FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF BUSINESS
AND ECONOMICS

Introduction to Computer Science: Graphs and Trees

Prof. Dr. Martin Matzner

Learning Objectives

After this lecture you can …

… define and distinguish graphs and trees

… classify binary trees

… insert elements in binary (search) trees

… delete elements in binary (search) trees

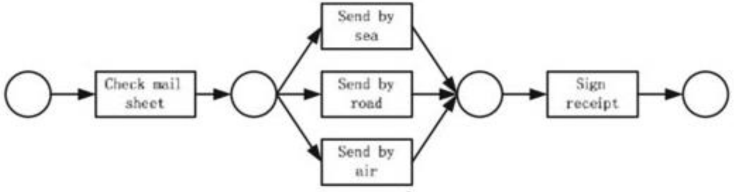
… set up data structures to store graphs and trees

… describe self-balancing binary trees

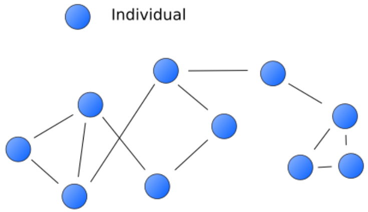
Business Information System Engineering

Examples of graphs and trees

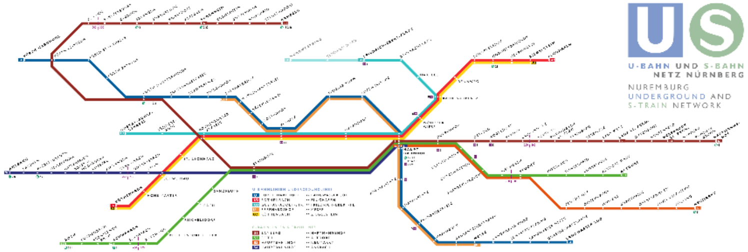
Process Models



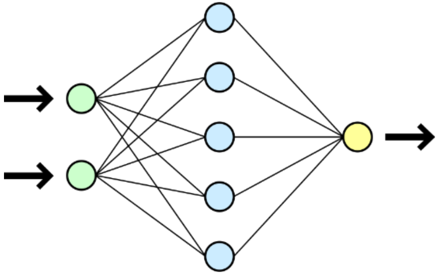
Social Networks



Maps



Neural Networks



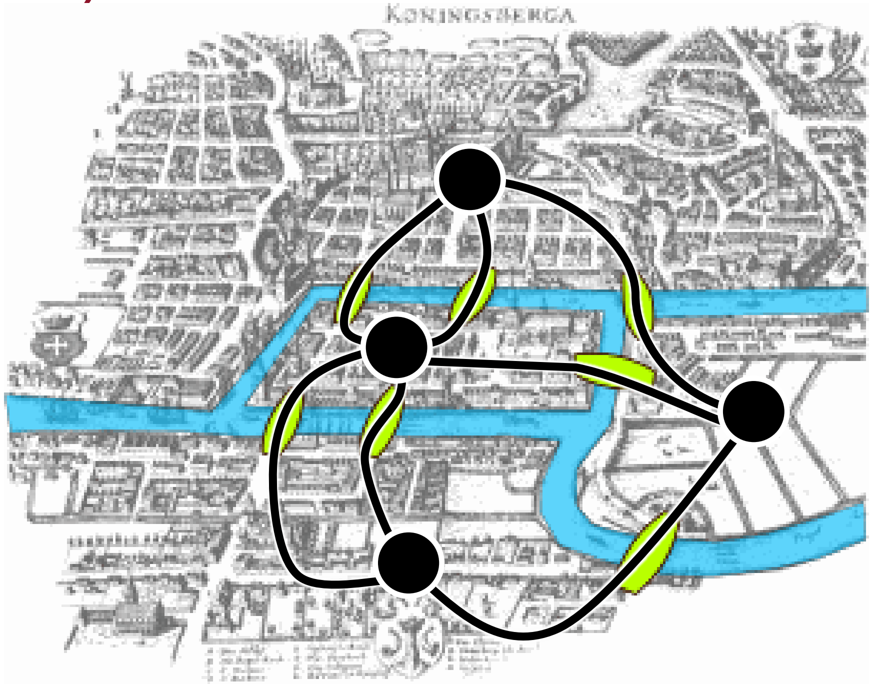
History of graph theory – Euler (1736)

The Seven Bridges of Königsberg

Devise a walk through the city that would cross each of those bridges once and only once.

By specifying the logical task unambiguously, solutions do not involve

- reaching an island or mainland bank other than via one of the bridges
- accessing any bridge without crossing to its other end



Definition of a Graph

Directed Graphs

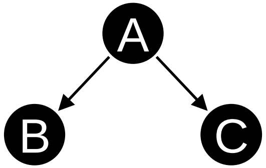
Definition

Definition: $\boldsymbol{G = (V, E)}$ ***where*:**

- ***V*** is the set of all **vertices in a graph**
  - $v \in V$ is one vertex of a graph
  - for each vertex we draw one node
- ***E*** is the set of all **edges in a graph**
  - $\mathrm{e} \in \mathrm{E}$ is one edge of a graph
  - $\mathrm{e} = (\mathrm{u}, \mathrm{v})$, e is a relation between two vertices
    - u is the start vertex
    - v is the end/destination vertex
  - For each edge, we draw an arrow from the start to the end node

$G = (V, \emptyset), with$
$V = \{A, B, C\}$
$\emptyset$ is an empty set

$G = (V, E)\ with:$
$V = \{A, B, C\},$
$E = \{(A, B), (A, C)\}$



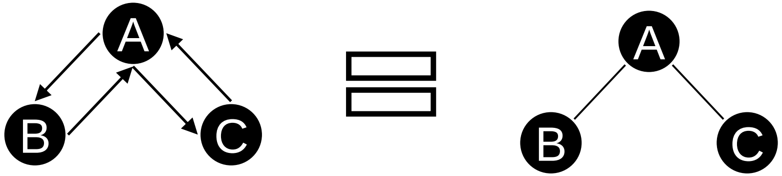
Definition of a graph

Undirected Graphs

Definition

If a graph: $\boldsymbol{G = (V, E)}$ *has a symmetric set of edges (E) we speak of so-called undirected graphs*:

$G = (V, E)$ *with:*
$V = \{A, B, C\},$
$E = \{(A, B), (A, C), (B, A), (C, A)\}$



Symmetry of E:

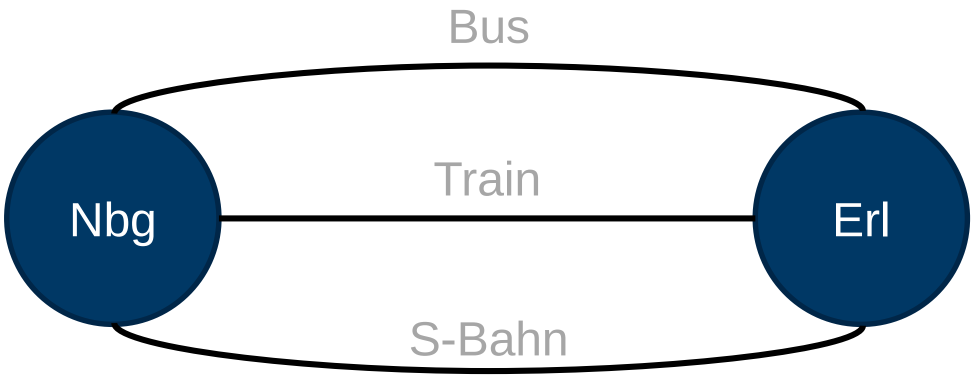
- $\forall\ (e(u, v) \in E\ \exists\ e(v, u) \in E)$
- For all edges from u to v in E there is also an edge from v to u.

