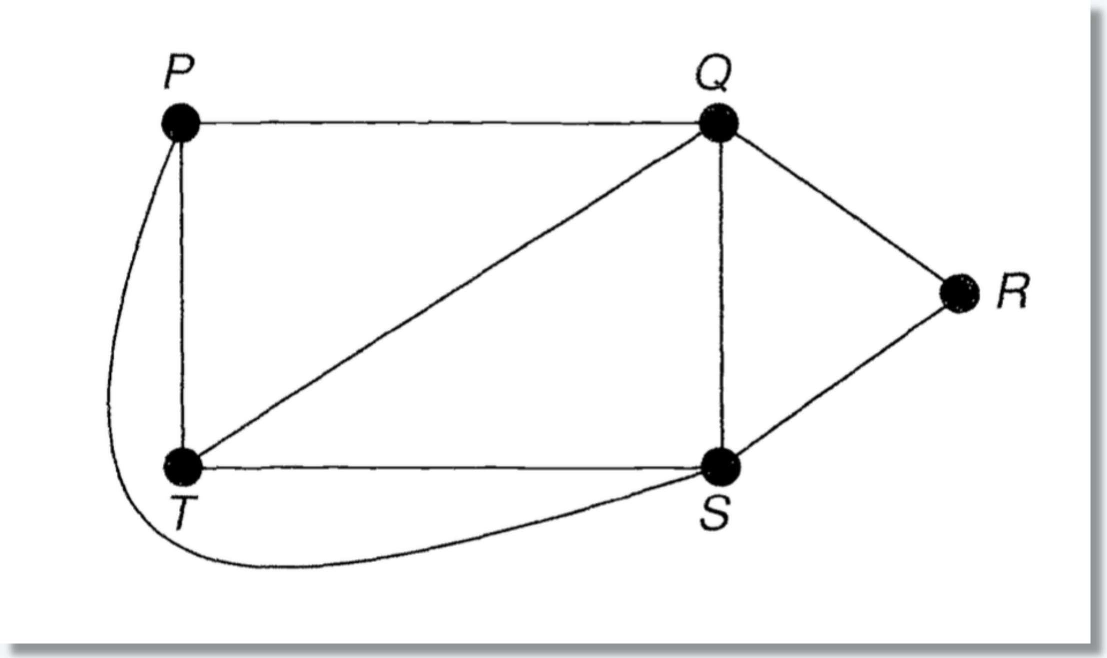
Another Interpretation:



If P, Q, R, S and T represent football teams, then the existence of an edge might correspond to the playing of a game between the teams at its end-points. Thus, team P has played against teams Q, S and T, but not against team R. In this representation, the degree of vertex is the number of games played by the corresponding team.

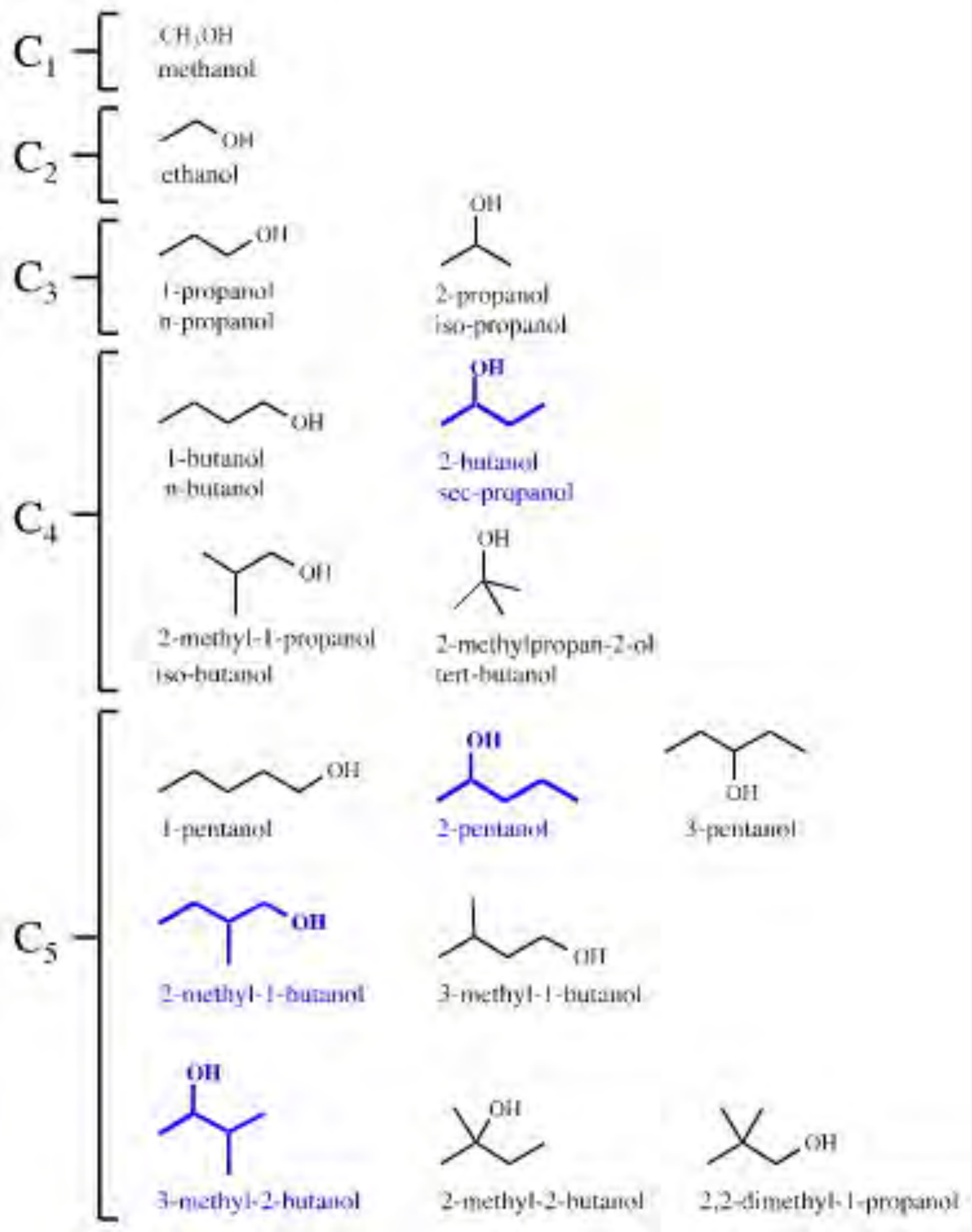
Two different graphs? No!





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 resulting graph still tells us whether there is a direct road from one intersection to another, and which football teams have played which. The only information we have lost concerns 'metrical' properties, such as the length of a road and the straightness of a wire.

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."

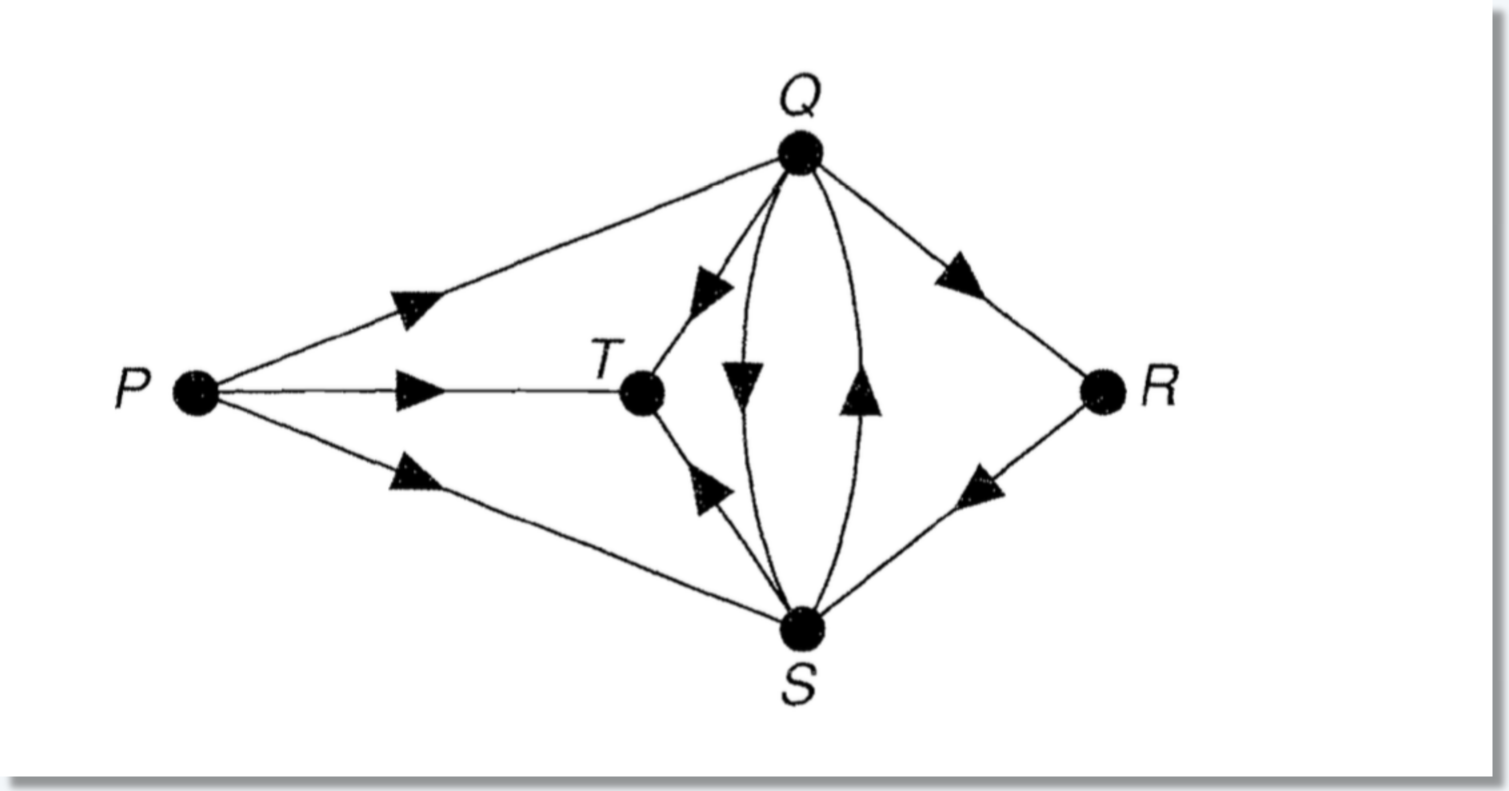
I have thus been led to see more clearly than ever I did before the existence of a common ground to the new mechanism, the new chemistry, and the new algebra. Underlying all these is the theory of pure colligation, which applies undistinguishably to the three great theories, all initiated within the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

