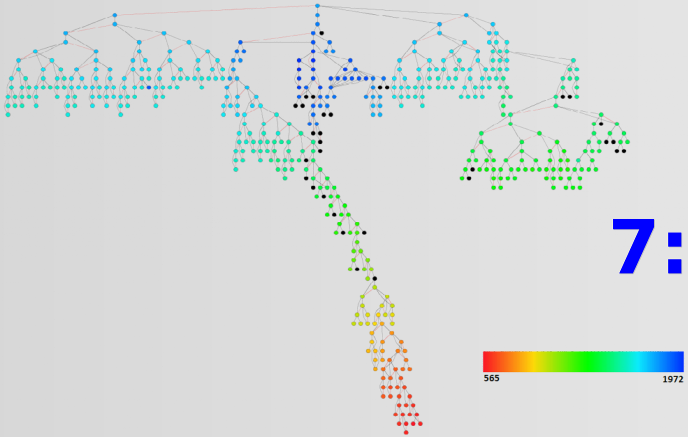
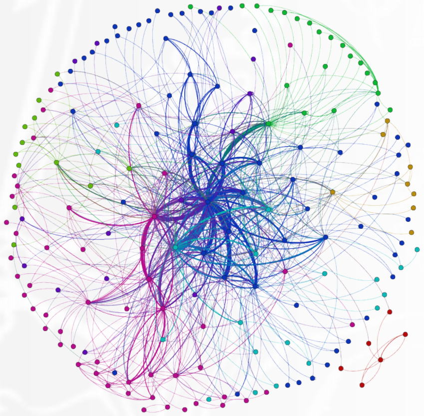
# EDAA40

# Discrete Structures in Computer Science



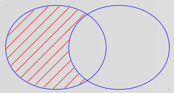

## 7: Trees and graphs



Jörn W. Janneck, Dept. of Computer Science, Lund University

$R = \{x : x \notin x\}$

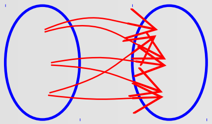
sets



$\supset$

$\heartsuit \subseteq P \times Q$

relations



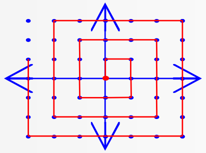
$\supset$

$f : A \longrightarrow B$

functions

$A \hookrightarrow B$

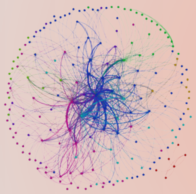
$\xrightarrow{\text{investigate}}$



infinity

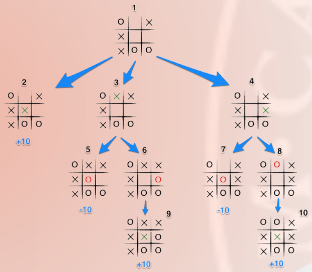
$\cup$

graphs



$\supset$

trees



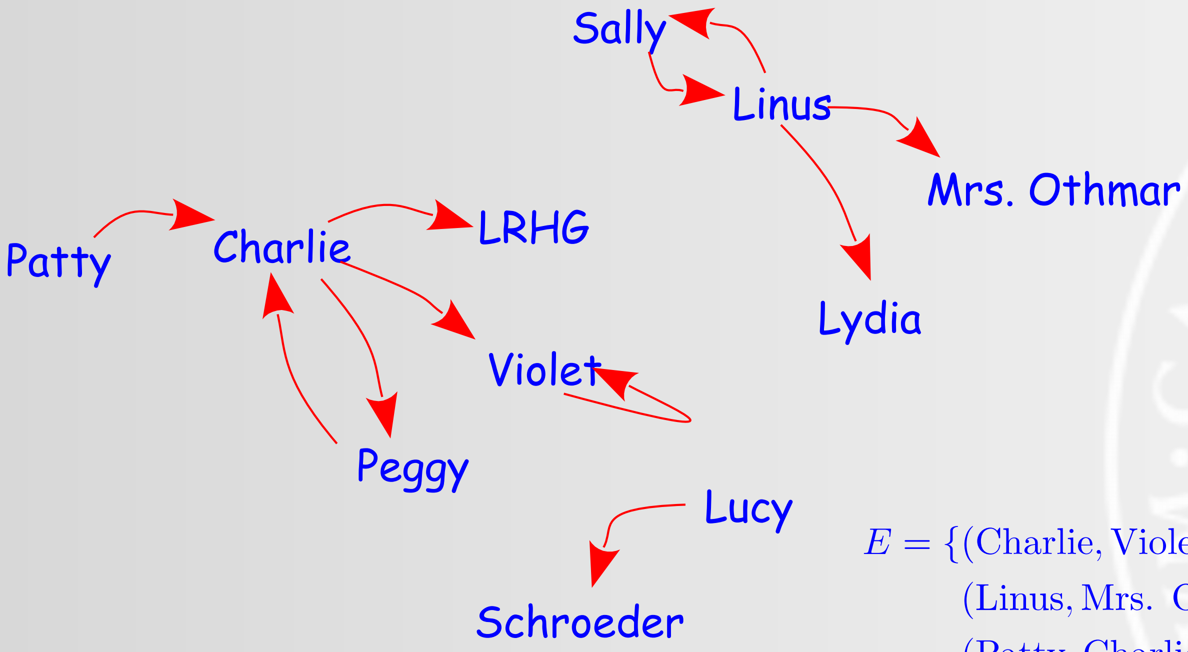
working with infinite
(or arbitrarily large) stuff

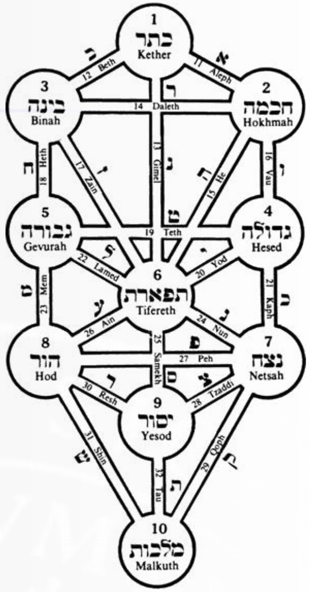
$\downarrow$

definition, construction,
recursion, induction
(also: proofs, logic)

# graphs

*A (directed) graph* is a pair $(V, E)$ where $V$ is a finite set of *vertices* (or *nodes*) and a relation $E \subseteq V \times V$, a set of *(directed) edges* (or *arcs*).