We leave out arrows and simply use lines in undirected graphs instead

Definition of a graph

Multigraphs

- In contrast to simple graphs, multigraphs allow so-called parallel edges
- Edges are parallel if they have the same start and end vertices
- Example: Public Transport travelling from Nuremberg to Erlangen
- $G = (\{Nbg, Erl\}, E\{(Nbg, Erl), (Nbg, Erl), (Nbg, Erl)\})$

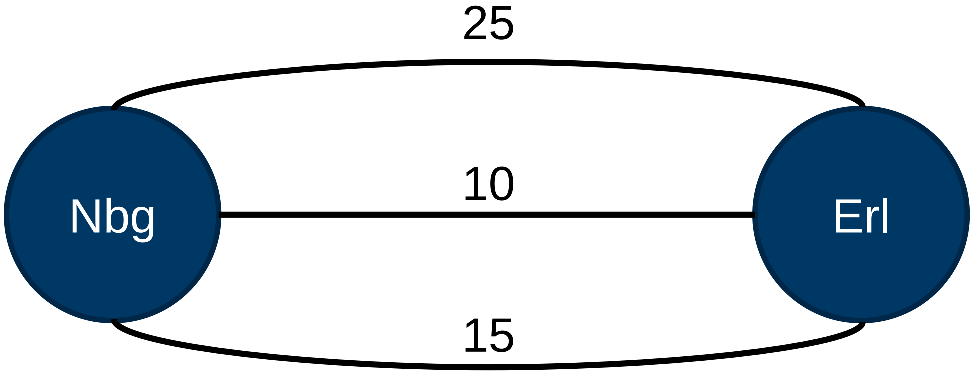


Weighted Edges

Measuring distance

- Edges can be annotated (weighed) with values like costs, time, or anything you find useful.
- In the previous example, we could either travel by Bus, Train or S-Bahn. Each of these means by travel takes a different time. In the following graph, we see three means to travel from Nürnberg to Erlangen (and back), where the ways take 25, 15, or 10 minutes.

Formal notation:
Undirected
G(V = {Nbg, Erl},
E = {(Nbg, Erl, 10),
(Nbg,Erl, 15),
(Nbg,Erl, 25)}



Adjacency Matrices

Data structure to store graphs

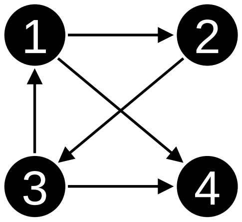
Definition

- ***Formal definition:***
  - Let $G(V,E)$ be a graph with $V = \{v_1, \dots, v_n\}$.
  - Then the $n \times n$ Matrix

    $$A_G = (a_{i,j})_{1 \leq i,j \leq n} \;\; where \qquad a_{i,j} = 1 \;\; if \;(v_i, vj) \in E$$
    $$a_{i,j} = 0 \;\; otherwise$$

    is called adjacency matrix of Graph G
- **Note:**
  - For weighed graphs, we use the weight instead of the 1 to indicate there is an edge

Example



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

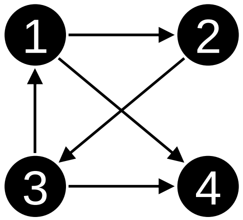
Adjacency Matrices

Data structure to store graphs

- ***(Simplified) Explanation:***
  - Imagine every node as a row and a column in the matrix:

Vertices	1	2	3	4	**Can I arrive here starting from n?**
1	No	Yes	No	Yes	
2	No	No	Yes	No	
3	Yes	No	No	Yes	
4	No	No	No	No	
Can I go to n from here?					

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



If you look at the column of a node, you find all the incoming edges

If you look at the row of a node, you find all the outgoing edges

Adjacency Matrices

Data structure to store graphs

- ***(Simplified) Explanation:***
  - *Now, since we know vertex 1 is in row 0 and column 0, vertex 2 is in row 1 and column 1. We can omit the vertices numbers in the resulting matrix.*
  - *After that we encode a “Yes” as an answer to the previous questions to 1 and a “No” to 0.*

Vertices	1	2	3	4
1	No	Yes	No	Yes
2	No	No	Yes	No
3	Yes	No	No	Yes
4	No	No	No	No

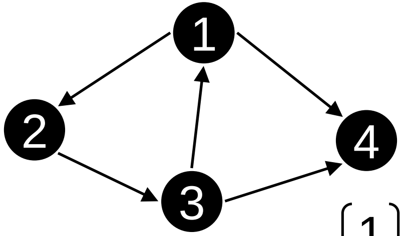
$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Graph drawing

Graphs might look different even though they are the same

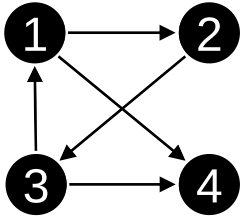
If we draw a graph from an adjacency matrix, the result might look different:

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



(1)

$$=$$



(2)

Even though the graphs (1) and (2) look differently, they are the same. The only difference is the placement of the vertices.

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
SCHOOL OF BUSINESS AND ECONOMICS

Graph Drawing

Adjacency Matrix to Graph

Classroom Task

Can you draw the graph from the following matrix?

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

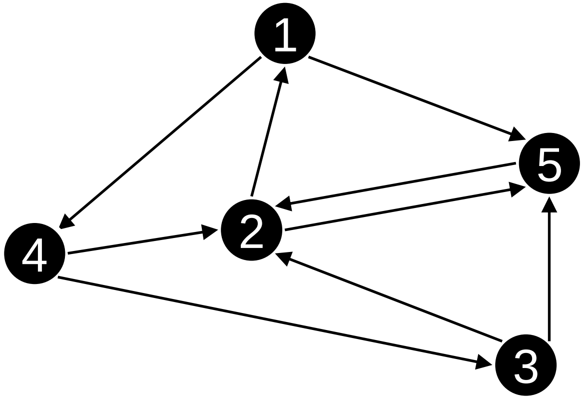
Adjacency List

Definition

Classroom Task

Can you draw the graph from the following matrix?

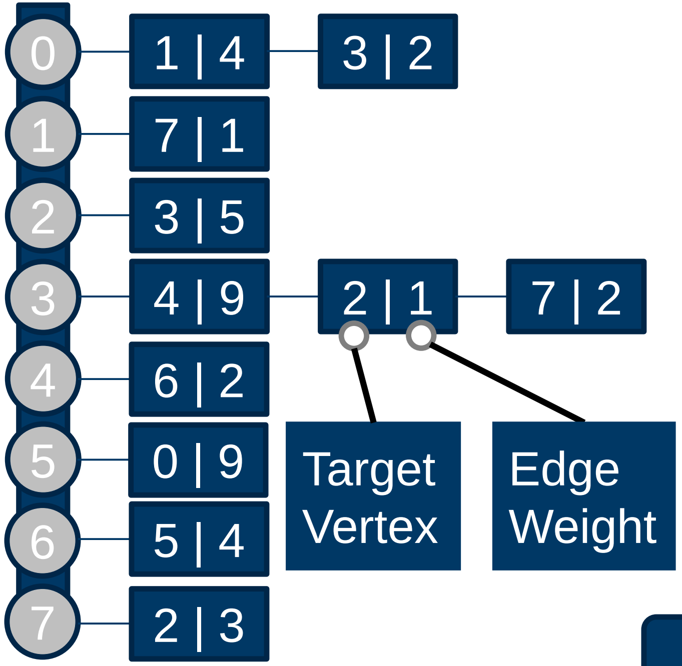
$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \equiv$$

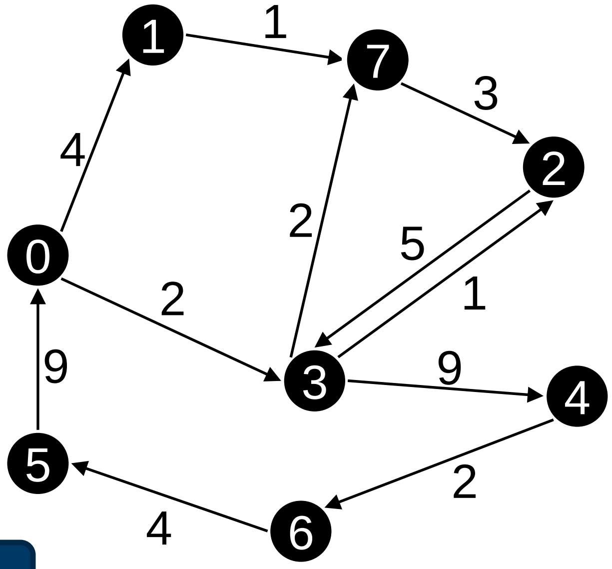


Adjacency List

Data structure to store graphs

Definition





Example

cf: https://visualgo.net/en/graphds

Adjacency List

Definition

Question

How could we implement an adjacency list?