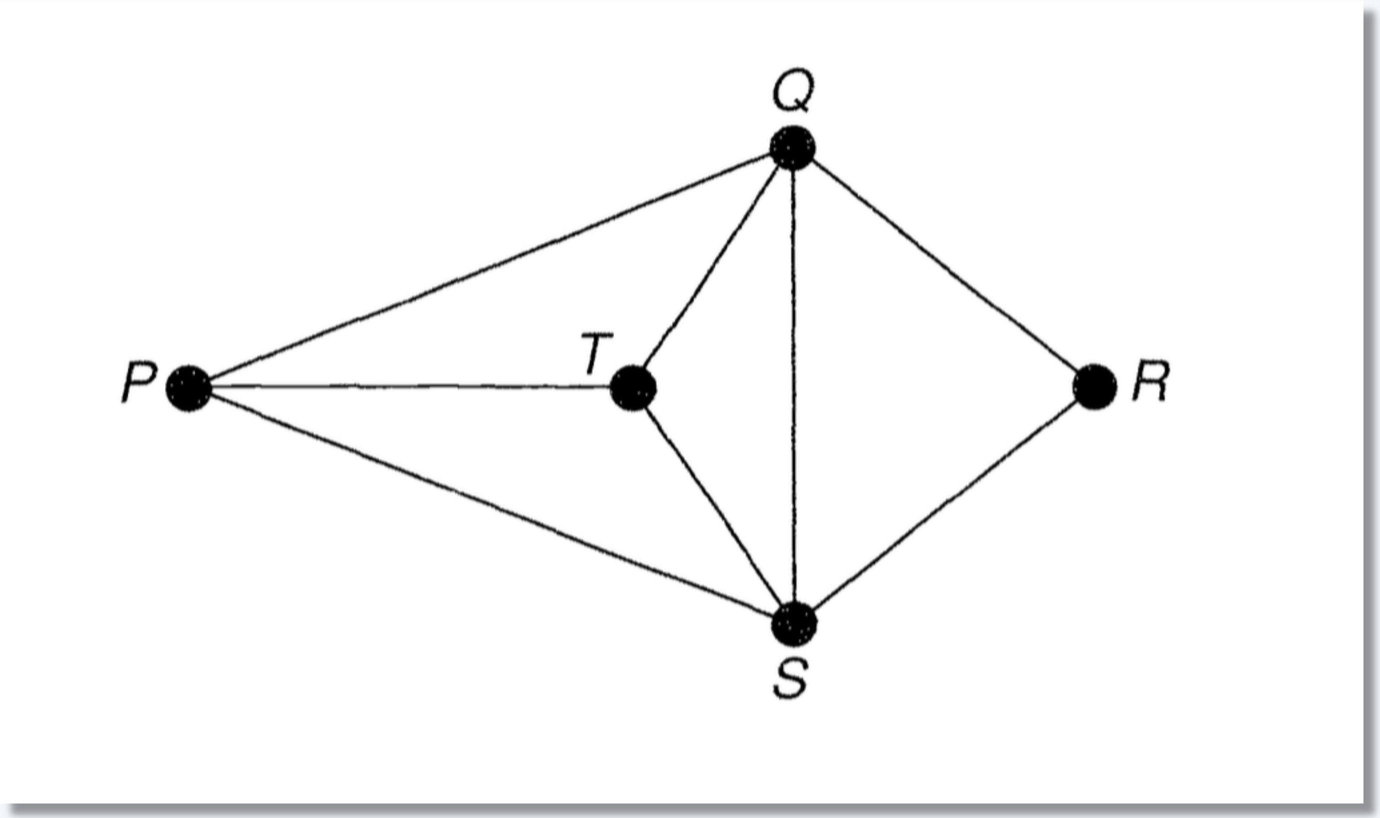
DR. PALMEN, of Helsingfors, has recently published an interesting memoir on the tracheal system of insects. He observes that although the gills of certain aquatic larvæ are attached to the skin very near to the points at which the spiracles open in the mature insects, and though spiracles and gills do 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)



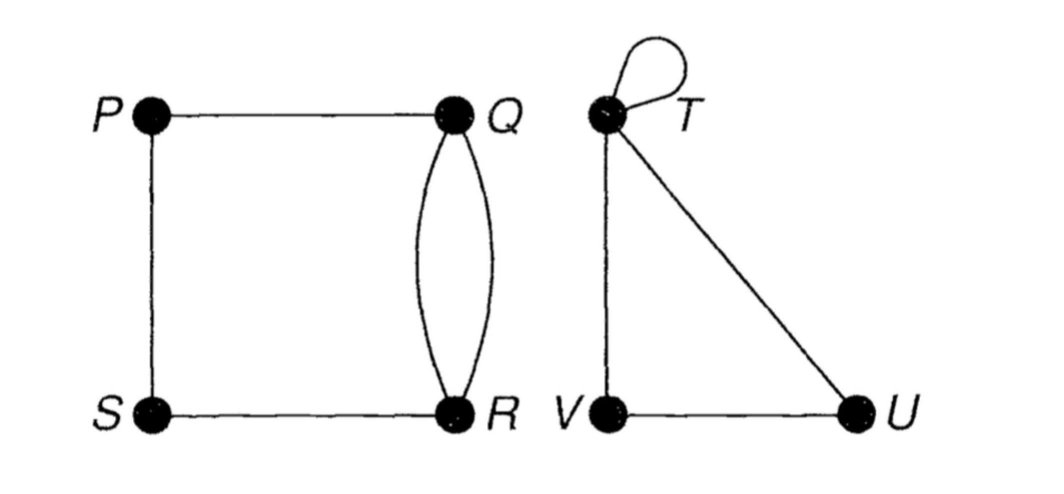
Assume again a graph depicts a roadmap. The study of **directed graphs** (or **digraphs,** as we abbreviate them) arises when making the roads into one-way streets. An example of a digraph is given above, the directions of the one-way streets being indicated by arrows. (In this example, there would be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

Walks, Paths, and Cycles



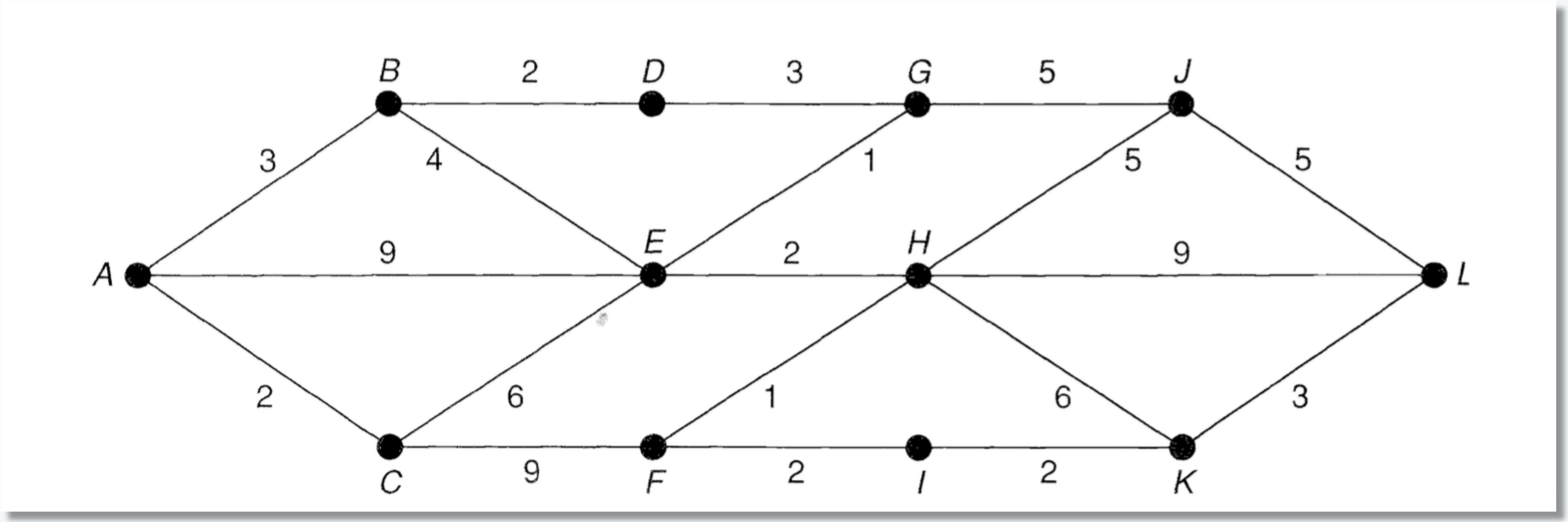
Much of graph theory involves 'walks' of various kinds. A **walk** is a 'way of getting from one vertex to another', and consists of a sequence of edges, one following after another. For example, in the above figure *P —> Q—>R* is a **walk of length 2**, and *P —> S —> Q —> T —> S —> R* is a walk of 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