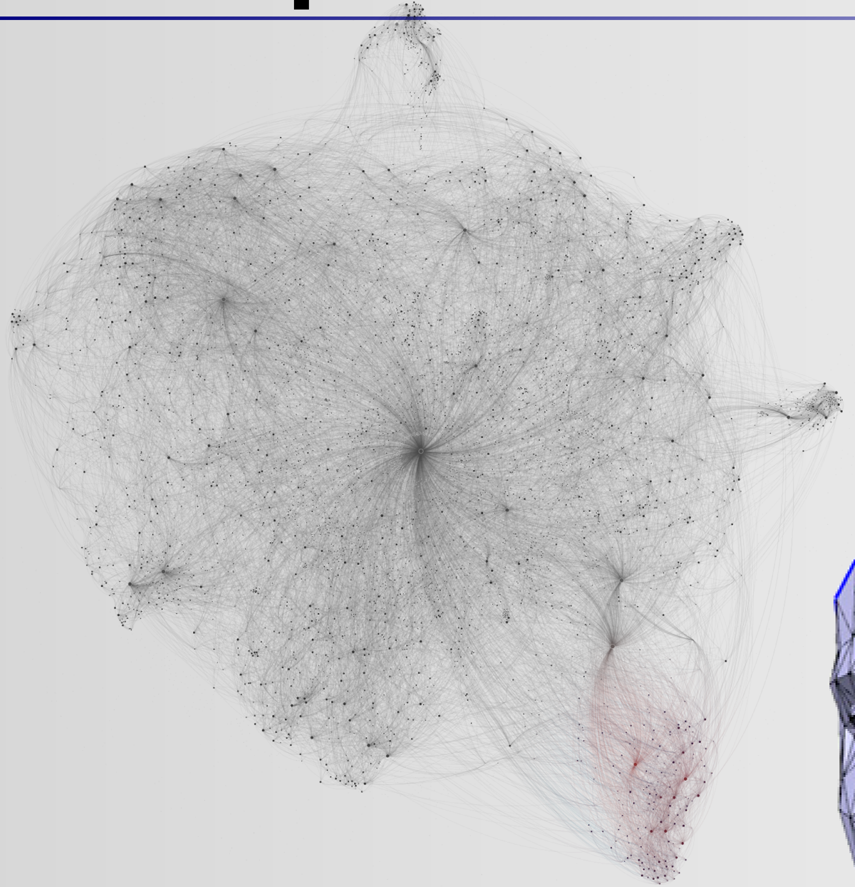




Tree of Life in Kabbalah
(ought to be: graph of life)

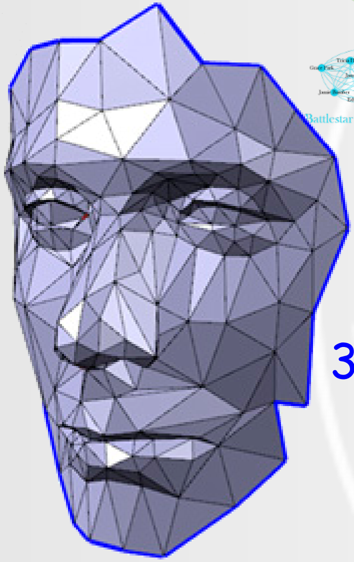
$$V = \{\text{Charlie}, \text{Patty}, \text{LRHG}, \text{Violet}, \text{Peggy}, \text{Lucy}, \text{Schroeder}, \text{Sally}, \text{Linus}, \text{Mrs. Othmar}, \text{Lydia}\}$$

$$E = \{(\text{Charlie}, \text{Violet}), (\text{Charlie}, \text{LRHG}), (\text{Charlie}, \text{Peggy}), (\text{Linus}, \text{Sally}), (\text{Linus}, \text{Mrs. Othmar}), (\text{Linus}, \text{Lydia}), (\text{Lucy}, \text{Schroeder}), (\text{Patty}, \text{Charlie}), (\text{Sally}, \text{Linus}), (\text{Violet}, \text{Violet}), (\text{Peggy}, \text{Charlie})\}$$