Adjacency List

Definition

Classroom Task

How could we implement an adjacency list?

In general, there are three “well-known” ways to implement an adjacency list:

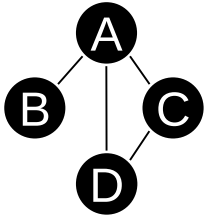
Hash Table

Arrays with indices

Object-oriented

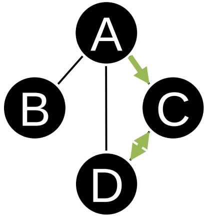
Definition of “Ways”

Walk, Trail, Path



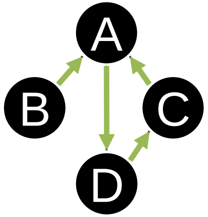
- A (finite) Walk:
  - is sequence of vertices leading to a vertex sequence.

<u>Walk:</u>
A-C-D-C



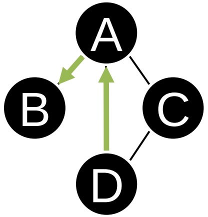
- A (finite) Trail:
  - Is a walk where all edges are distinct

<u>Trail:</u>
B-A-D-C-A



- A (finite) Path:
  - Is a walk where all vertices are distinct

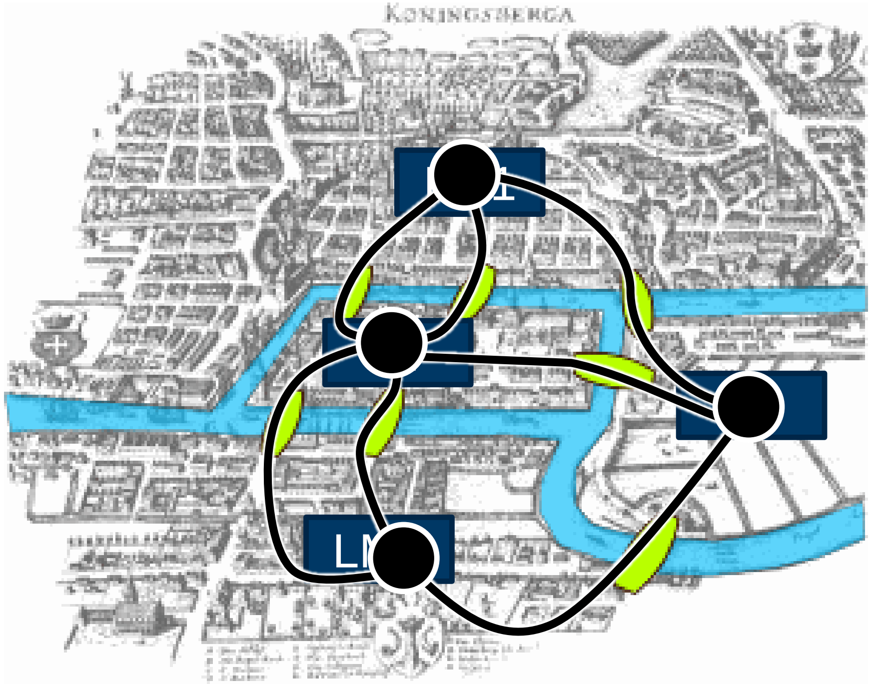
<u>Path:</u>
D-A-B



Task: The first graph drawn (1736)

The Seven Bridges of Königsberg

- What‘s the definition of the graph Euler used in Königsberg?
  LM = Landmass
  I = Island
- Is the graph directed?
- Is it a multi or a simple graph?



Solution

Undirected Multigraph $G = (E, V)$

$V = \{LM1, LM2, I1, I2\}$

$E = \{(LM1, I1), (LM1, I1), (LM1, I2),$
$(I1, I2), (I1, LM2), (I1, LM2), (I2, LM2)\}$

Finding in Graphs

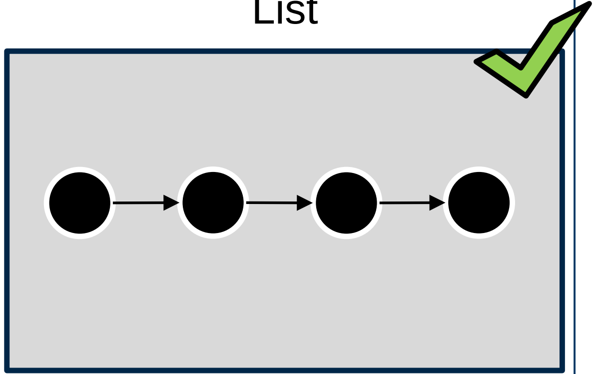
Short

More on Graphs and Finding Elements in an upcoming "Short"!

Abstract data structures

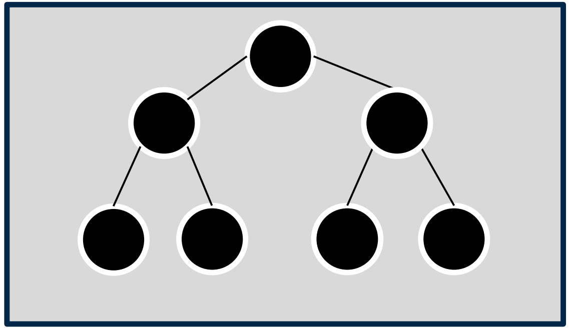
Data structures with links

List



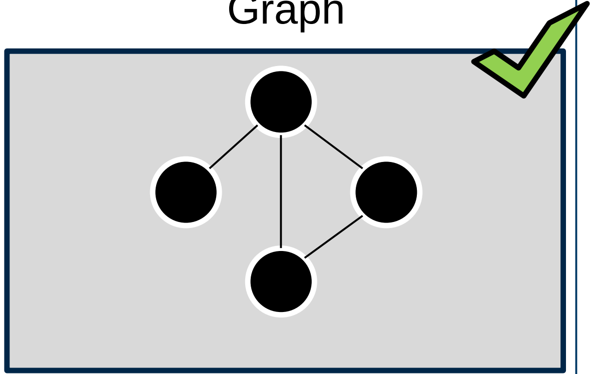
Each element has at most 1 predecessor and one successor

Tree



Each element has at most 1 predecessor and 0 to n successors

Graph



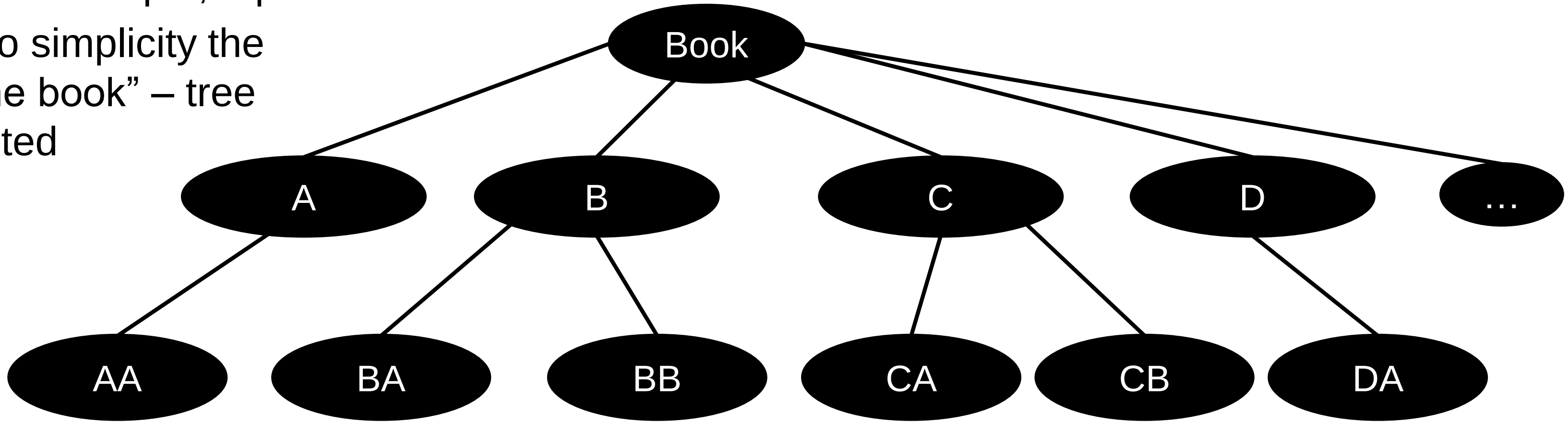
Each element has 0 to n predecessors and 0 to n successors