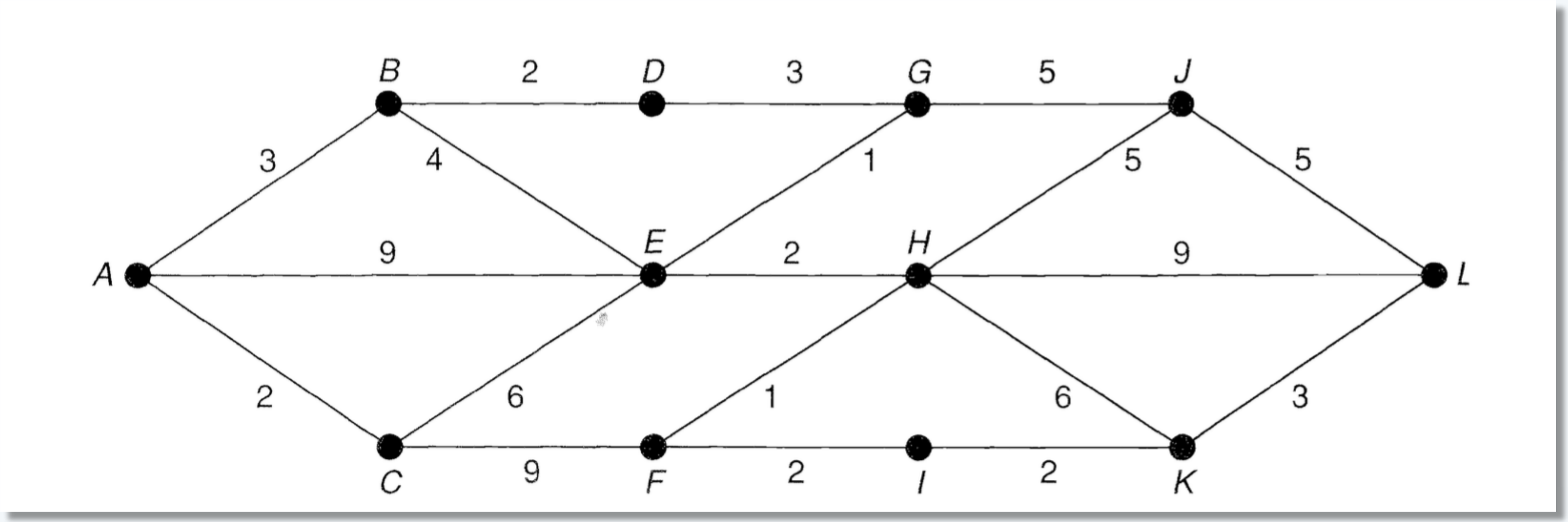
Some graphs are in two or more parts. For example, consider the graph whose vertices are the stations of the Copenhagen Metro and the New York Subway, and whose edges are the lines joining them. It is impossible to travel from Østerport to Grand Central Station using only edges of this graph, but if we confine our attention to the Copenhagen Metro only, then we can travel from any station to any other. 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*).
Example: Suppose that we have a 'map' of the form shown above, in which the letters A to L refer to towns that are connected by roads. Then the weights may denote the length of these roads.

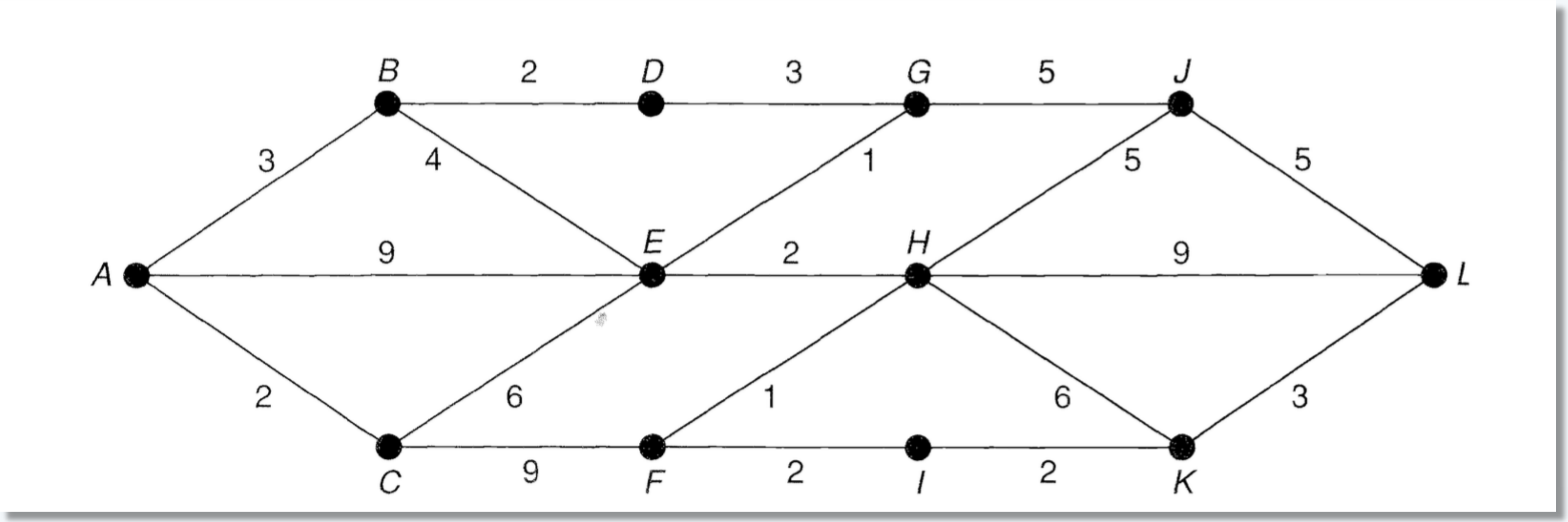
Shortest Path (between *one* pair of vertices)



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**?

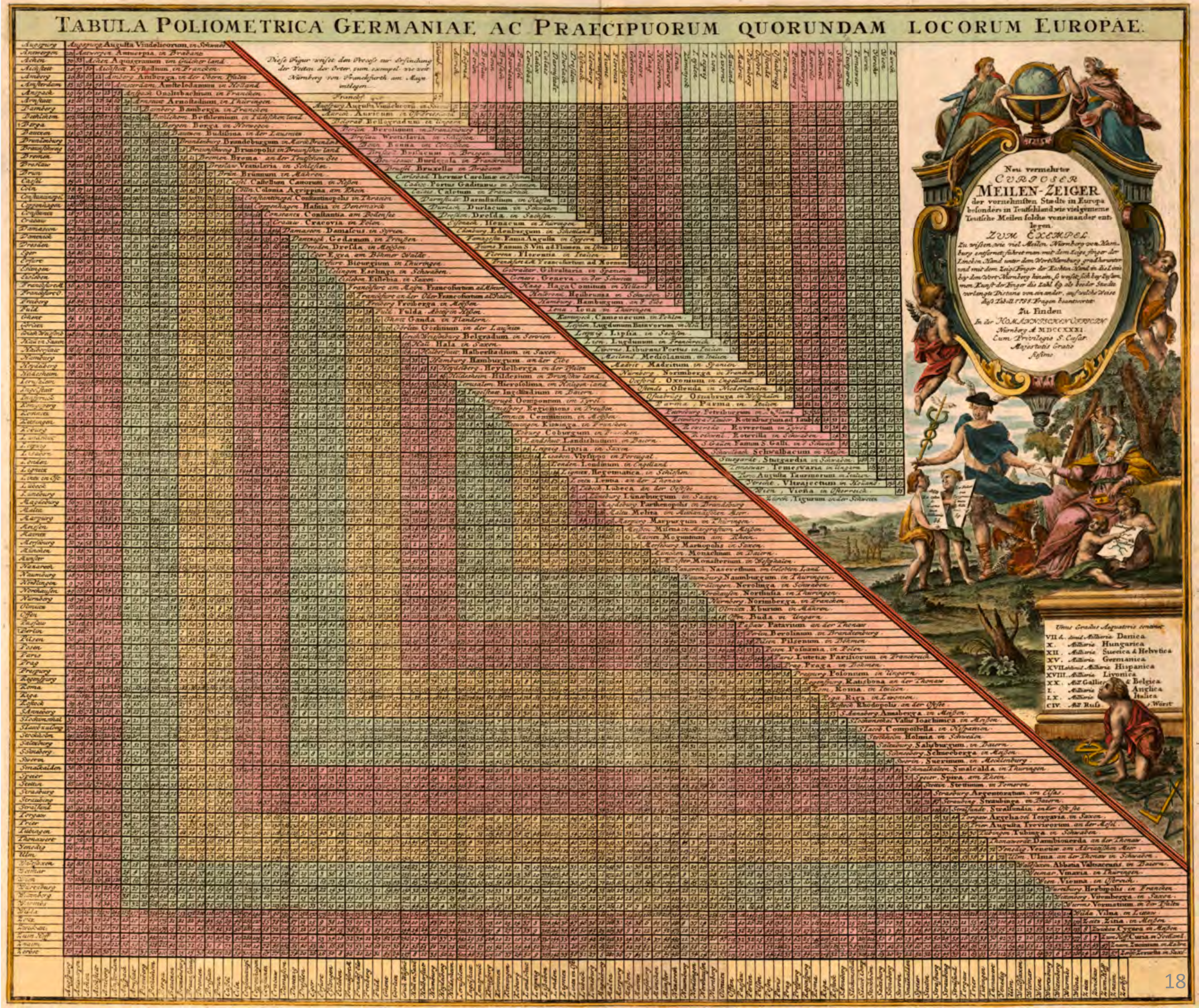
This problem is called the **All-Pairs Shortest Path Problem**

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

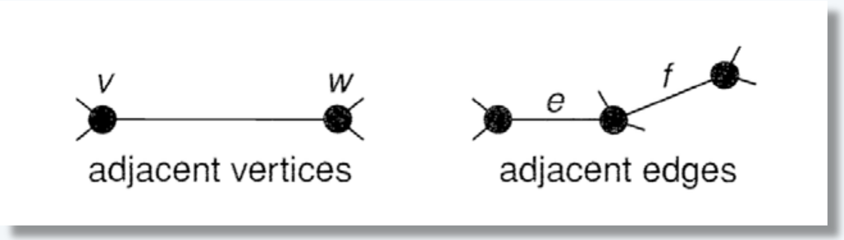
One of the most decorative tables of distances (in Roman miles) between major European cities printed in the eighteenth century. Not only were the data extremely useful for traveling but also for sending a letter, because distance, not weight, determined the price.