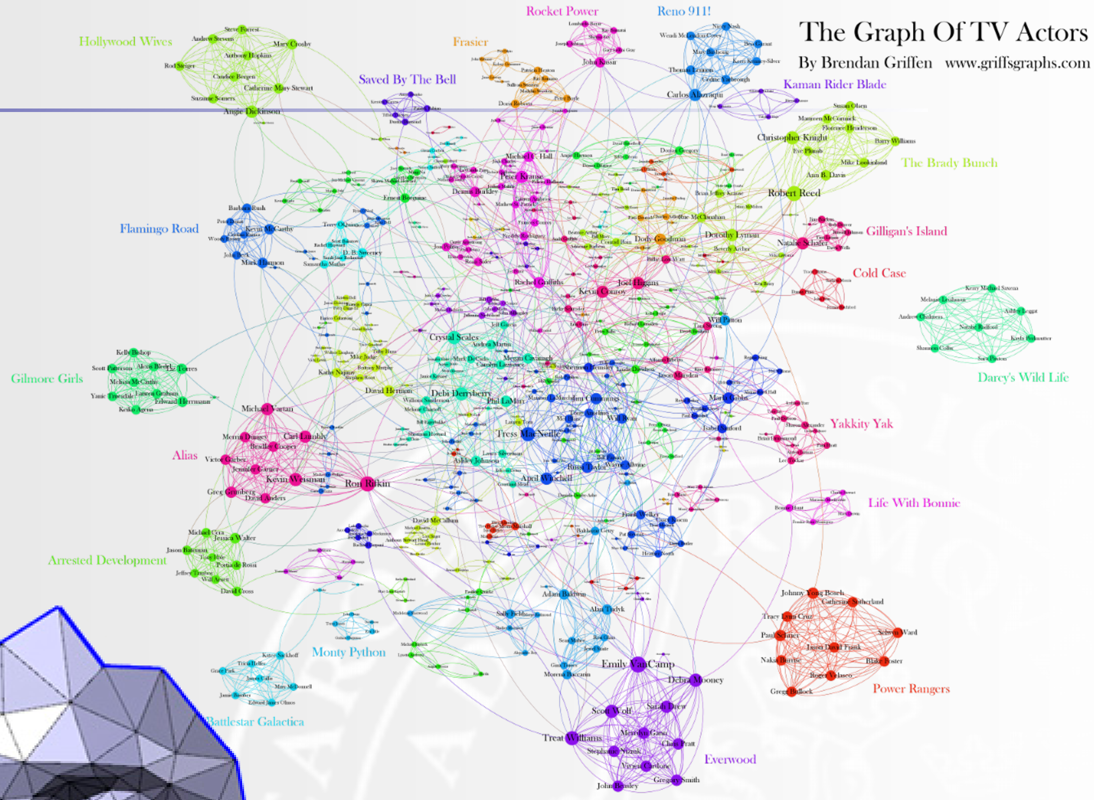
# examples



Ubuntu package dependencies
highlighted: Qt and KDE



3D mesh




TV actor collaborations

# adjacency matrix

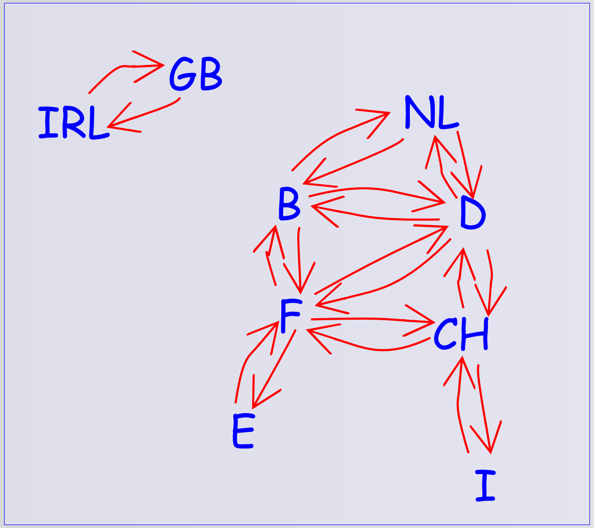
Recall the matrix representation of relations.

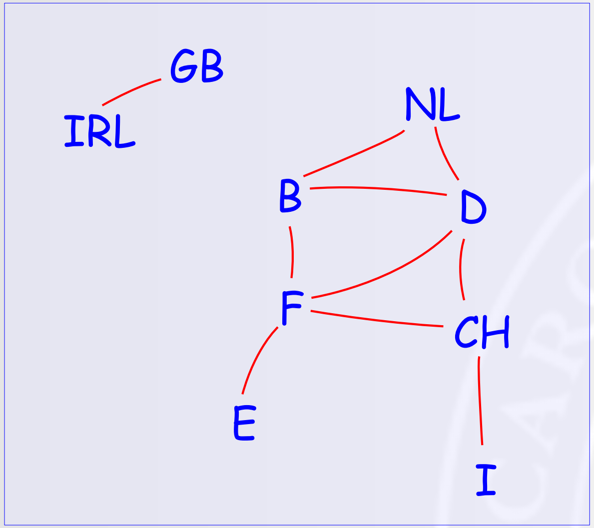
With graphs,
the adjacency matrix is always square.

| | Charlie | Linus | Lucy | Patty | Sally | Violet | Peggy | Lydia | Schroeder | LRHG | Mrs Othmar |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Charlie | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| Linus | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Lucy | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Patty | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Sally | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Violet | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Peggy | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Lydia | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Schroeder | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LRHG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Mrs Othmar | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# directed & undirected graphs

*An (undirected) graph* is a pair $(V, E)$ where $V$ is a set of *vertices* (or *nodes*) and a symmetric relation $E \subseteq V \times V$, a set of *(undirected) edges* (or *arcs*).




$$V = \{F, E, B, D, NL, CH, I, GB, IRL\}$$

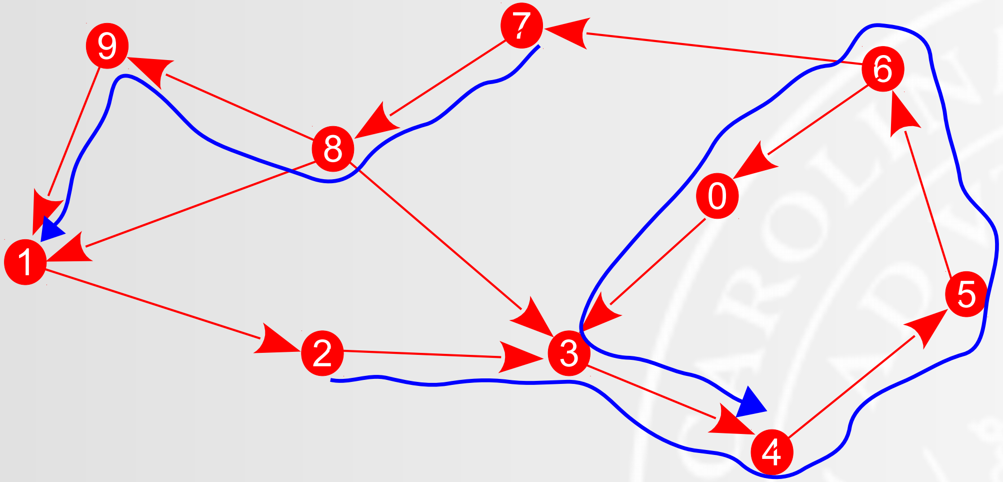
$$\begin{aligned} E = \{&(F,E),(E,F),(F,B),(B,F),(F,D),(D,F),\\ &(F,CH),(CH,F),(F,I),(I,F),(B,NL),(NL,B),\\ &(B,D),(D,B),(D,NL),(NL,D),(D,CH),(CH,D),\\ &(CH,I),(I,CH),(GB,IRL),(IRL,GB)\} \end{aligned}$$