Trees

Representation of hierarchical information

- For example, a phone book:

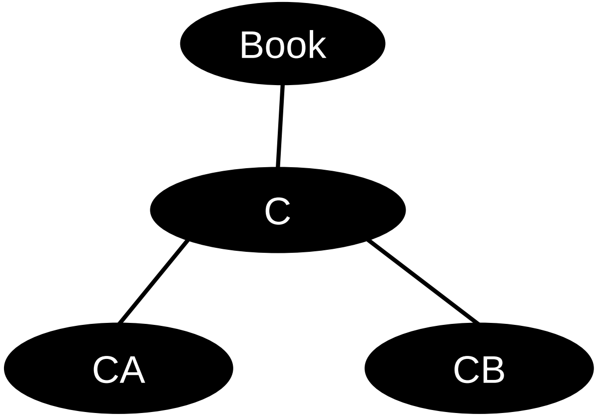
Due to simplicity the "phone book" – tree is limited



Each name contains several letters. Each level of the tree represents one letter.

Trees

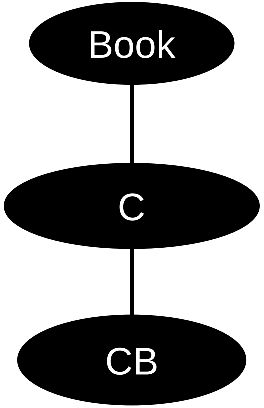
Searching a person with the name CB



With one search step, we limit the phone book search space from 26 (A - Z) nodes to 1 (C).

Trees

Searching a person with the name CB

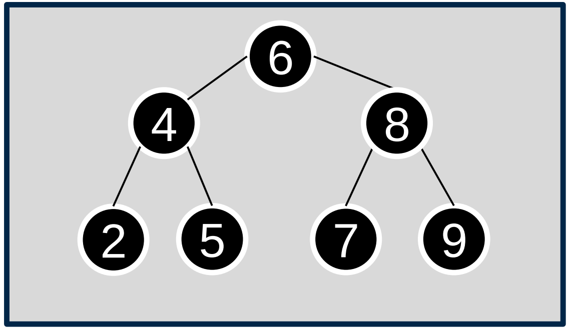


In the next step we found the name “CB”

Tree

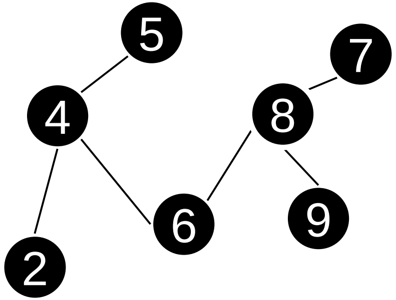
Definition

Definition



Trees are connected acyclic undirected graphs.

A tree or not a tree?



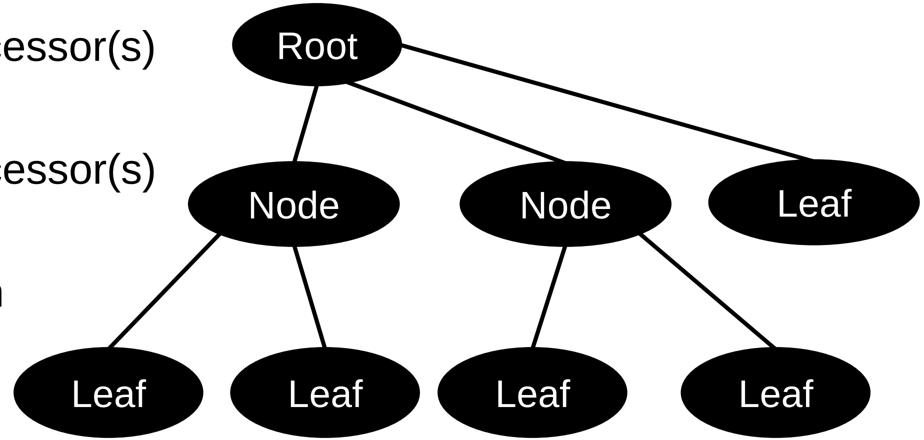
Iff (if and only if) there is exactly one path between two vertices of your choice, it is a tree.

<u>Reminder Path</u>:
A walk between two vertices where every vertex and edge is distinct.

Trees

Nodes

- **Root**
  - The only vertex without any predecessors, (“Beginning of the tree”)
- **(inner) Node**
  - Node with a predecessor and n successor(s)
- **Leaf**
  - Node with a predecessor and 0 successor(s)
- **Node:**
  - Nodes are what we called vertices in graph theory.

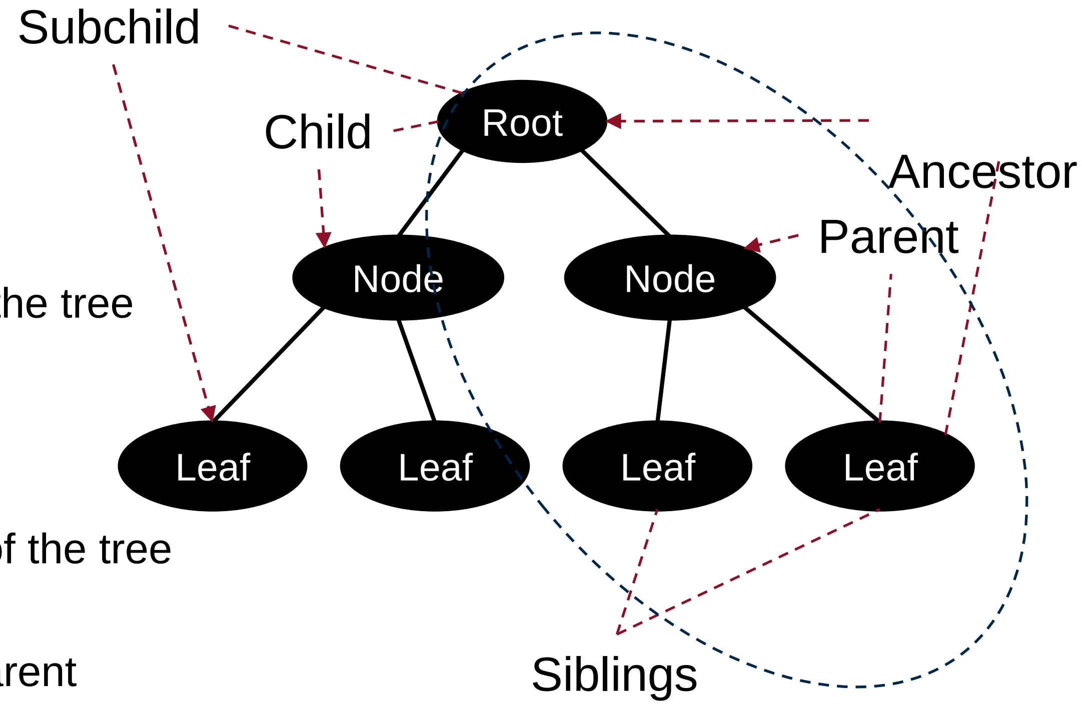


Trees

Some more nodes

- **Parent**
  - Direct predecessor
- **Ancestor**
  - Indirect predecessor in the tree
- **Child**
  - Direct successor
- **Subchild**
  - Successor in the latter of the tree
- **Siblings**
  - Nodes with the same parent

