Pylogeny and Evolution

Prof. Huson
Script by Sylvia Siegel

WS 2015/ 2016

1 Graphs and Trees

Definition 1.1 *Graphs represent relationships.*
A node represents an object, while edges represent relationships.

Example 1.1 ***Examples in Biology***

- *food webs*

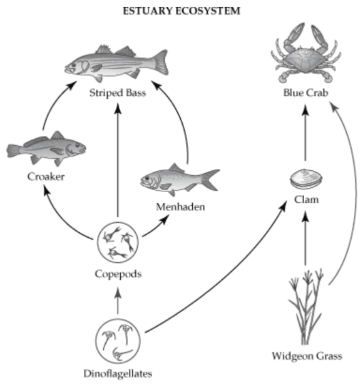


Figure 1: A food web of maritine animals.

- *metabolic pathways*

$$\boxed{\text{Substrate 1}} \xrightarrow{\text{Enzyme}} \boxed{\text{Substrate 2}}$$

Figure 2: Example for a simple metabolic pathway. Substrate 1 is converted to substrate 2 with the help of enzyme.

- *Gene interaction Networks*

$$\begin{tikzcd} & \boxed{\text{Gene 3}} \arrow[dl, "\text{supress}"'] & \\ \boxed{\text{Gene 1}} \arrow[rr, "\text{supress}"] & & \boxed{\text{Gene 2}} \arrow[ul, "\text{activate}"'] \end{tikzcd}$$

Figure 3: Example for a gene interaction network.

- *Trees*

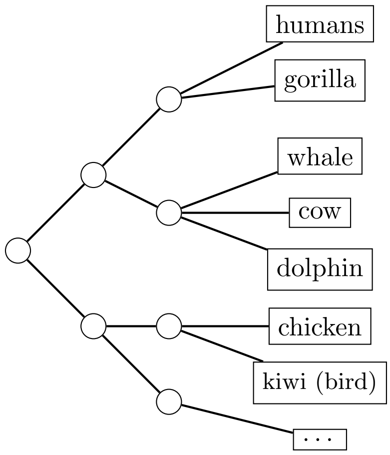


Figure 4: Shortened Tree of Life as example for a tree

⇒ *Graph theory is an important topic in mathamatics.*

<u>Some classic results:</u>

Definition 1.2 ***planar graph:***
Is graph G planar, i.e can it be drawn in the plane without edges crossing?

Example 1.2 • *K(3,2): Is it posible to conect all blue nodes to all white nodes without any edges crossing?*

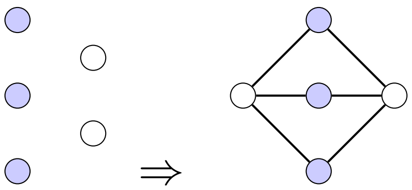


Figure 5: K(3,2)-problem and its solution

- *K(3,3)*

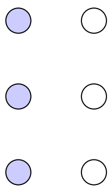


Figure 6: For the K(3,3)-Problem there is no posibility to connect each blue node with each white without any edges crossing

⇒ *K(3,3) is not planar.*

- K_4 *: connect each node out of 4 with all other nodes:*
- K_5 *is not planar*

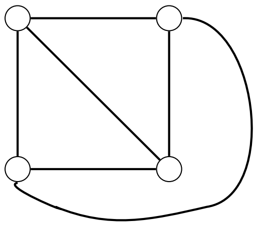


Figure 7: Solution for K_4

Definition 1.3 *G is planar* $\Leftrightarrow$ *G does not "contain"* $K(3,3)$ *or* K_5.

Example 1.3 ***The 4 color Theorem:*** *any planar given "map" can be colored in a way that no 2 countries with the same color share a comon border (edges are no borders), by only using 4 different colors.*
Proof: *by using the 5 color theorem*
First proofed in the 70s by Kenneth Appel and Wolfgang Haken using a pascal programm called "hand-hand-hand", which handled many many cases. It was the first major thorem proofed by using a computer.