Sometimes, an asymmetric E is used to represent an undirected Graph. In those cases, the *symmetric closure* of E is assumed.

$$E^{\leftrightarrow} = E \cup E^{-1}$$

# paths

Given a graph $(V, E)$, a *path* is a finite sequence $a_0, \dots, a_n$ in $V$ with $n \geq 1$ such that $(a_{k-1}, a_k) \in E$ for $1 \leq k \leq n$. The *length* of the path is n.




A *cycle* is a path $a_0, \dots, a_n$ where $a_0 = a_n$.
A graph that does not contain cycles is called *acyclic*.

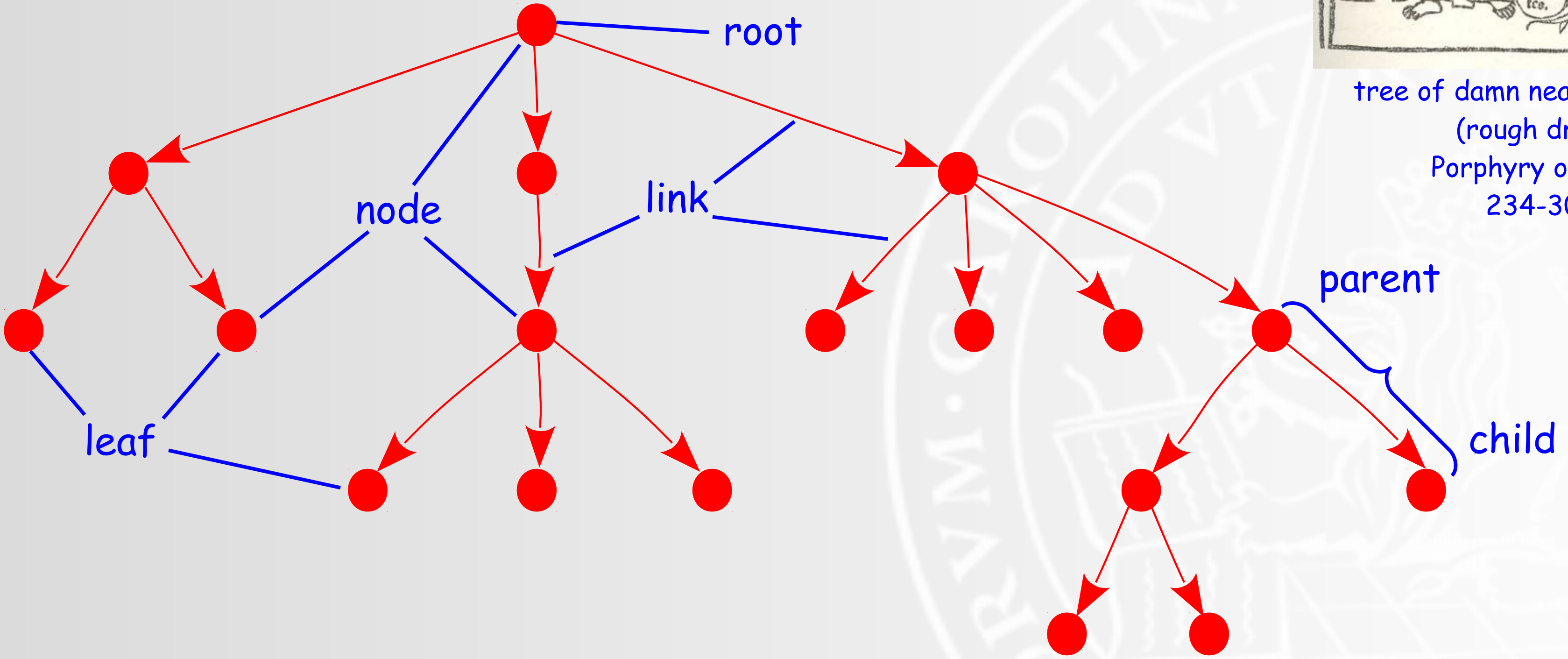
Find the cycles.

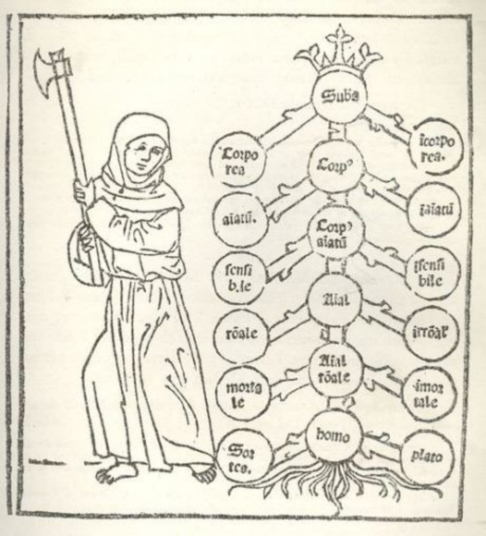
# trees

*A (rooted) tree* is a graph $(T, R)$ such that T is empty or there is an $a \in T$ such that:

(i) for every $x \in T, x \neq a$ there is exactly one path from a to x

(ii) there is no path from a to a.