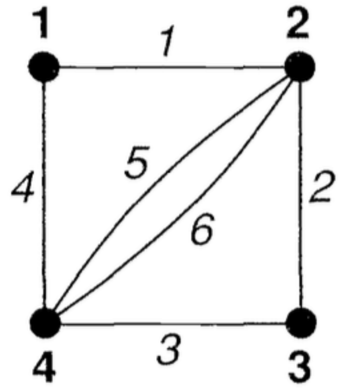


(From the “Historic Maps Collection”, Princeton University Library, link: [here](http://libweb5.princeton.edu/visual_materials/maps/websites/thematic-maps/introduction/introduction.html)

http://libweb5.princeton.edu/visual_materials/maps/websites/thematic-maps/introduction/introduction.html)

Matrix Representations for Graphs



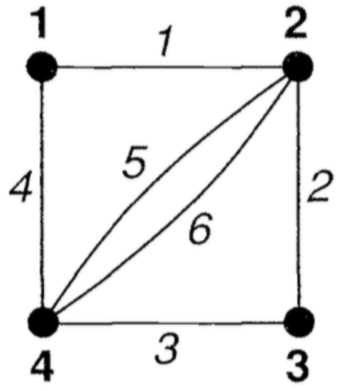


$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

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****.*

Matrix-Matrix Multiplication
Recap

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ -1 & 2 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 5 \end{pmatrix} = \begin{pmatrix} & & \\ & & \end{pmatrix}$$

Matrix-Matrix Multiplication
Recap

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ -1 & 2 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 5 \end{pmatrix} = \begin{pmatrix} \quad & \quad & \quad \end{pmatrix}$$

Matrix-Matrix Multiplication Recap

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ -1 & 2 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 5 \end{pmatrix} = \begin{pmatrix} 6 & 12 & 20 \\ 10 & 14 & 8 \end{pmatrix}$$

$$M \times N = R$$

$$r_{ij} = \sum_k m_{ik} * n_{kj}$$

Matrix-Matrix Multiplication
Recap

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ -1 & 2 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 5 \end{pmatrix} = \begin{pmatrix} 6 & 12 & 20 \\ 10 & 14 & 8 \end{pmatrix}$$

Zero-based Numbering (“Zero indexed”)

$$\begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \end{pmatrix}$$

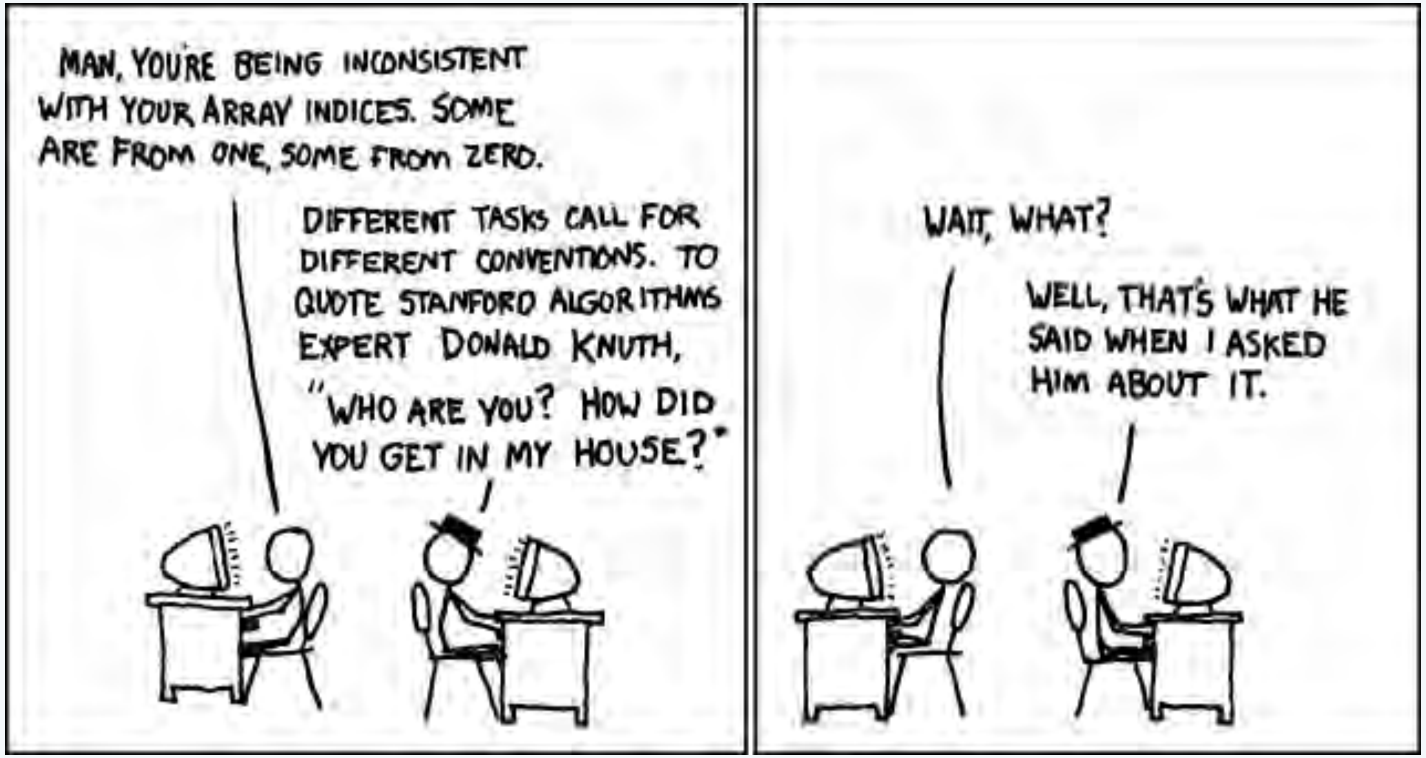
One-based Numbering (“One indexed”)

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{pmatrix}$$

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)

Matrix-Matrix Multiplication in Python (for Square Matrices)

```python
# Assume M and N are both square (size x size) matrices
def multSquareMatrices(M,N):
    size = len(M)
    result = [[0 for x in range(size)] for y in range(size)]

    for i in range(size):
        for j in range(size):
            for k in range(size):
                result[i][j] = result[i][j] + M[i][k] * N[k][j]

    return result
```

Provided Code: `matMult.py`

Number of additions per `result[i][j]` entry:	`size`
Number of multiplications per `result[i][j]` entry:	`size`
Number of entries in the `result` matrix:	`size x size`
Overall number of operations (additions and multiplications):	`2 x size x ( size x size )`
Overall computational runtime:	$\mathcal{O}(\text{size}^3)$

Matrix-Matrix Multiplication in Python

```
# Assume two matrices M and N, not necessarily squared
# (not needed further on in the lecture)
def multGeneral(M,N):

    result = [[0 for x in range(len(N[0]))] for y in range(len(M))]

    for i in range(len(M)):
        for j in range(len(N[0])):
            for k in range(len(N)):
                result[i][j] = result[i][j] + M[i][k] * N[k][j]

    return result
```

Provided Code: `matMult.py`

Comments to Python Code

- Creating a list of three 0's :

```
In [8]: [0 for i in range(3)]
Out[8]: [0, 0, 0]
```

- Creating a list of two lists with three 0's (i.e., a "matrix" of size 2 x 3):

```
In [8]: [0 for i in range(3)]
Out[8]: [0, 0, 0]

In [9]: [[0 for i in range(3)] for j in range(2)]
Out[9]: [[0, 0, 0], [0, 0, 0]]
```

Matrices in Python: Implemented as Lists of Lists:

```
In [1]: M
Out[1]: [[1, 0, 2, 3], [-1, 2, 2, 1]]

In [2]: N
Out[2]: [[1, 2, 3], [4, 5, 2], [1, 2, 1], [1, 2, 5]]

In [3]: S
Out[3]: [[1, 2, 0], [2, 0, 1], [-1, 2, 3]]

In [4]: multSquareMatrices(S,S)
Out[4]: [[5, 2, 2], [1, 6, 3], [0, 4, 11]]

In [5]:
```

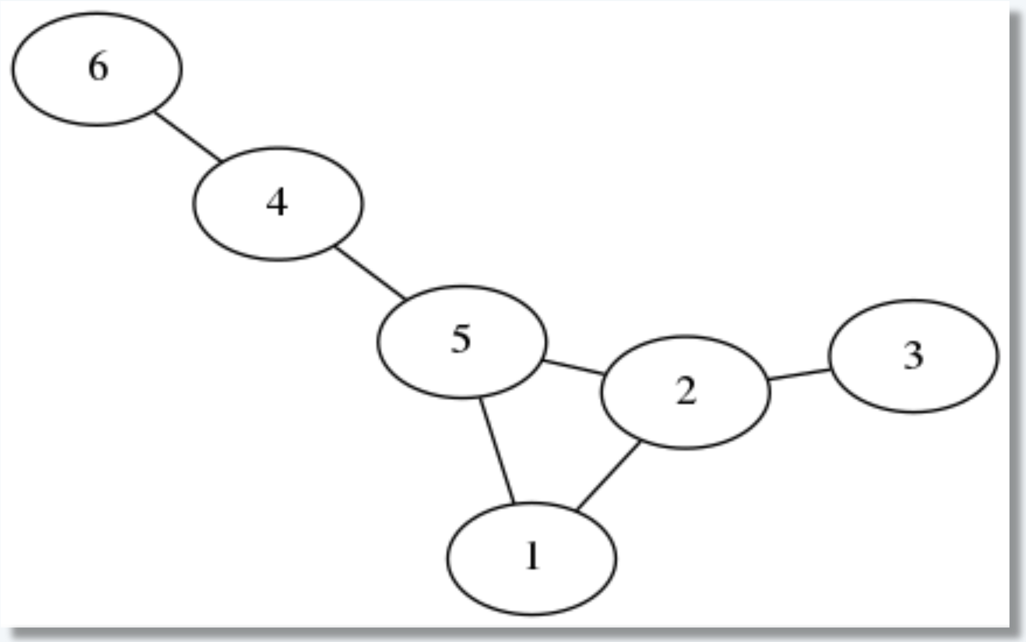
```
M = [[ 1, 0, 2, 3],
     [-1, 2, 2, 1]]

N = [[ 1, 2, 3],
     [ 4, 5, 2],
     [ 1, 2, 1],
     [ 1, 2, 5]]

S = [[ 1, 2, 0],
     [ 2, 0, 1],
     [-1, 2, 3]]

print("Initial Matrix M:\n")
printMatrix(M)

print("Initial Matrix N:\n")
printMatrix(N)

print("M x N:\n")
printMatrix( multGeneral(M,N) )
```