Definition 1.4 *A* ***undirected Grap*** *G(V,E) has a finite node set V and edge set E, where edge* $e = \{v, w\}$ *with* $v, w \in V$.
v,w are ***endpoints*** *of e and are* ***incident*** *to e.*
Two edges e1 and e2 $\in E$ *are* ***adjacent****, if* $e1 \cap e2 \neq 0$ *i.e. they share a node.*

Definition 1.5 *The* ***degree*** *of a node* $d(v) = |\{e \in E | v \in e\}|$
(# of edges of v)

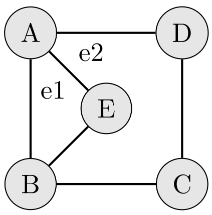
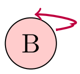


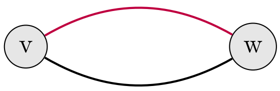
Figure 8: **Example to Definition 1.5:** d(a)= 3; e1, e2 are incident to e and e1, e2 are adjacent

We will usually assume:

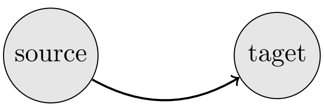
1. $|\{e\}| = 2$, *i.e. no self-loops*



2. *all edges are different*



Definition 1.6 *A* ***directed graph*** *G = (V,E) has a node set V and edge set E, with e = (V, W) = (source node, taget node)*
Like to draw this:



Definition 1.7 .

- ***Indegree:*** *Indeg(v)* $= |\{(w, v) \in E\}|$
  # of all edges that go into the node.
- ***Outdegree:*** *Outdeg(v)* $= |\{(v, w) \in E\}|$
  # of all edges that go out of the node.
- ***Degree:*** *Deg(v) = Indeg(v) + Outdeg(v)*

1.1 Implementing a graph

1. Implement directed graph then add return edges

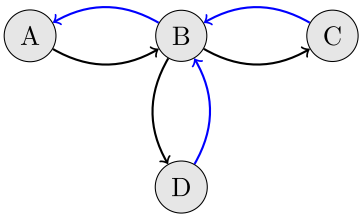


Figure 9: D (black) Graph. U (black & blue) extends Graph D

Note: source and taget does not matter in undirected graphs.

Definition 1.8 *Let* $G^* = (V^*, E^*)$, *subgraph of G = (V,E), if* $V^* \subseteq V$ *and* $E^* \subseteq E$ *with* $E^* \subseteq V^* x V^*$

Definition 1.9 *An* ***Euleianr-tour (path)*** *describes an path on a given graph G where every edge is visited just once. An eulerian path exists iff (=if and only if) the number of nodes with odd degrees is 0 or 2.*

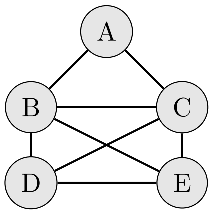


Figure 10: Example for an Eulerian path: Only D and E have a odd number auf edges. So there exists an eulerian tour in this graph.

Definition 1.10 *Let G = (V,E) be a (directed or undirected) graph. A* ***subgraph*** *G' = (V',E') of G is a graph whose node and edges sets are subsets of those of G, that is, V'* $\subseteq$ *V and E'* $\subseteq$ *E, such that the edges in E' only contain nodes in V'.*
Let $U \subseteq V$. *The subgraph* $G|_U = (U, E|_U)$ ***induced*** *by U has node set U and induced edge set* $E|_U$ *consisting of all edges in G whose endpoints both lie in U.*

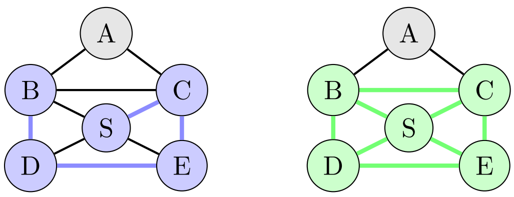


Figure 11: Example for subgraph (blue) of G (black) and an induced subgraph (green) of G.