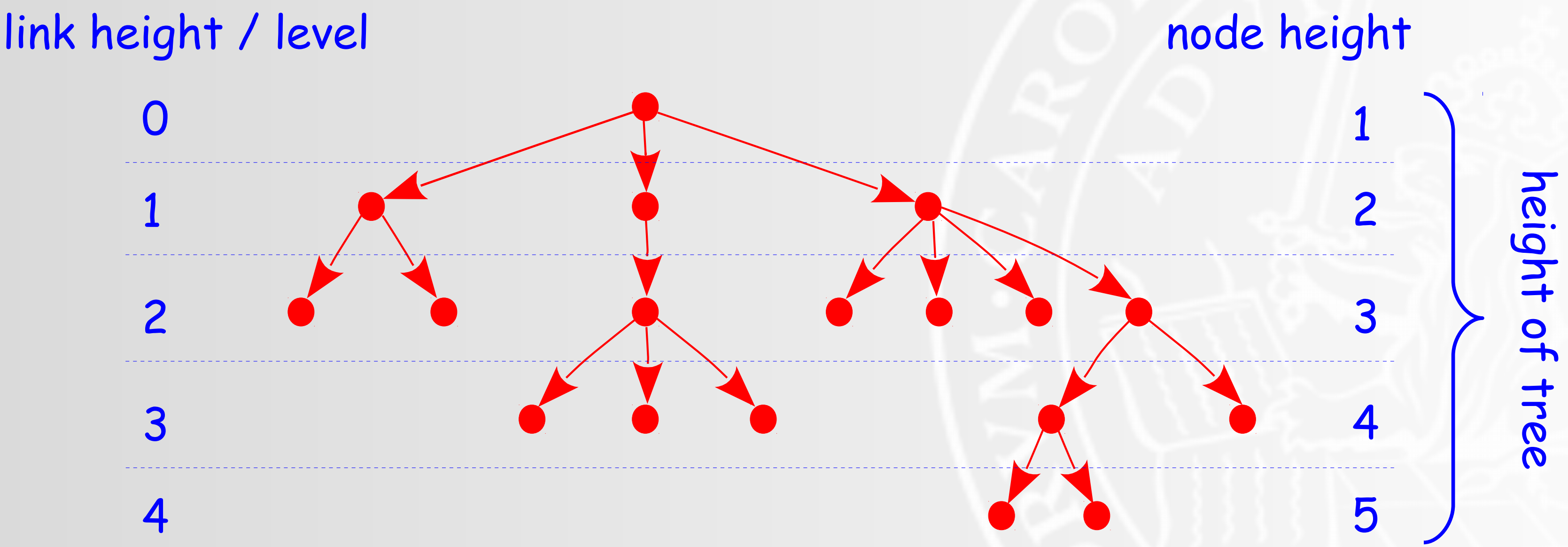



tree of damn near everything
(rough draft)
Porphyry of Tyre
234-305

# properties

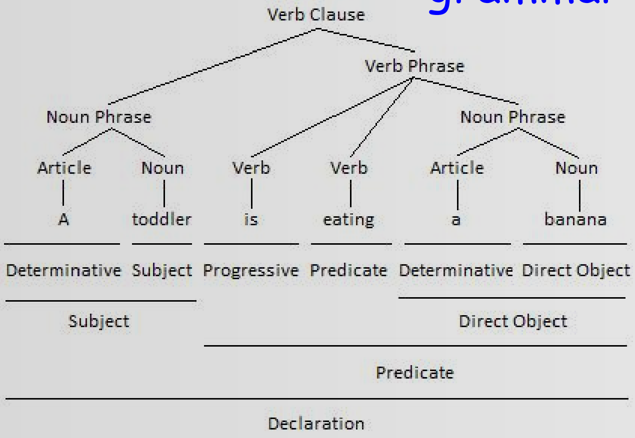
(1) Any non-empty tree has a unique root.
(2) A root has no parent.
(3) Every non-root has exactly one parent.
(4) A tree with n nodes has n-1 links.
(5) A tree contains no cycles.

What does (4) imply?