Provided Code: `matMult.py`

“Matrix” dimensions:

`M` has `len(M)` many rows and `len(M[0])` many columns
`N` has `len(N)` many rows and `len(N[0])` many columns
The result needs to have `len(M)` many rows and `len(N[0])` many columns

Powers of the Adjacency Matrix



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

$$A^k = \underbrace{A \times A \ldots \times A}_{k \text{ times}} \text{ is called the k-th power of the adjacency matrix}$$

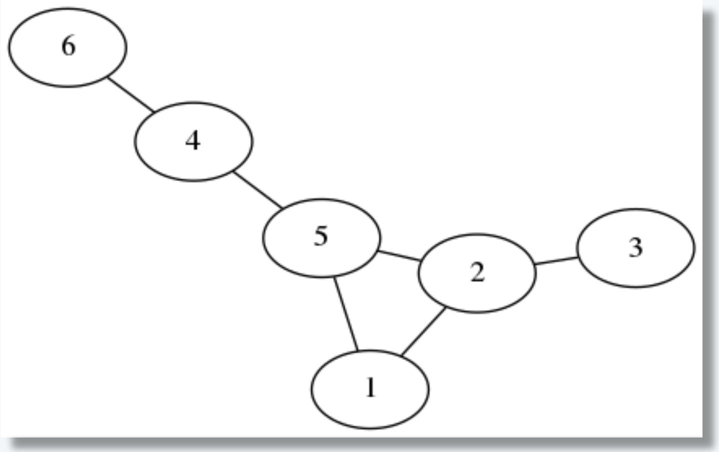
Theorem:
If G is a graph with adjacency matrix A, and vertices with indices $1, \ldots, n$ then for each positive integer k

the ij-th entry of A^k

is

the number of different walks using exactly k edges from node i to node j

$$A^2 = \begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 0 \\ 1 & 3 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \quad A^3 = \begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 2 & 4 & 1 & 1 & 4 & 1 \\ 4 & 2 & 3 & 1 & 5 & 1 \\ 1 & 3 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 4 & 2 \\ 4 & 5 & 1 & 4 & 2 & 0 \\ 1 & 1 & 0 & 2 & 0 & 0 \end{pmatrix} \end{array} \quad A^4 = \begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 8 & 7 & 4 & 5 & 7 & 1 \\ 7 & 12 & 2 & 6 & 7 & 1 \\ 4 & 2 & 3 & 1 & 5 & 1 \\ 5 & 6 & 1 & 6 & 2 & 0 \\ 7 & 7 & 5 & 2 & 13 & 4 \\ 1 & 1 & 1 & 0 & 4 & 2 \end{pmatrix} \end{array}$$



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

Example :

Consider the two vertices with index 4 and 5 in A^4

Length 4 walks:

1) 4 -> 5 -> 1 -> 2 -> 5
2) 4 -> 5 -> 2 -> 1 -> 5

There are 2 walks of length 4.

Furthermore, A^4_{45}=2.

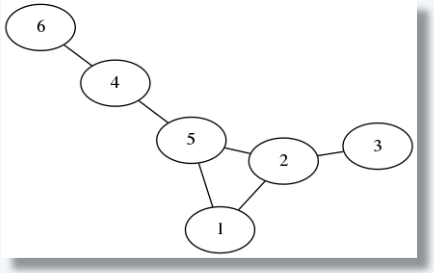
$$A^2 = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 2 & 1 & 1 & 1 & 1 & 0 \\ 2 & 1 & 3 & 0 & 1 & \boxed{1} & 0 \\ 3 & 1 & 0 & 1 & 0 & 1 & 0 \\ 4 & 1 & 1 & 0 & 2 & 0 & 0 \\ 5 & 1 & 1 & 1 & 0 & 3 & 1 \\ 6 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}$$

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

$$A^4 = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 8 & 7 & 4 & 5 & 7 & 1 \\ 2 & 7 & 12 & 2 & 6 & 7 & 1 \\ 3 & 4 & 2 & 3 & 1 & 5 & 1 \\ 4 & 5 & 6 & 1 & 6 & \boxed{2} & 0 \\ 5 & 7 & 7 & 5 & 2 & 13 & 4 \\ 6 & 1 & 1 & 1 & 0 & 4 & 2 \end{array}$$

In Python3

```python
def printMatrix(M):
    for row in M:
        print(["%3.0f" % a for a in row])
    print("\n")


A = [[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]]

# make a copy of X
R = deepcopy(A)

print("Initial Matrix:\n")
printMatrix(A)

for i in range(2,5):
    print("%d-th power of A : \n"%i)
    R = multSquareMatrices(R,A)
    printMatrix(R)
```



Provided Code: `adjacencyMatMult.py`

```
Eule:IntroCS2017 daniel$ ipython adjacencyMatMult.py
Initial Matrix:

['  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']


2-th power of A :

['  2', '  1', '  1', '  1', '  1', '  0']
['  1', '  3', '  0', '  1', '  1', '  0']
['  1', '  0', '  1', '  0', '  1', '  0']
['  1', '  1', '  0', '  2', '  0', '  0']
['  1', '  1', '  1', '  0', '  3', '  1']
['  0', '  0', '  0', '  0', '  1', '  1']


3-th power of A :

['  2', '  4', '  1', '  1', '  4', '  1']
['  4', '  2', '  3', '  1', '  5', '  1']
['  1', '  3', '  0', '  1', '  1', '  0']
['  1', '  1', '  1', '  0', '  4', '  2']
['  4', '  5', '  1', '  4', '  2', '  0']
['  1', '  1', '  0', '  2', '  0', '  0']


4-th power of A :

['  8', '  7', '  4', '  5', '  7', '  1']
['  7', ' 12', '  2', '  6', '  7', '  1']
['  4', '  2', '  3', '  1', '  5', '  1']
['  5', '  6', '  1', '  6', '  2', '  0']
['  7', '  7', '  5', '  2', ' 13', '  4']
['  1', '  1', '  1', '  0', '  4', '  2']


Eule:IntroCS2017 daniel$
```

Proof: (also on blackboard)

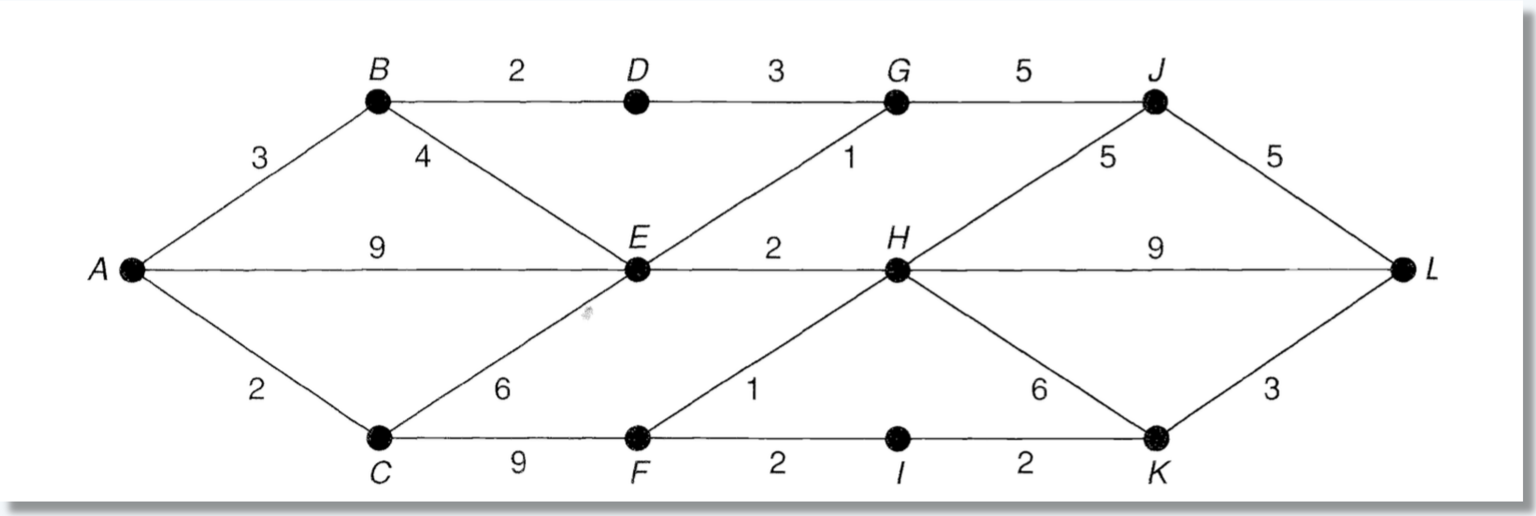
Let G be a graph with adjacency matrix A, and vertices $1, \ldots, n$. We proceed by induction on k to obtain the result.

Base Case:
Let $k = 1$. $A^1 = A$. a_{ij} is the number of edges from i to j, which is identical to the number of walks of length 1 from i to j.