Definition 1.11 *Let G=(V,E) be a graph. A **undirected path** in G is defined as:* $p(v_0, e_1, v_1, e_2, v_2, \dots e_k, v_k)$ *with* $v_i \in V$, $e_i \in E$ *and* $e_i = \{v_{i-1}, v_i\}$ *and* $e_i \neq e_j \forall i \neq j$.
*p **connects*** v_0 *to* v_k. *If* $v_0 = v_k$, *then p is called a **cycle**.*
If G is directed, then p is directed if $e_i = (v_{i-1}, v_i)$
*A undirected Graph G = (V,E) is **connected**, if* $\forall v, w \in V : \exists$ *path p form v to w.*
*A directed graph G = (V,E) is **(weakly) connected** if any two nodes are connected by an undirected path.*
*G is called **(strongly) connected**, if there is a directed path from v to w and a directed path from w to v.*

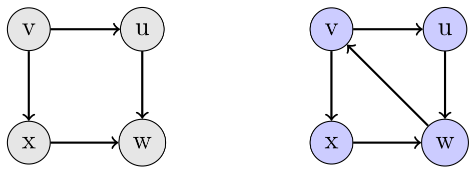


Figure 12: Example for connected graphs: The gray graph is weakly connected, while the blue graph is strongly connected.

Example 1.4 ***The Chinese Postman theorem:*** *The postman searches for the best tour to deliver the mail. He wants to visit each edge once in a tour and minimize the edges.*

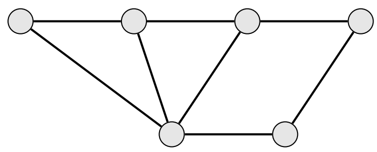


Figure 13: Example of a posible map for the postman theorem

$\Rightarrow$ *Is the graph undirected, an optimal tour can be planed in polynominal time.*
Is the graph directed, again an optimal tour can be planed in polynominal time.
But if the graph is both (directed and undirected), then the problem is NP hard.

1.2 Trees

Definition 1.12 *A **directed acyclic graph (DAG)** is a directed graph with noch directed cycles.*

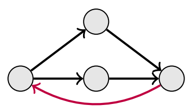
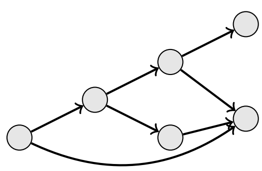
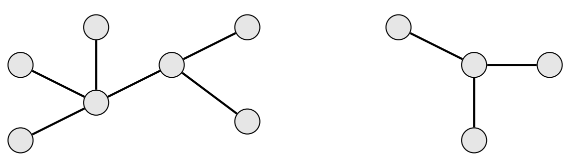


Figure 14: This graph showes a graph, without the red path this graph is a DAG, with the red path a cycle exists.

Definition 1.13 *A **Tree** is a connected graph with no undirected cycles. Any such tree is called **unrooted**.*



This graph is not a (rooted) tree.



This graph consists out of several unrooted trees and is called a **forest**.

*In a **rooted tree** exactly one (!one) node* $\zeta \in V$ *is declared **root** and we write* $T = (V, E, \zeta)$.
We can consider a rooted tree als directed graph, by directing all edges away from the root.

Definition 1.14 .
*In an <u>unrooted tree</u>: nodes of degree 1 are called **leaf**.*
*In a <u>rooted tree</u>: nodes of outdegree 0 are called **leaf**.*
*All other nodes are called **internal nodes**.*

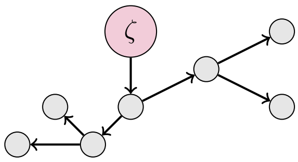


Figure 15: Example for Def 1.13: all edges are directed away from the root ζ.

Definition 1.15 *For a rooted tree :*

- *there has to be one node with indegree 0, the root*
- *The tree is called* ***biforcation (binary, fully resolved)*** *if each internal node has degree 3 (In directed rooted tree: indegree 1, outdegree 2) and* ***multiforcation****, else.*