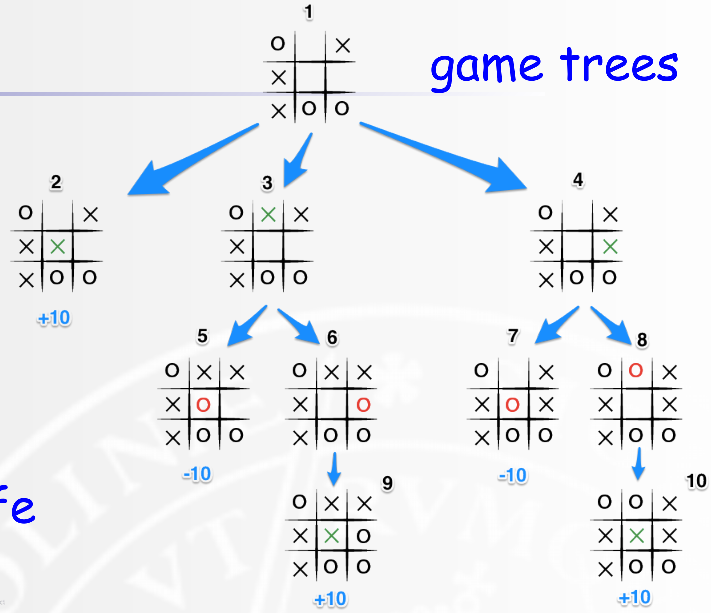


# examples

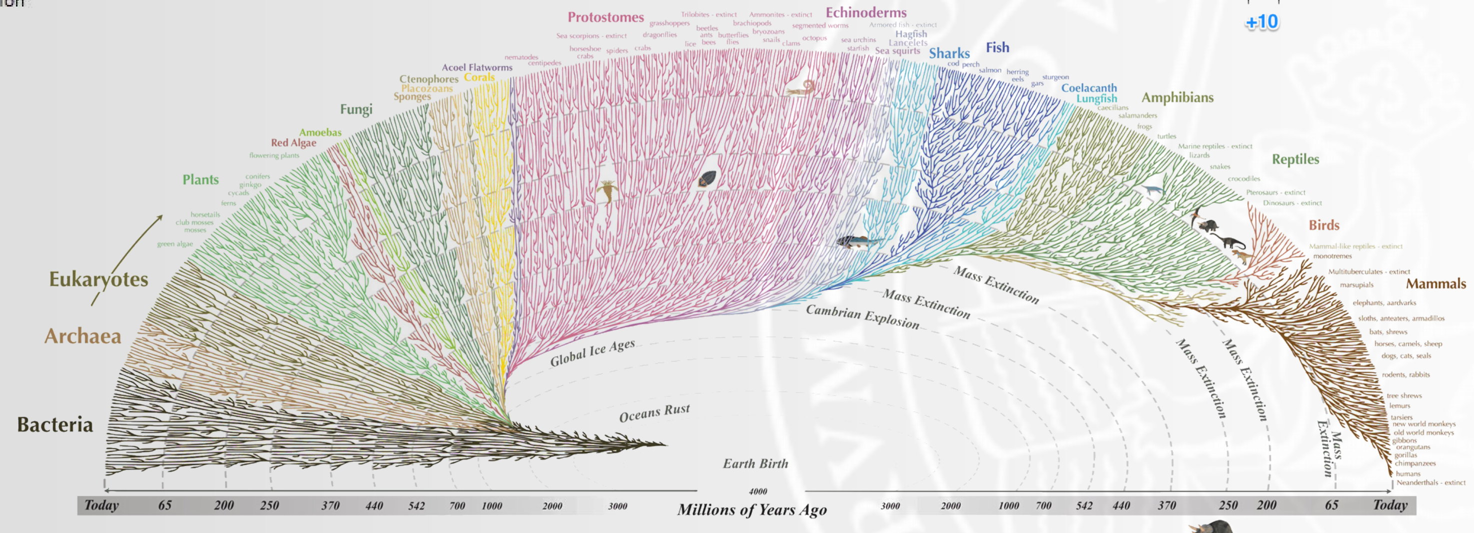
grammar trees




game trees




the actual tree of life



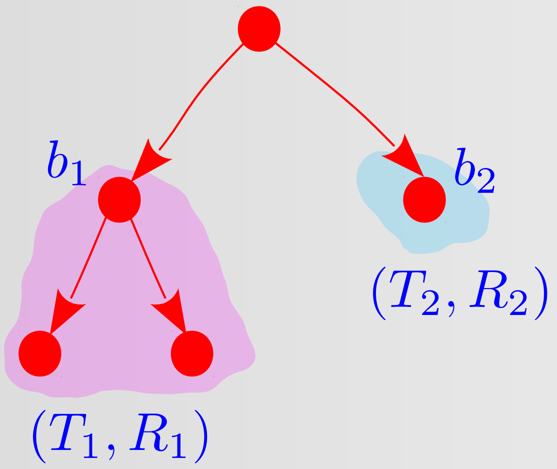

All the major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct

# defining trees recursively

(1) The empty graph $(\emptyset, \emptyset)$ is a tree.

(2) Given a family of disjoint trees $(T_i, R_i)_{i=1..n}$, i.e. $T_i \cap T_j = \emptyset$ when $i \neq j$, and with roots $B = \{b_i \in T_i : 1 \leq i \leq n\}$, as well as a fresh $a \notin \bigcup_{i \in 1..n} T_i$ we can create a new tree with root a:

$$T = \{a\} \cup \bigcup_{i=1..n} T_i \qquad R = \{(a,b) : b \in B\} \cup \bigcup_{i=1..n} R_i$$

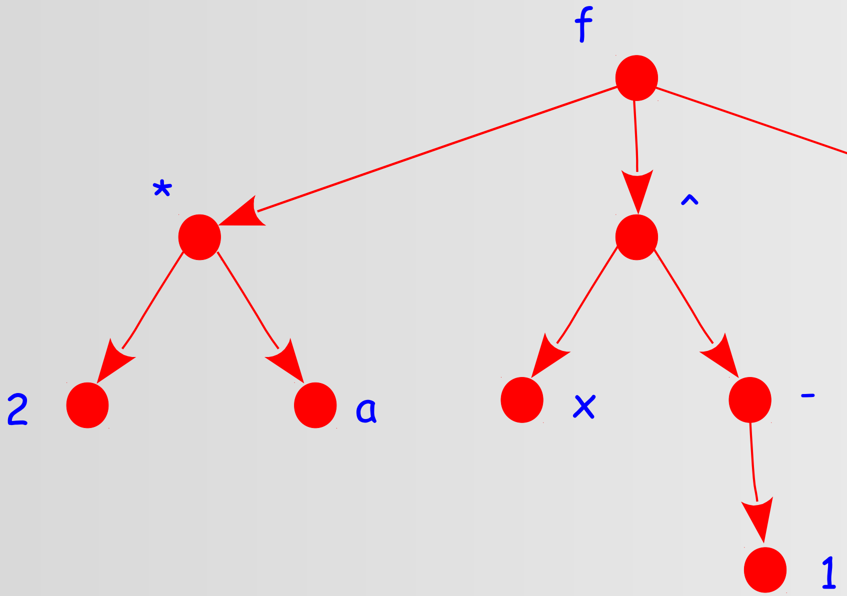



This construction is bottom-up.
Compare to the top-down construction in SLAM Section 7.2.2.

# labeled trees

Given a tree $(T, R)$, and a set of *labels* $L$, a *labeling* is a function $\lambda : T \longrightarrow L$

A tree with a labeling is called a *labeled tree.*




In practice, the labeling function is often realized by adding data to the nodes of a tree.

What expression might the tree represent?