Neighbors of Node



Trees

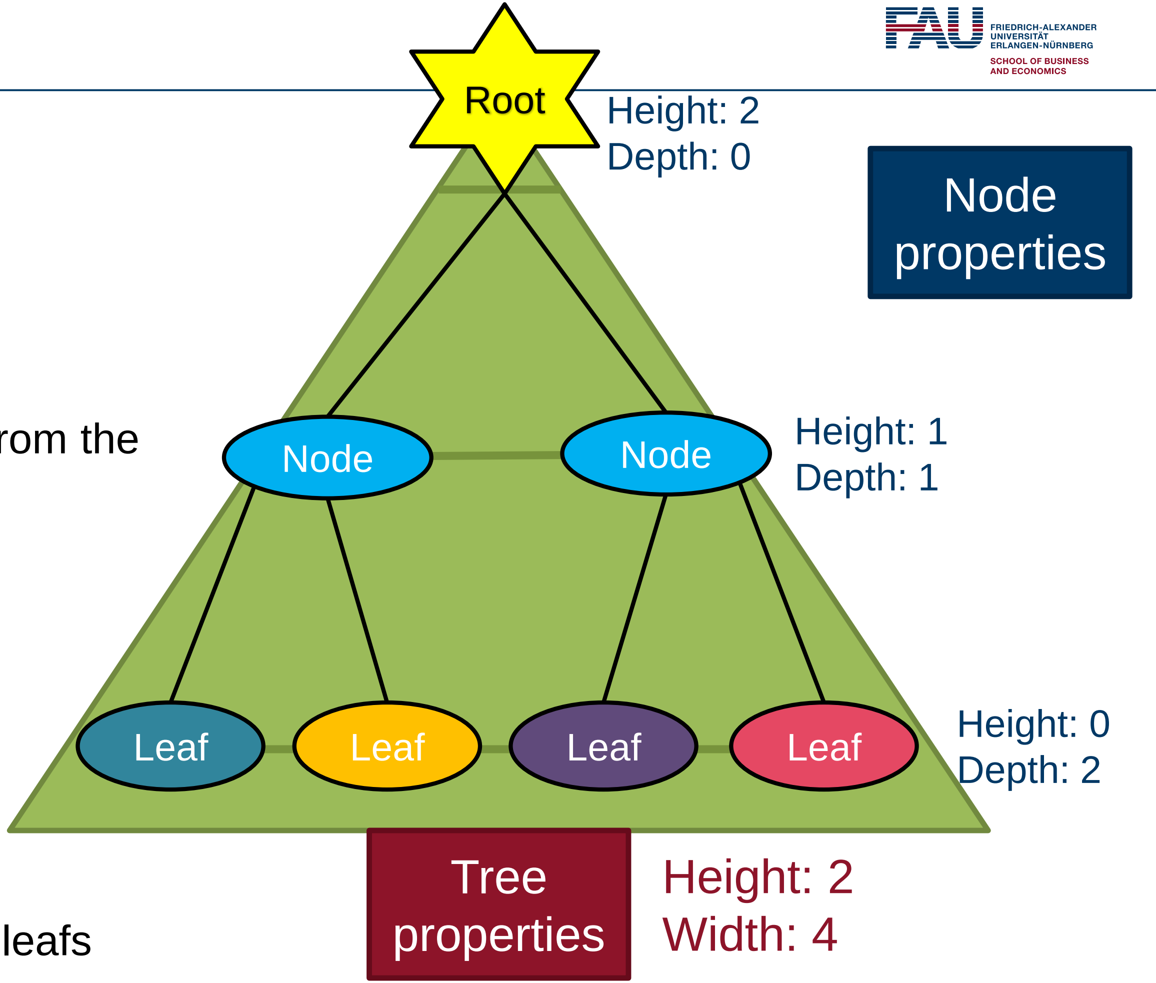
Properties

Node properties

- **Height**
  The number of edges to walk from the node to a leaf.
- **Depth**
  The number of edges to walk from a node to the root.

Tree properties

- **Height**
  The height of the root node
- **(Vertical) Width**
  The longest path between two leafs

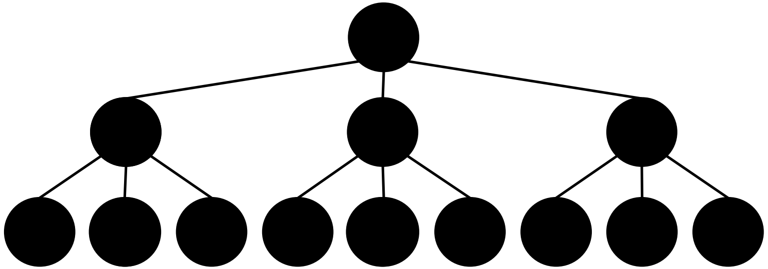


Trees

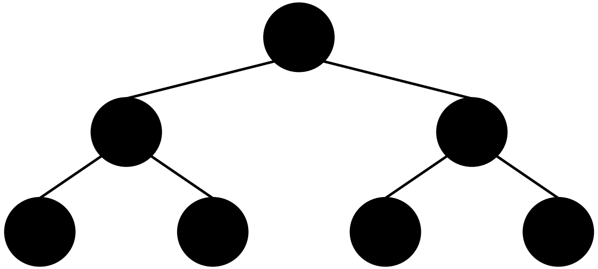
Degree of nodes

- **Degree (Tree)**
  is the maximum number of children
- **Degree (Node)**
  Its number of children

What is the degree of any leaf?



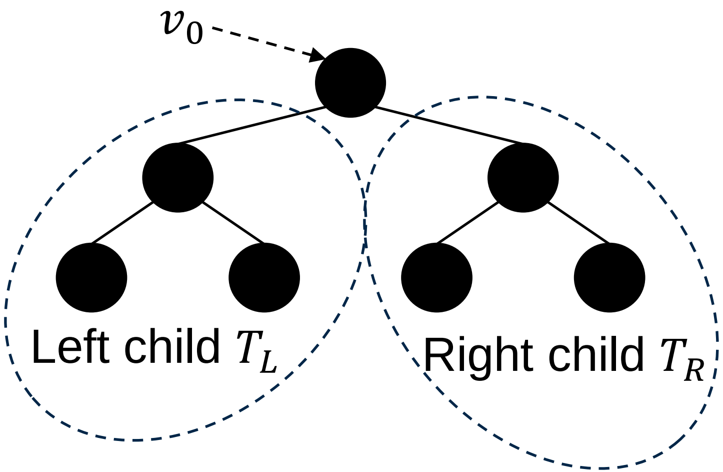
A tree with degree 3

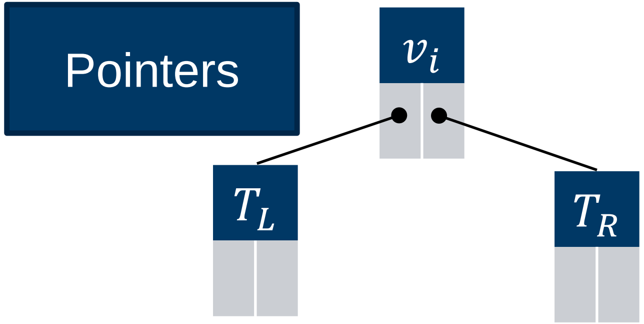


A tree with degree 2 (Binary Tree)

Binary Tree
Implementation

- Mathematically, a binary tree, can be defined as the triple
  - $T_B = (T_L, v_i, T_R)$
    - Where T is a tree
    - Where v is the root of the i^{th} subtree
    - Note that T_B can be empty
- A binary tree is always a binary tree when the two children are binary trees
- To implement binary trees, we use either structs with pointers from v_i to the children, arrays, or object-oriented programming