Inductive Step:
Assume true for a positive integer k. Let b_{ij} be the ij-th entry of A^k, and let a_{ij} be the ij-th entry of A. By the inductive hypothesis b_{ij} is the number of walks of length k from i to j. Consider the ij-th entry of $A^{k+1} = A \times A^k$, i.e, $A_{ij}^{k+1} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$. Consider $a_{i1}b_{1j}$. This is equal to the number of walks of length 1 from i to 1 *times* the number of walks of length k from 1 to j. This is therefor equal to the number of walks of length $k+1$ from i to j, where 1 is the second vertex. This argument holds for each vertex m, i.e., $a_{im}b_{mj}$ is the number of walks from i to j in which m is the second vertex. Therefore, the sum is the number of all possible walks from i to j.

Algorithm for All-Pairs Shortest Path

Weighted Graph G with weights on edges:



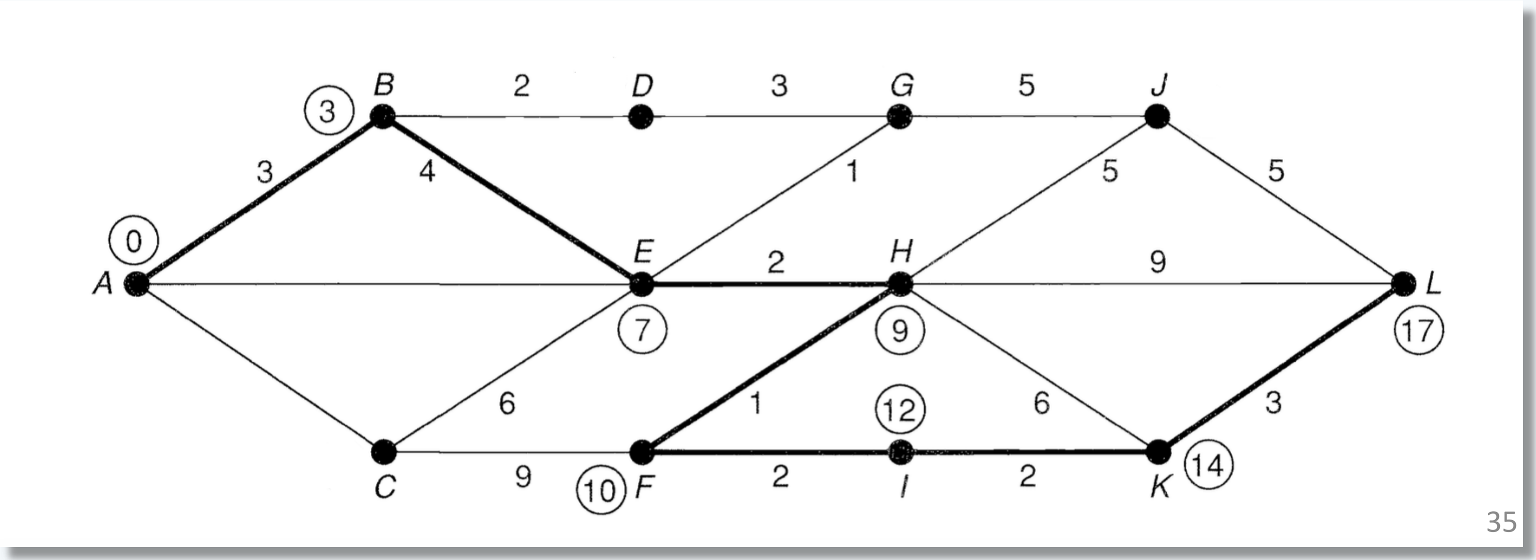
- What is the distance (=length of **the** shortest path) between A and L ?

17

<u>Generalization:</u>

- What are the distances of ALL paths (=lenghts of ALL shortest paths) between all pairs of nodes?

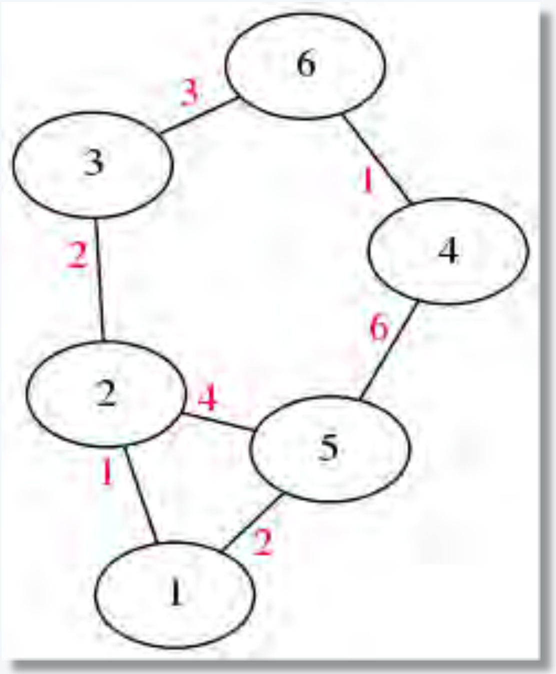
  … and how can we find all these distances?



The Edge Weight Matrix W

Example:

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



weights are depicted in red

Definition:

$$W_{ij} = \begin{cases} \text{the weight of the edge } (i,j) & \text{if the edge } (i,j) \text{ exists} \\ 0 & \text{if } i = j \\ \infty & \text{else} \end{cases}$$

Interpretation:

W_{ij} is the distance from vertex i to vertex j using maximally 1 edge

Note: Matrix W has entries corresponding to infinity, as it might be impossible to reach vertex j from vertex i via **1** edge.

We assume all weights are not negative, i.e., larger or equal to 0.

A modified Matrix-Matrix Multiplication

$$\begin{pmatrix} 1 & 0 & 2 \\ 1 & 2 & 4 \\ 3 & 1 & 2 \end{pmatrix} \odot \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 1 & 2 & 5 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 2 \\ 2 & 3 & 4 \\ 3 & 4 & 3 \end{pmatrix}$$

$$M \odot N = R$$

<u>Definition:</u>

$r_{ij} = \min_k \{m_{ik} + n_{kj}\}$

<u>Example:</u>

$r_{33} = \min\{3 + 3, 1 + 2, 2 + 5\} = 3$

Note: this operation is very similar to the standard matrix-matrix multiplication: however, for computation of the ij-th entry the multiplication is replaced by addition, and addition is replaced by the minimum operation.

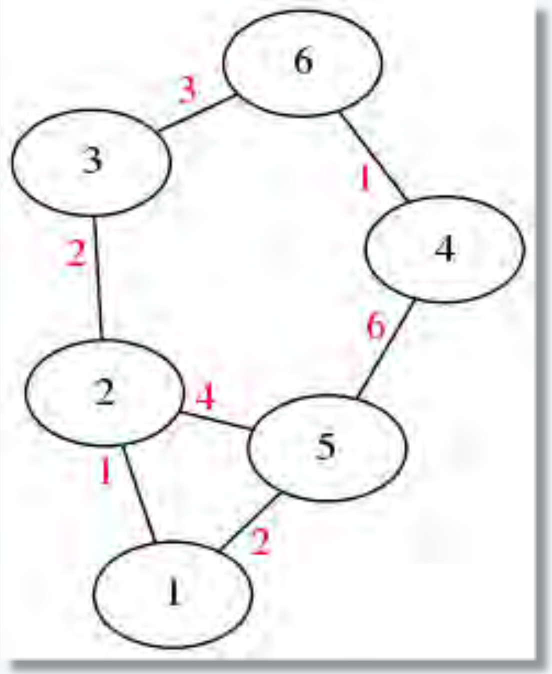
Theorem:

If G is a weighted graph with edge weight matrix W, and vertices with indices $1, \dots, n$ then for each positive integer k

$$\text{the } ij\text{-th entry of } W^k = \underbrace{W \odot W \odot \ldots \odot W}_{k \text{ times}}$$

is

the length of the shortest path from i to j using maximally k edges



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

<u>Examples :</u>

Consider the two vertices with index 4 and 1 in W^4

Shortest Path using maximally 4 edges:

4 -> 6 -> 3 -> 2 -> 1 (distance 7)

Consider the two vertices with index 5 and 3 in W^4

Shortest Path using maximally 4 edges:

5 -> 1 -> 2 -> 3 (distance 5)

$$W^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 3 & 8 & 2 & \infty \\ 1 & 0 & 2 & 10 & 3 & 5 \\ 3 & 2 & 0 & 4 & \boxed{6} & 3 \\ 8 & 10 & 4 & 0 & 6 & 1 \\ 2 & 3 & 6 & 6 & 0 & 7 \\ \infty & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array}$$

$$W^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 3 & 8 & 2 & 6 \\ 1 & 0 & 2 & 6 & 3 & 5 \\ 3 & 2 & 0 & 4 & 5 & 3 \\ 8 & 6 & 4 & 0 & 6 & 1 \\ 2 & 3 & 5 & 6 & 0 & 7 \\ 6 & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array}$$

$$W^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 3 & 7 & 2 & 6 \\ 1 & 0 & 2 & 6 & 3 & 5 \\ 3 & 2 & 0 & 4 & 5 & 3 \\ \boxed{7} & 6 & 4 & 0 & 6 & 1 \\ 2 & 3 & \boxed{5} & 6 & 0 & 7 \\ 6 & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array}$$

Matrix-Matrix Multiplication in Python (for Square Matrices)

```
# Assume M and N are both square (size x size) matrices
def multSquareMatrices(M,N):
    size = len(M)
    result = [[0 for x in range(size)] for y in range(size)]

    for i in range(size):
        for j in range(size):
            for k in range(size):
                result[i][j] = result[i][j] + M[i][k] * N[k][j]

    return result
```

Modified Matrix-Matrix Multiplication in Python (for Square Matrices)

```
def multModSquareMatrices(M,N):
    size = len(M)
    result = [[inf for x in range(size)] for y in range(size)]

    for i in range(size):
        for j in range(size):
            for k in range(size):
                result[i][j] = min(result[i][j], M[i][k] + N[k][j])

    return result
```