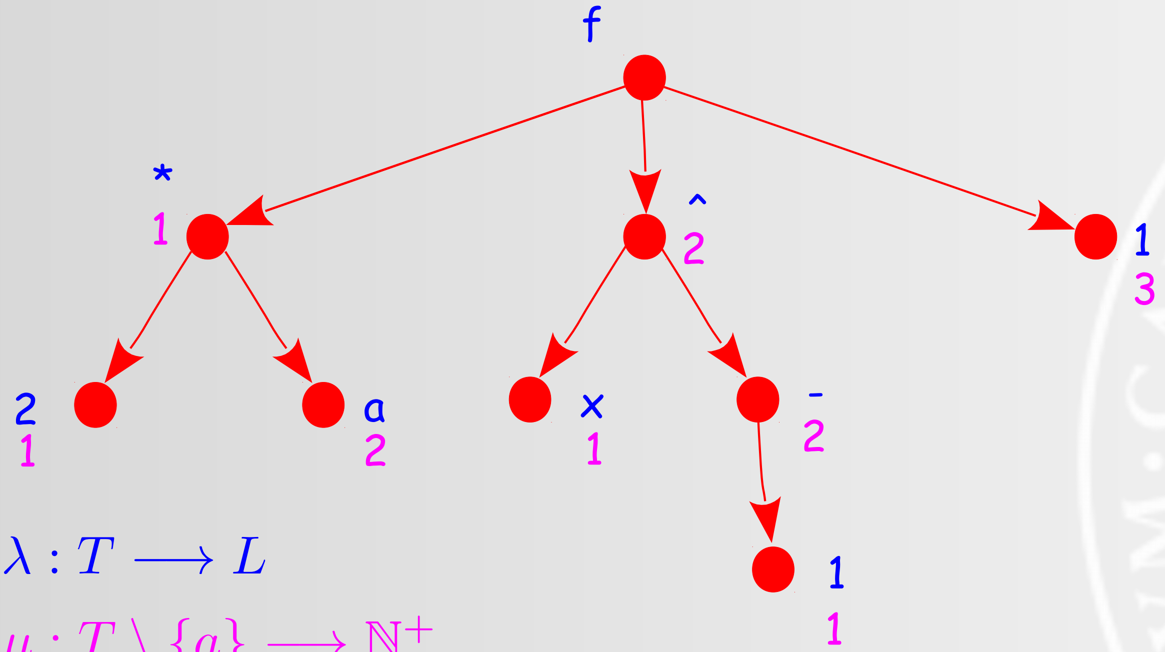
Does it?

# ordered trees

Given a tree $(T, R)$ with root a, we say it is *ordered* if there is a function

$$\mu : T \setminus \{a\} \longrightarrow \mathbb{N}^+$$

such that for every node its n children are labeled $1..n$.




$\lambda : T \longrightarrow L$

$\mu : T \setminus \{a\} \longrightarrow \mathbb{N}^+$

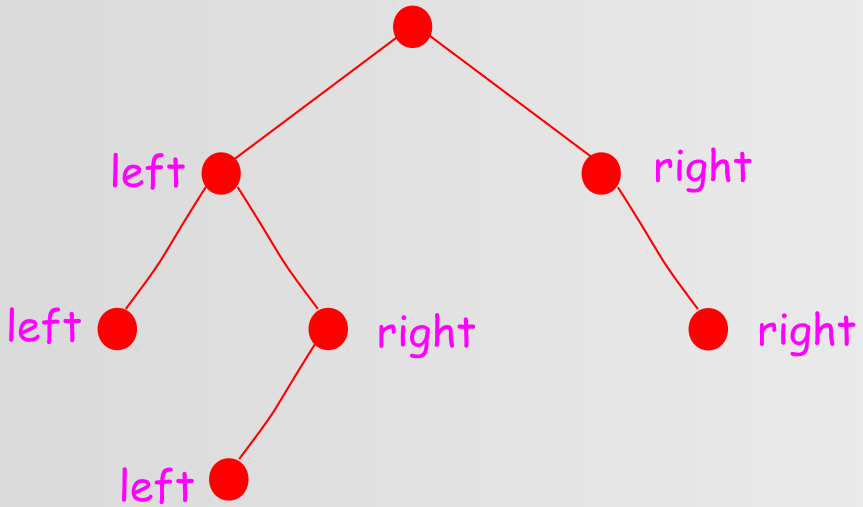
Ordering is usually represented implicitly by the left-to-right order of child nodes in the tree data structure.

# binary trees

Given a tree $(T, R)$ with root a, we say it is *binary* if every node has at most two children and there is a labeling function

$$\beta : T \setminus \{a\} \longrightarrow \{\text{left}, \text{right}\}$$

such that no two children of the same node are labeled identically.

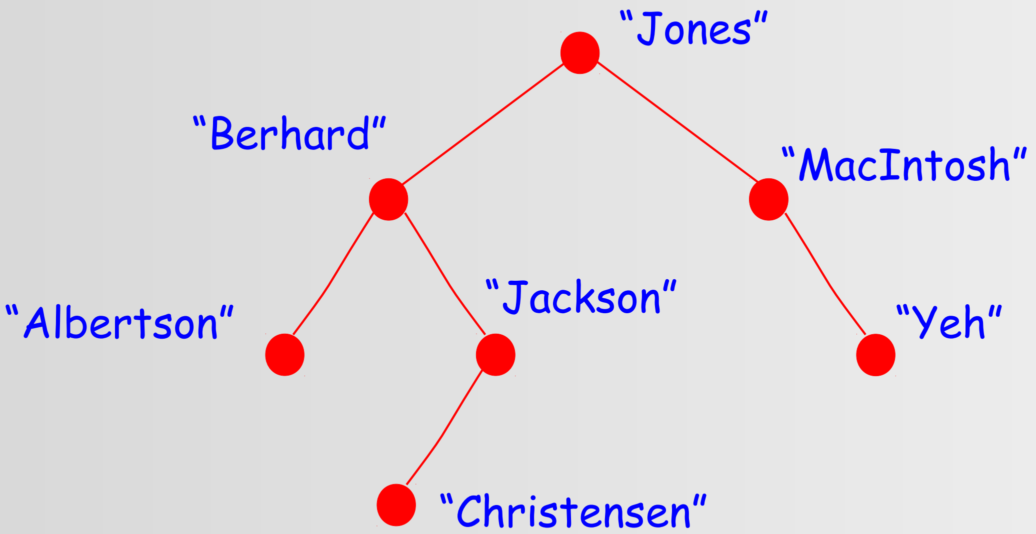



$\beta : T \setminus \{a\} \longrightarrow \{\text{left}, \text{right}\}$

Labels are usually represented by the left-to-right order of child nodes and angled links.