Binary Tree

Implementations

Only works with complete trees

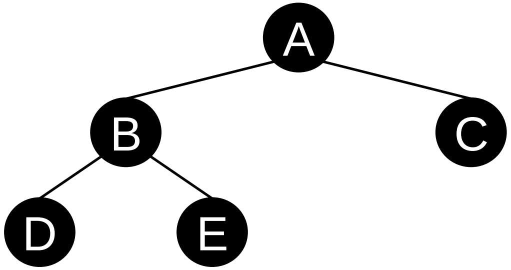
Two-dimensional array

A	B	C	D	E
0	1	2	3	4
1	3	-1	-1	-1
2	4	-1	-1	-1

Store the indices of the children in two dimensions

One-dimensional array

A	B	C	D	E
0	1	2	3	4



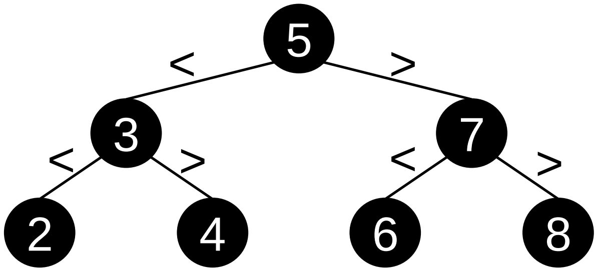
$$\mathrm{node}_i = \mathrm{array}[i]$$
$$\mathrm{left\ child}_i = \mathrm{array}[2i+1]$$
$$\mathrm{right\ child}_i = \mathrm{array}[2i+2]$$

Binary Search Tree

Divide and Conquer

- We already got to know two famous divide and conquer approach:

In the “average-case” binary search tree, we eliminate half of the search space in one operation



Read from left to right

- In a binary search tree, the elements are inserted using operators (e.g. “<“)
- Every element in a binary search tree is unique
- Every right child is larger than its parent and every left child is lower than its parent

Binary Search Tree

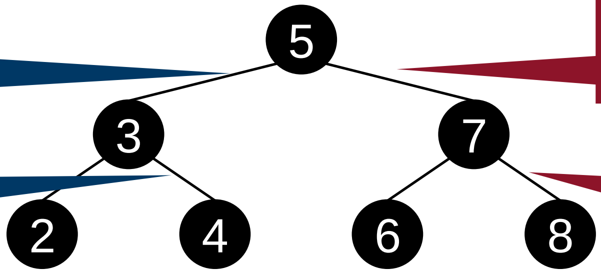
Finding in Binary Search Trees

Find the 4 in the tree

Go left because 4 < 5

Go right because 4 > 3

return true;



Find the 9 in the tree

Go right because 9 > 5

Go right because 9 > 7

return false;

Binary Search Trees

Creating Binary Search Trees

Classroom Task

Let’s create a binary search tree with the following values!
50, 30, 70, 60, 10, 20, 25, 40

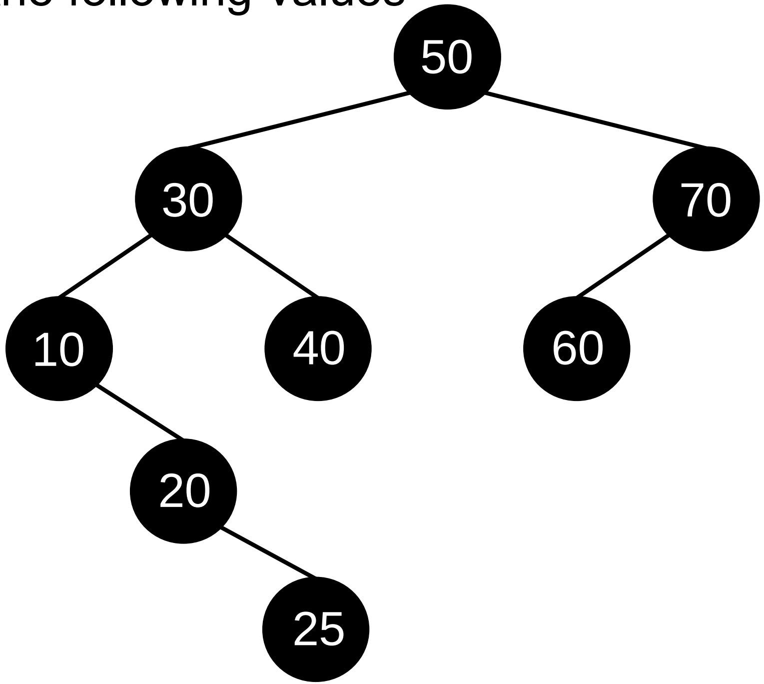
Binary Search Trees

Creating a binary search tree

Let's create a binary search tree with the following values
50, 30, 70, 60, 10, 20, 25, 40

- The first element we add is always the root.

50	30	70	60	10	20	25	40



Binary Search Trees

Creating binary search trees

Question

What is a problem when creating binary search trees?

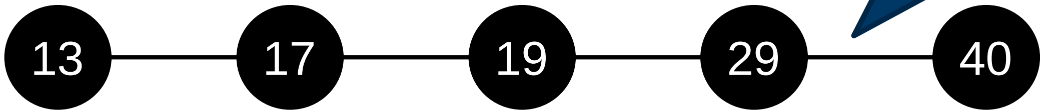
Binary Search Trees

Creating binary search trees

What is a problem when creating binary search trees?

Create the following binary tree and insert the elements in the given order:
13, 17, 19, 29, 40

We lost the usefulness of binary search trees by inserting sorted elements



=

Degenerate:
every parent has only one child ~ **Single Linked List**

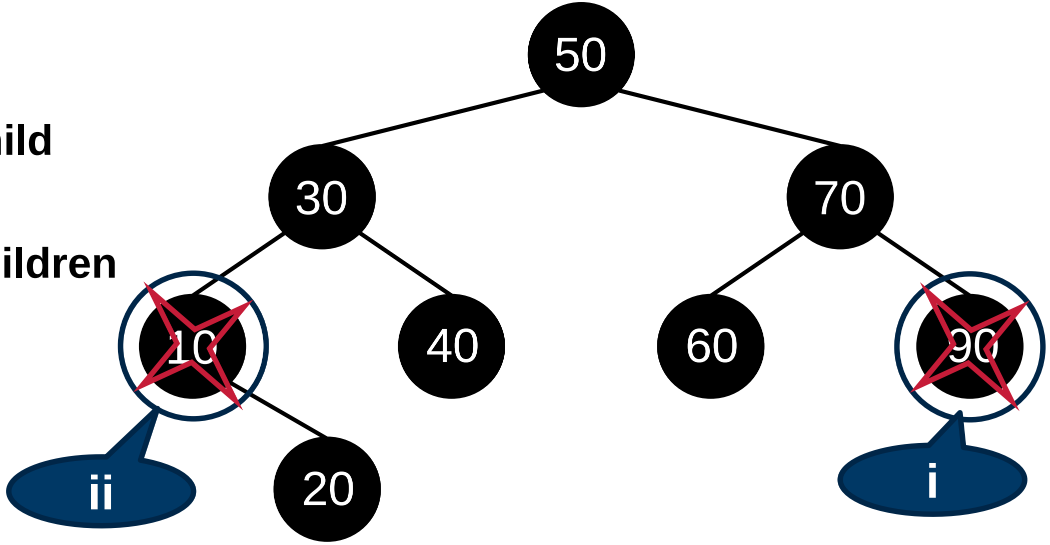
Binary Search Trees
Deleting Nodes

We need to distinguish three cases:

i. **Deleting a leaf**
   Just delete the node

ii. **Deleting a node with one child**
   Swap child to own position

iii. **Deleting a node with two children**



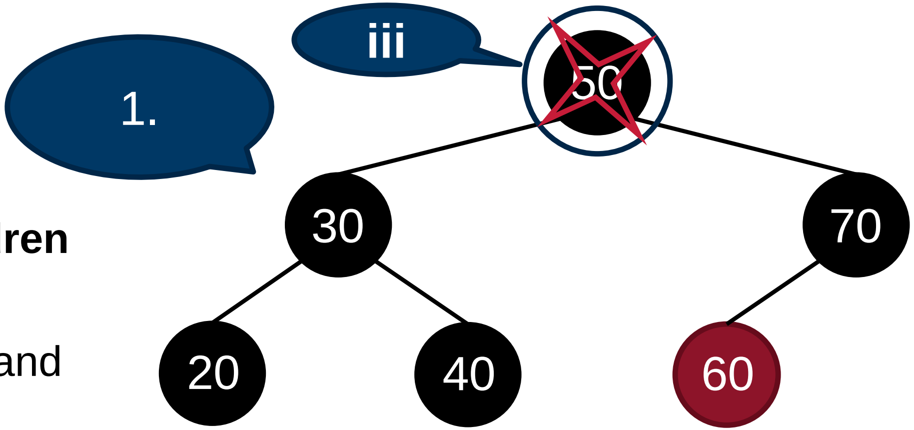
Binary Search Trees

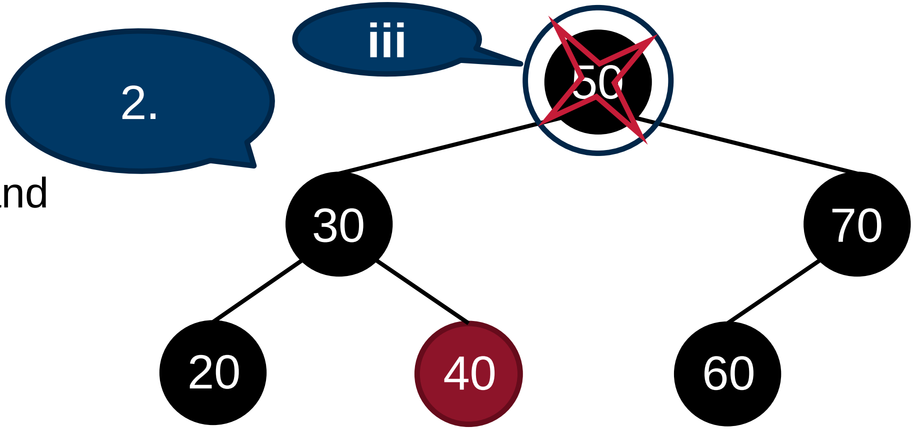
Deleting Nodes

iii Deleting a node with two children

Two strategies:

1. Find minimum of right subtree and replace with the deleted node





2. Find maximum of left subtree and replace with the deleted node

Binary Search Trees

Binary Tree Visualizer

Definition

Traversal: Recursive pseudocode and example

Preorder (5-3-1-2-4-7-6-9-8)

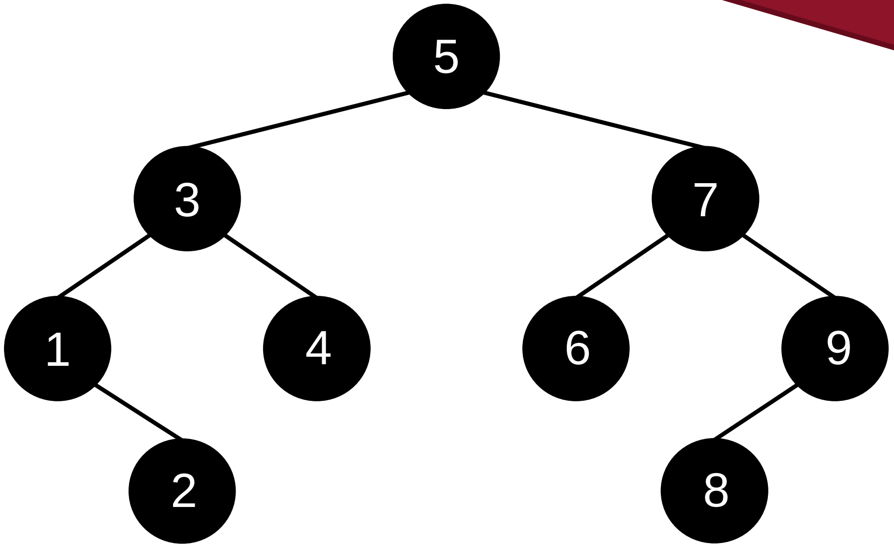
1. Print value
2. Go to left child
3. Go to right child

Inorder (1-2-3-4-5-6-7-8-9)

1. Go to left child
2. Print value
3. Go to right child

Postorder (2-1-4-3-6-8-9-7-5)

1. Go to left child
2. Go to right child
3. Print value



Inorder traversal with the printing operation prints the values in sorted order.

Binary (Search) Trees

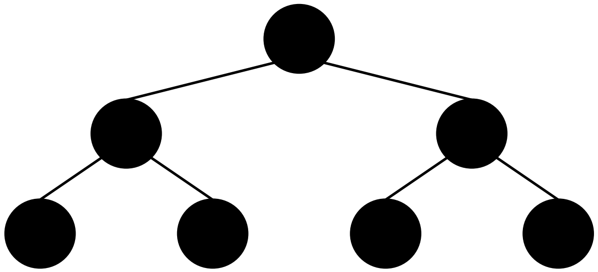
Types of trees

Definition

- **Full**
  - Every node has either 2 or 0 children
- **Complete**
  - A complete binary tree is filled at least down to the leaf level.
- **Balanced**
  - Height balanced:
    The difference of heights between a node's subtrees is $< \Delta\mathrm{h}$ (for us ∓ 1)
  - Fully balanced:
    The difference of nodes in each subtree is $< \Delta n$ (for us ∓ 1)

Perfect

Complete, Full and fully balanced



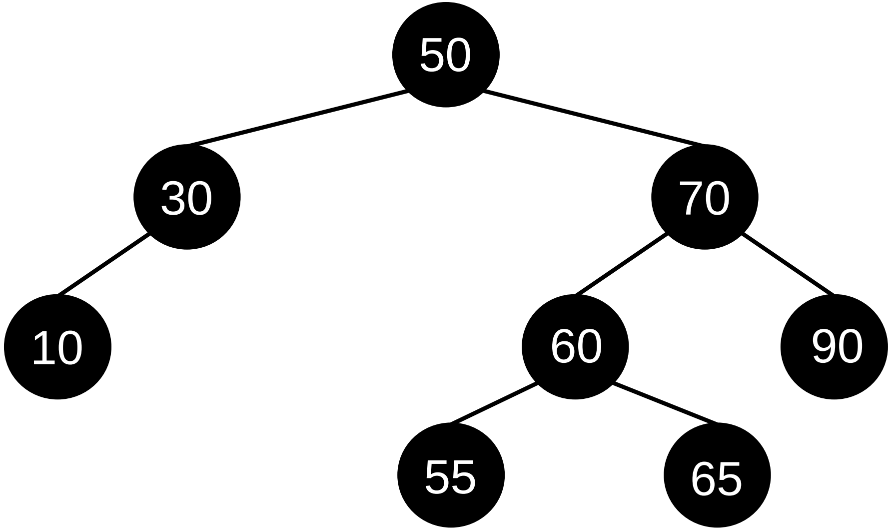
A perfect binary tree

Binary (Search) Trees

Types of binary (search) trees

Classroom Task

Classify the following tree!