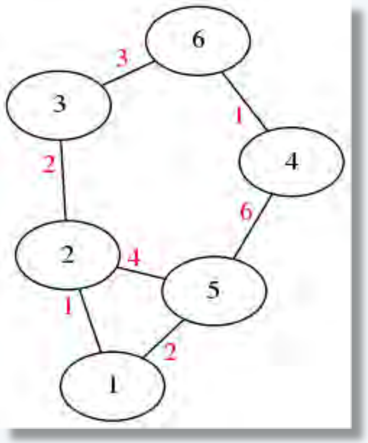
Standard Matrix-Matrix Multiplication:

```
# Assume M and N are both square (size x size) matrices
def multSquareMatrices(M,N):
    size = len(M)
    result = [[0 for x in range(size)] for y in range(size)]

    for i in range(size):
        for j in range(size):
            for k in range(size):
                result[i][j] = result[i][j] + M[i][k] * N[k][j]

    return result
```

Provided Code: `shortestPaths.py`

In Python3

```
W = [[   0,   1, inf, inf,   2, inf],
     [   1,   0,   2, inf,   4, inf],
     [ inf,   2,   0, inf, inf,   3],
     [ inf, inf, inf,   0,   6,   1],
     [   2,   4, inf,   6,   0, inf],
     [ inf, inf,   3,   1, inf,   0]]

# make a copy of X
R = deepcopy(W)

print("Initial Matrix:\n")
printMatrix(W)

for i in range(2,5):
    print("%d-th power of W : \n"%i)
    R = multModSquareMatrices(R,W)
    printMatrix(R)
```




```
Eule:IntroCS2017 daniel$ python3 shortestPaths.py
Initial Matrix:

['  0', '  1', 'inf', 'inf', '  2', 'inf']
['  1', '  0', '  2', 'inf', '  4', 'inf']
['inf', '  2', '  0', 'inf', 'inf', '  3']
['inf', 'inf', 'inf', '  0', '  6', '  1']
['  2', '  4', 'inf', '  6', '  0', 'inf']
['inf', 'inf', '  3', '  1', 'inf', '  0']


2-th power of W :

['  0', '  1', '  3', '  8', '  2', 'inf']
['  1', '  0', '  2', ' 10', '  3', '  5']
['  3', '  2', '  0', '  4', '  6', '  3']
['  8', ' 10', '  4', '  0', '  6', '  1']
['  2', '  3', '  6', '  6', '  0', '  7']
['inf', '  5', '  3', '  1', '  7', '  0']


3-th power of W :

['  0', '  1', '  3', '  8', '  2', '  6']
['  1', '  0', '  2', '  6', '  3', '  5']
['  3', '  2', '  0', '  4', '  5', '  3']
['  8', '  6', '  4', '  0', '  6', '  1']
['  2', '  3', '  5', '  6', '  0', '  7']
['  6', '  5', '  3', '  1', '  7', '  0']


4-th power of W :

['  0', '  1', '  3', '  7', '  2', '  6']
['  1', '  0', '  2', '  6', '  3', '  5']
['  3', '  2', '  0', '  4', '  5', '  3']
['  7', '  6', '  4', '  0', '  6', '  1']
['  2', '  3', '  5', '  6', '  0', '  7']
['  6', '  5', '  3', '  1', '  7', '  0']
```

Provided Code: `shortestPaths.py`
Note: python3 required because of `inf`

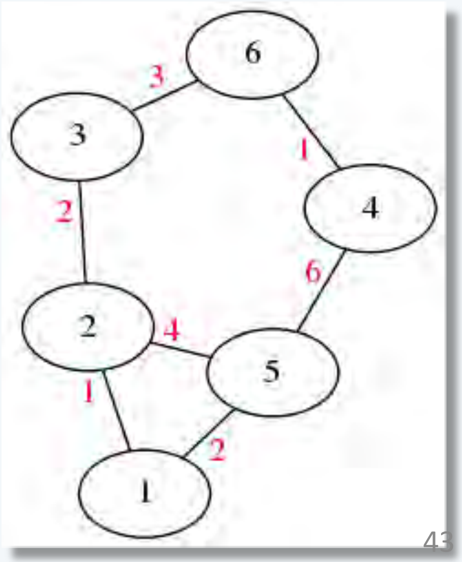
$$W = \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & \infty & \infty & 2 & \infty \\ 1 & 0 & 2 & \infty & 4 & \infty \\ \infty & 2 & 0 & \infty & \infty & 3 \\ \infty & \infty & \infty & 0 & 6 & 1 \\ 2 & 4 & \infty & 6 & 0 & \infty \\ \infty & \infty & 3 & 1 & \infty & 0 \end{pmatrix} \end{array} \quad W^2 = \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 3 & 8 & 2 & \infty \\ 1 & 0 & 2 & 10 & 3 & 5 \\ 3 & 2 & 0 & 4 & 6 & 3 \\ 8 & 10 & 4 & 0 & 6 & 1 \\ 2 & 3 & 6 & 6 & 0 & 7 \\ \infty & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array}$$

$$W^3 = \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 3 & 8 & 2 & 6 \\ 1 & 0 & 2 & 6 & 3 & 5 \\ 3 & 2 & 0 & 4 & 5 & 3 \\ 8 & 6 & 4 & 0 & 6 & 1 \\ 2 & 3 & 5 & 6 & 0 & 7 \\ 6 & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array} \quad W^4 = \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 3 & 7 & 2 & 6 \\ 1 & 0 & 2 & 6 & 3 & 5 \\ 3 & 2 & 0 & 4 & 5 & 3 \\ 7 & 6 & 4 & 0 & 6 & 1 \\ 2 & 3 & 5 & 6 & 0 & 7 \\ 6 & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array} \quad W^5 = \begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 3 & 7 & 2 & 6 \\ 1 & 0 & 2 & 6 & 3 & 5 \\ 3 & 2 & 0 & 4 & 5 & 3 \\ 7 & 6 & 4 & 0 & 6 & 1 \\ 2 & 3 & 5 & 6 & 0 & 7 \\ 6 & 5 & 3 & 1 & 7 & 0 \end{pmatrix} \end{array}$$

$$W \neq W^2 \neq W^3 \neq W^4 = W^5 = W^6 = \ldots$$

Which value of k is necessary, in order to have W^k contain all the pairwise distances of all vertexes?

Answer: $n-1$ (which is identical to $|V|-1$)



Assume all edge weights are not negative. The number of edges needed for a shortest path can maximally be n-1, where n is the number of vertices in the graph. If the path would go via n edges, then you would have to visit at least one vertex twice, but then the path cannot be a shortest path anymore. Obviously $W^k = W^{n-1}$ for all k>n-1.

Lemma:

If G is a weighted graph with edge weight matrix W, and vertices with indices $1, \ldots, n$ then

$$\text{the } ij\text{-th entry of } W^{n-1} = \underbrace{W \odot W \odot \ldots \odot W}_{n-1 \text{ times}}$$

is

the distance from i to j

$D := W^{n-1}$ is called the distance matrix of the graph G.

Computation of the Distance Matrix by Repeated Squaring

$$W^{n-1} = \underbrace{\left(\underbrace{\left(\underbrace{\left(\underbrace{\left(\underbrace{(W \odot W)}_{W^2} \odot W\right)}_{W^3} \odot W\right)}_{W^4} \odot W\right)}_{W^5} \odot \dots \odot W\right)}_{W^{n-1}}$$

n-2 matrix-matrix multiplication are needed in order to compute the distance matrix $D = W^{n-1}$

$$W^{(2^k)} = \underbrace{\left(\left(\underbrace{\left(\underbrace{\left(\underbrace{(W \odot W)}_{W^2}\right)^2}_{W^4}\right)^2}_{W^8}\right)^2 \dots\right)^2}_{W^{(2^k)}}$$

k matrix-matrix multiplication are needed (namely squaring a matrix k times) in order to compute the matrix $W^{(2^k)}$

2^k has to be larger or equal to n-1, or equivalently, k has to be larger or equal to $\log_2(n-1)$

<u>Example:</u> Consider a graph G with 101 vertices. In order to compute the distance matrix $\mathrm{D} = W^{100}$, the left approach needs to make 99 matrix-matrix multiplications. The right approach (called repeated squaring) requires only 7 matrix-matrix multiplications, as $2^7 = 128$, and $\mathrm{D} = W^{128} = W^{100}$