# binary search trees

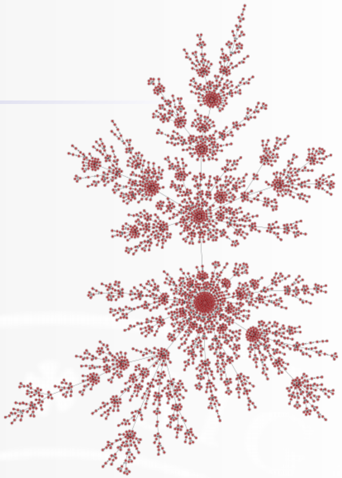
Given a binary tree $(T,R)$, with root $a$, binary labels $\beta : T \setminus \{a\} \longrightarrow \{\text{left}, \text{right}\}$, and labeling function $\lambda : T \longrightarrow L$ and a totally ordered label set L.
It is a *binary search tree* iff for all nodes their label is greater than any label in their left subtree, and less than any label in their right subtree.



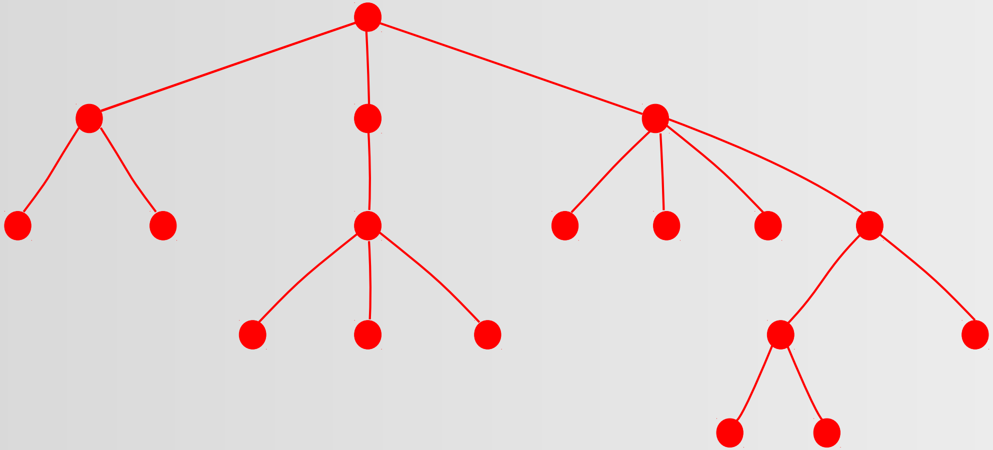

Could the same be achieved with an ordered tree, instead of a binary tree?

IOW, is a binary tree a special case of an ordered tree, or something else?

# unrooted trees



A structure $(T, S)$ is an *unrooted* (*undirected*) *tree* iff $(T, R)$ is a rooted tree and S is the symmetric closure of R.

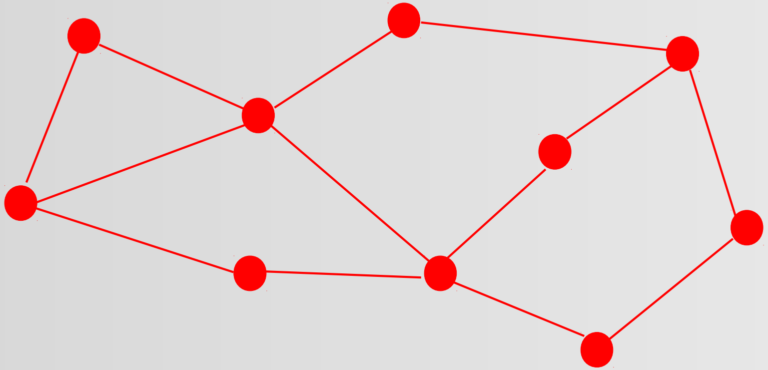


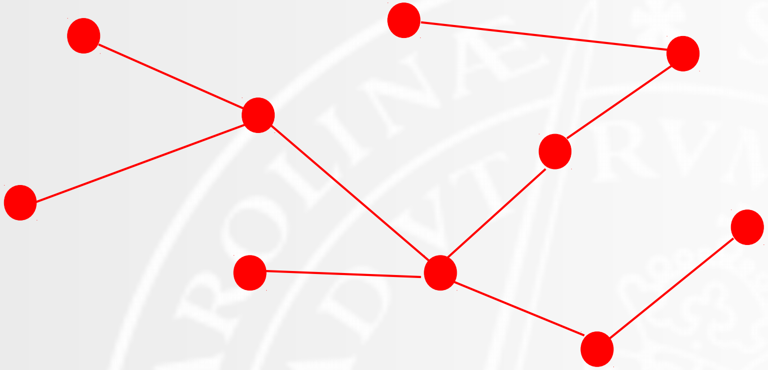
There is no unique visual representation of an unrooted tree.

# spanning trees

Given an undirected graph $(T, S)$, an unrooted tree $(T, R)$ is a *spanning tree* for it iff

$$R \subseteq S$$





How to construct a spanning tree?

There may be many spanning trees for any given graph.

# example: spanning tree protocol