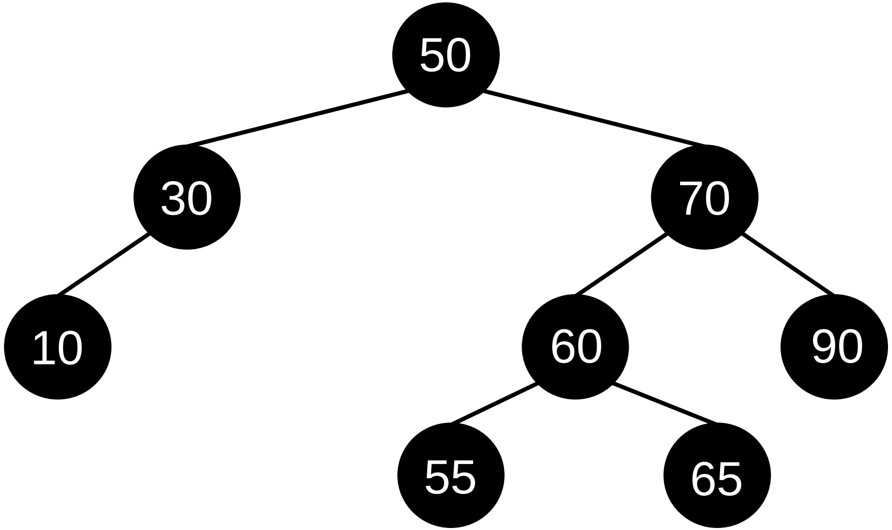


Binary (Search) Trees

Types of binary (search) trees

Classroom Task

Classify the following tree!



Neither full nor complete
but height balanced.

Binary (Search) Trees

Types of binary (search) trees

Question

Why do we like balanced trees better than unbalanced ones?

Binary Search Trees

Tree properties

Why do we like balanced trees better than unbalanced ones?

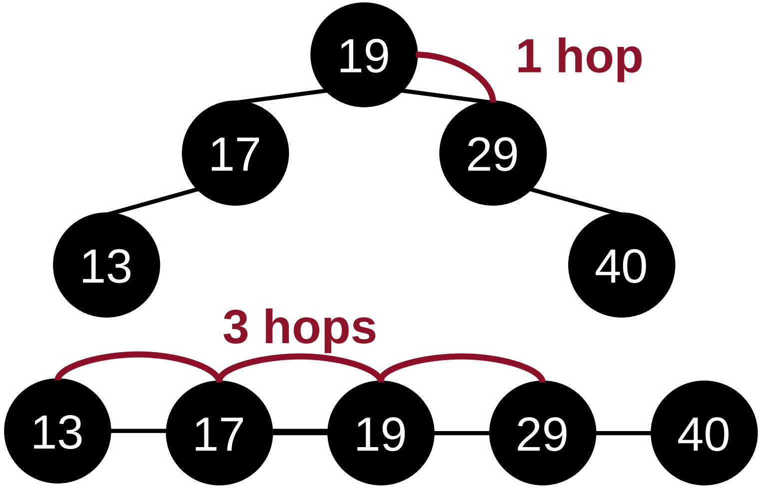
Find 29 in both data structures

Average runtime of searching:

$O(\log n)$

Vs.

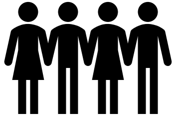
O(n)



AVL - Trees

Motivation

If balanced trees are so much better than unbalanced trees, why don’t we get self-balancing trees?



Ok then, let’s take it one step further with AVL-Trees!

AVL Trees

Adelson-**V**elsky and **L**andis (AVL) Trees

- Dates to 1962, and computer science
- The first self-balancing tree data structure
- Searching, Inserting and Deleting is $O(\log n)$ in the average and worst case
- **Idea**:
  Define a structural invariant. Every time one updates (delete or inserts) the tree check the invariant, and if required enforce it.

Animation

AVL Trees

Definition

Definition

The Invariant that AVL trees enforce is as follows:

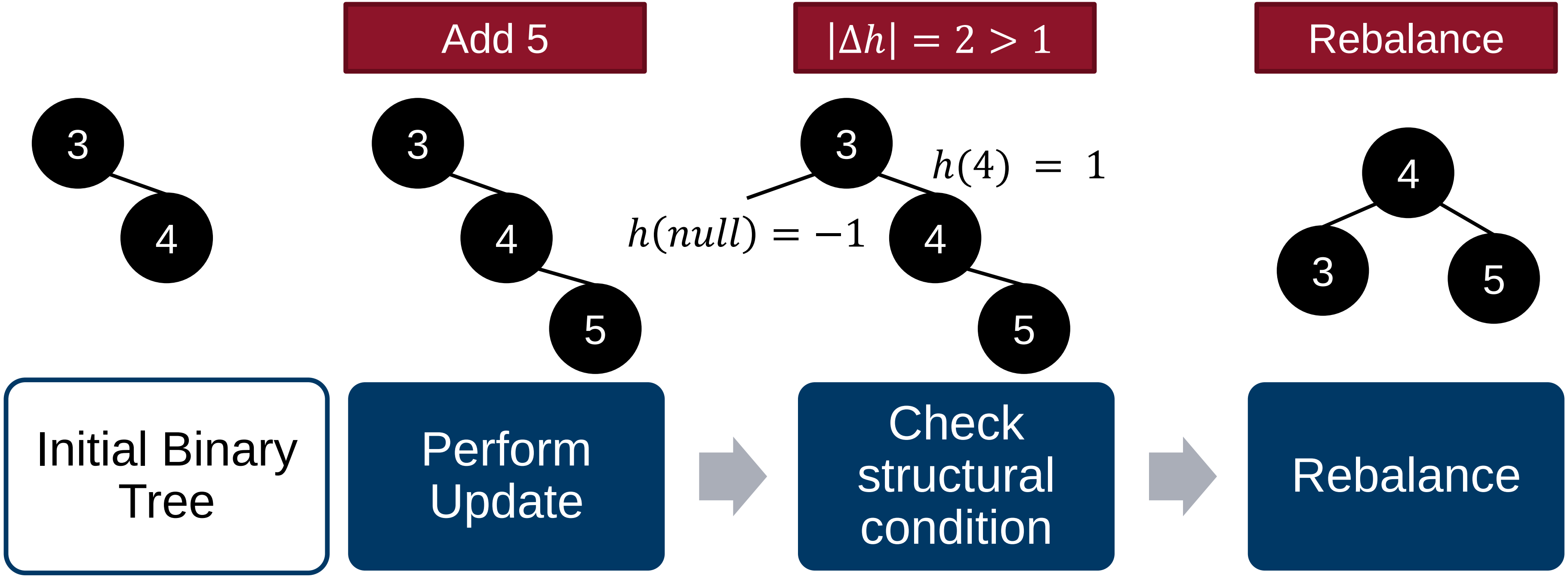
Let v be any node in a binary search tree and $h(v)$ be the function to determine its height.
The height of both children $v.leftchild$ and $v.rightchild$ differ by 1 at most.

Structural Invariant (AVL):
$|h(v.left) - h(v.right)| \leq 1$

Whenever an update violates the AVL invariant, the tree "**rebalances**".

AVL Trees

Update – Insertion and Deletion



AVL Trees

Short

More on AVL Trees in an upcoming “Short”!

Learning Objectives

After this lecture you can …

… define and distinguish graphs and trees

… classify binary trees

… Insert elements in binary (search) trees

… Delete elements in binary (search) trees

… set up data structures to store graphs and trees

… describe self-balancing binary trees

Revisited

Exam Date

Exam Date:
19.02.2020

Mock Exam:
Prof. Harth – Q & A Session

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF BUSINESS
AND ECONOMICS

Chair of Digital Industrial Service Systems

Prof. Dr. Martin Matzner

Friedrich-Alexander-Universität Erlangen-Nürnberg

School of Business and Economics

wiwi-is-kontakt@fau.de

twitter.com/ismama

References

- Euler, Leonhard. "From the Problem of the Seven Bridges of Königsberg." In *Classics of Mathematics*, Ronald Calinger, ed. Englewood Cliffs, NJ: Prentice Hall, 1995
- Algorithmen und Datenstrukturen – Prof. Dr. Pflaum, FAU
- Meinel, C., Mundhenk, M.: Mathematische Grundlagen der Informatik. 2006
- Cormen, T. H., Leiserson, C. E., Rivest, R. L. , Stein, C.: Introduction to Algorithms. 3rd Ed. (2009)
- Adelson-Velsky, Georgy; Landis, Evgenii (1962). "An algorithm for the organization of information". Proceedings of the USSR Academy of Sciences (in Russian). 146: 263–266. English translation by Myron J. Ricci.