Test in Python3

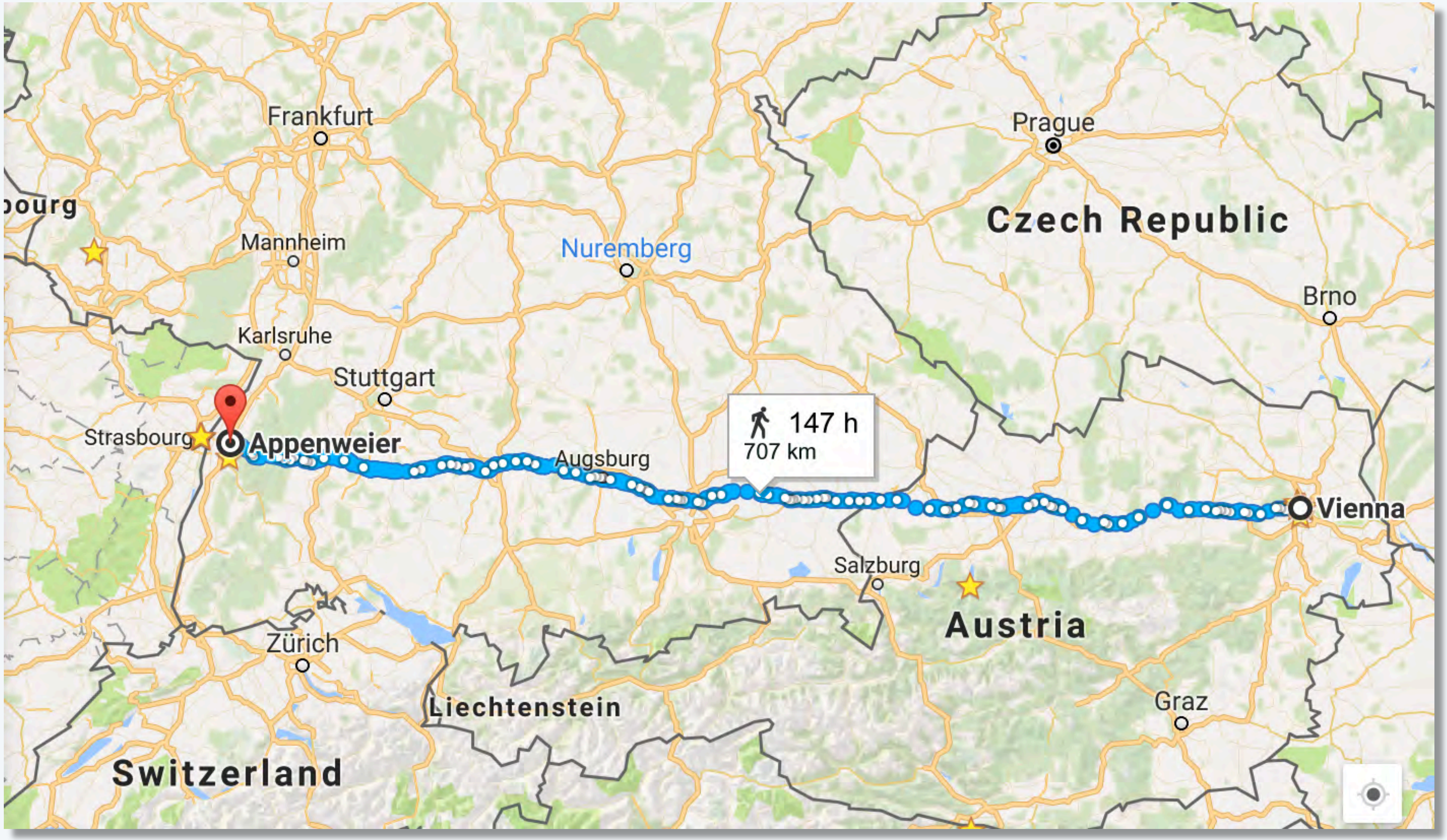
```
# make a random edge weight matrix of size x size
size=100
W = [[0 for x in range(size)] for y in range(size)]
for i in range(size):
    for j in range(i,size):
        r = randint(0,10)
        W[i][j] = r
        W[j][i] = r

# make a copy of the edge weight matrix W
R = deepcopy(W)

print("Comparing runtimes for distance matrix computation for matrices of size %d x %d"%(size,size))

# find the distance matrix by (n-2) subsequent matrix matrix multiplications
# R = (((W*W)*W)*...*W) = W^(n-1)
t1 = time()
for i in range(0,size-2):
    R = multModSquareMatrices(R,W)
print("The n-2 multiplications for computing D took %3.2f seconds"%(time()-t1))

# set the R=W (re-initialize)
R = deepcopy(W)

# find the distance matrix by ceil(log_2(n-1)) subsequent matrix matrix mulitplications via repeated squaring
# R = (((W^2)^2)^2...)^2
t1 = time()
for i in range(0,ceil(log2(size-1))):
    R = multModSquareMatrices(R,R)
print("The ceil(log_2(n-1)) multiplications took %3.2f seconds"%(time()-t1))
```

Note: `ceil(log2(size-1))` returns the smallest integer larger or equal to `log2(size-1)`, i.e., `R` will be the distance matrix after this `for` loop.

```
Eule:IntroCS2017 daniel$ python3 timing.py
Comparing runtimes for distance matrix computation for matrices of size 100 x 100
The n-2 multiplications for computing D took 40.27 seconds
The ceil(log_2(n-1)) multiplications took 2.93 seconds
Eule:IntroCS2017 daniel$
```

Provided Code: `timing.py`

The most obvious Application of Computing the Distance Matrix:



Another Application of the Distance Matrix: Predicting Boiling Points of Paraffins

In 1947 Harry Wiener defined the **Wiener-Index** of a graph G in order to predict the boiling point of different paraffins. He used the graph representation G of the carbon backbone of a molecule with *n* carbon atoms and calculated the Wiener-Index the sum of all distances between all pairs of vertexes, i.e.

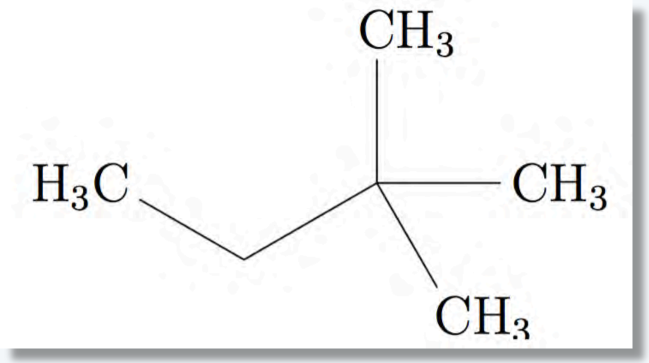
$$\mathcal{W}(G) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij}$$

He predicted the boiling point t_B to be

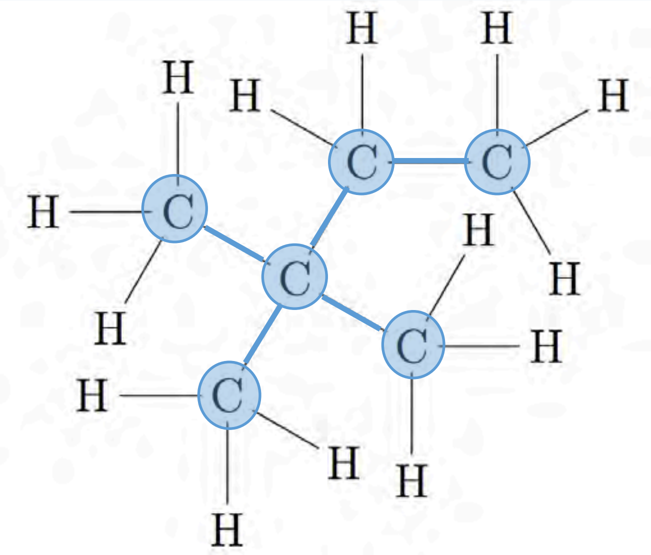
$$t_B = t_0 - \left(\frac{98}{n^2} (w_0 - \mathcal{W}(G)) + 5.5 \cdot (p_0 - p) \right)$$

$$\begin{aligned} \text{with} \quad t_0 &= 745.42 \cdot \log_{10}(n + 4.4) - 689.4 \\ w_0 &= \frac{1}{6} \cdot (n+1) \cdot n \cdot (n-1) \\ p_0 &= n - 3 \\ p &= \text{the number of shortest paths } i \to \ldots \to j \text{ of length 3 in } G \text{ with } i < j \\ &= \text{half of the number of entries "3" in the distance matrix } D \end{aligned}$$

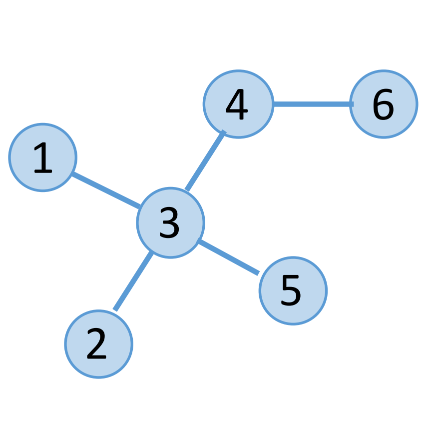
Wiener Index : Boiling Point Prediction, Example (2,2-dimethylbutan)



The chemical compound



The carbon backbone



Graph G

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

Edge Weight Matrix

Note: Depending on how you chose to label your graph, the edge weight matrix might look different. This won't matter for the subsequent calculations.

$$D = W^{n-1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 2 & 1 & 2 & 2 & 3 \\ 2 & 0 & 1 & 2 & 2 & 3 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 2 & 2 & 1 & 0 & 2 & 1 \\ 2 & 2 & 1 & 2 & 0 & 3 \\ 3 & 3 & 2 & 1 & 3 & 0 \end{pmatrix} \end{array}$$

Distance Matrix

$$\mathcal{W}(G) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} = 28$$

$$t_0 = 68.72$$

$$w_0 = \frac{1}{6} \cdot 5 \cdot 6 \cdot 7 = 35$$

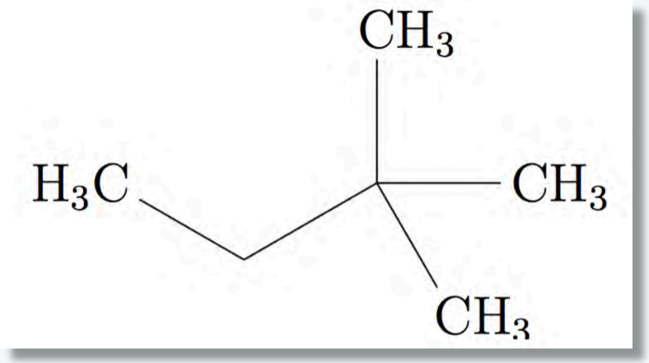
$$p_0 = 6 - 3 = 3$$

$$p = 3$$

$$\begin{aligned} t_B &= t_0 - \left(\frac{98}{n^2} (w_0 - \mathcal{W}(G)) + 5.5 \cdot (p_0 - p) \right) \\ &= 68.72 - \frac{98}{36} (35 - 28) + 5.5 \cdot (3 - 3) \\ &= 49.66 \end{aligned}$$

Calculation of Wiener Index and other parameters, as well as the resulting boiling point prediction.

Wiener Index : Boiling Point Prediction, Example (2,2-dimethylbutan)



$$\text{Predicted Boiling Point:} \quad t_B = 49.66$$

$$\text{Real Boiling Point:} \quad t_B^{\text{real}} \approx 49.7 - 50.0$$

The prediction of boiling points of paraffins based on the Wiener-Index of the corresponding molecular graph is amazingly accurate. Try it yourself (see exercises)! Intuitively, the Wiener-Index quantifies the “compactness” of a graph (or molecule). Long single chained molecules with n carbons have a smaller Wiener-Index than molecules that contain many branches. Long molecules are easier to break, and have usually a lower boiling point.