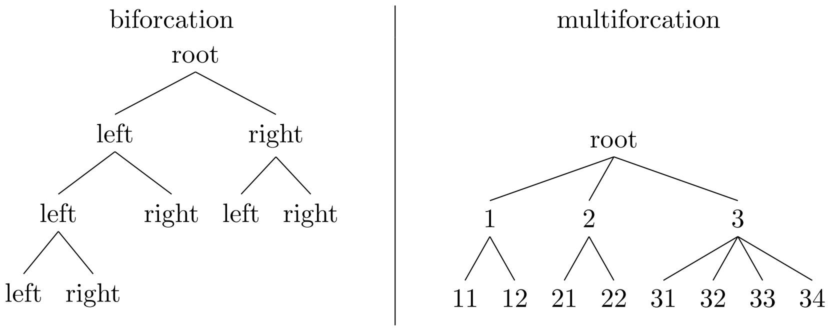
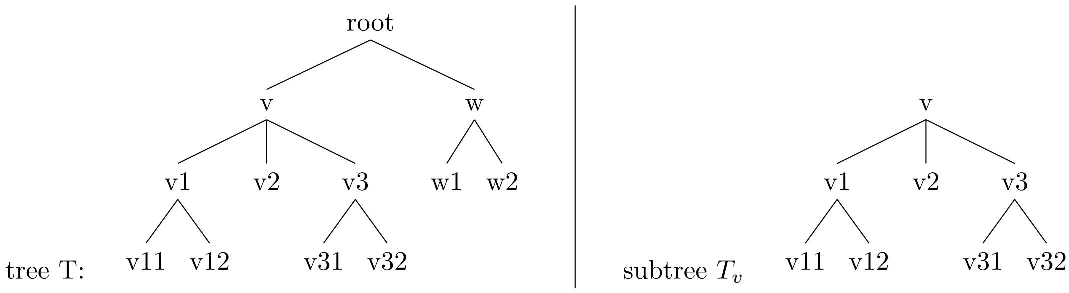


Figure 16: example trees for biforcation and multiforcation

Definition 1.16 *Let T rooted tree and $v \in V$ node in T. A* ***subtree*** *T_v rooted at v looks like this: T_v is tree with root v that is* ***induced by v*** *and all nodes reachable from v by a directed path starting at v.*



1.3 Tree traversals

For the following algorithms let T be a biforcating rooted tree (compare figure 17)
(The algorithms run analog for multiforcation trees)

- **Pre-order traversal (top down)**
  Examine (e.g. print out a lable) the root of T
  Traverse left subtree
  Travers right subtree
  Result on tree T: $\zeta, j, h, g, a, b, c, i, d, e, f$
  (Algoritmic representation: root, traverse T_j, traverse T_f)
- **Post-order traversal (bottom up)**
  Traverse left subtree
  Travers right subtree
  Examine

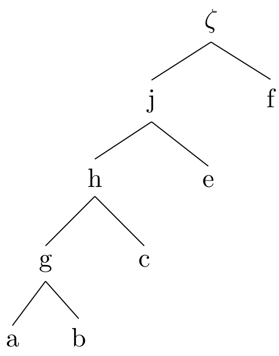


Figure 17: We will use this tree T for all traversals

Result on tree T: $a, b, g, c, h, d, e, i, j, f, \zeta$

- **Inorder traversal**
  Traverse left subtree,
  Examine,
  Travers right subtree
  Result on tree T: $f, \zeta, e, j, c, h, b, g, a$
  Extention for multification trees: Sometimes it might be required to examine the root between several children.

- **Break first**
  Put root ζ in a queue
  While queue not empty
  Pop off queue
  Examine
  Add children to end of queue

  Result on tree T:

	Q:= ζ	v $\leftarrow \zeta$
v: ζ	Q = (j, f)	v $\leftarrow$ j
v: j	Q = (f, h, i)	v $\leftarrow$ f
v: f	Q = (h, i)	v $\leftarrow$ h
v: h	Q = (i, g, c)	v $\leftarrow$ i
v: i	Q = (g, c, d, e)	v $\leftarrow$ g
v: g	Q = (c, d, e, a, b)	v $\leftarrow$ c
v: c	Q = (d, e, a, b)	...
v: d	Q = (e, a, b)	
v: e	Q = (a, b)	
v: a	Q = (b)	
v: b	Q = ()	

  The first row of the tabular represents the order in which the nodes where examined.
  Proof: Quque higher level $\rightarrow$ first in quque.

- Pre-, post- and Inorder traversal are **depth first traversals**.

2 References

2.1 Figures

Figure	Source
Figure 1	`http:////mdk12.msde.maryland.gov//instruction//clg//public_release//biology//g3_e5_i2.html`