Recap

Maps

- Key-Value-Storage, used frequently!
- Interface: `Map<K, V>`
- Implementation: `HashMap<K, V>`
  - Keys and values are stored in pairs
  - Pairs in which the keys have the same `hashCode()` end up together in a linked list

Recursion

MIEJSCE ZAREZERWOWANE

MyLinkedList with Recursive Implementation of size()

```
public class MyLinkedList {

  public int size() { return prefirst.size() - 1; }

  // ...

  private class ListElement {
    T value;
    ListElement next;

    ListElement(T value) { this.value = value; }

    public int size() {
      if (next == null)
        return 1;
      return next.size() + 1;
    }
  }
}
```

MyLinkedList with Recursive Implementation of size()

```
public class MyLinkedList {

  public int size() { return prefirst.size() - 1; }

  // ...

  private class ListElement {
    T value;
    ListElement next;

    ListElement(T value) { this.value = value; }

    public int size() {
      if (next == null)
        return 1;
      return next.size() + 1;
    }
  }
}
```

size() is recursive, because it may call itself

Session 8: Recursion, Part 2

Fortgeschrittene Programmierung (Java 2)

Nils Reiter
nils.reiter@uni-koeln.de

June 6, 2022

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

DROSTE'S CACAO
DROSTE
netto
1/10 K.G.

Recursion

- Recursion (adjective: recursive) occurs when a thing is defined in terms of itself or of its type W Recursion

Recursion

- Recursion (adjective: recursive) occurs when a thing is defined in terms of itself or of its type W Recursion

Natural numbers

- 0 is a natural number
- If n is a natural number, $n + 1$ is also a natural number

Recursion

$3! = 3 \cdot 2 \cdot 1$

- Recursion (adjective: recursive) occurs when a thing is defined in terms of itself or of its type

W Recursion

Definition of the factorial

Non-recursive definition

- $n! = \prod_{i=1}^{n} i$

Recursive definition

- $0! = 1$ $\underline{\text{(base case)}}$
- $\boxed{n!} = \underline{n} \times \boxed{(n-1)!}$ $\underline{\text{(recursion step)}}$

Recursion

- Recursion (adjective: recursive) occurs when a thing is defined in terms of itself or of its type

W Recursion

Some German Sentences

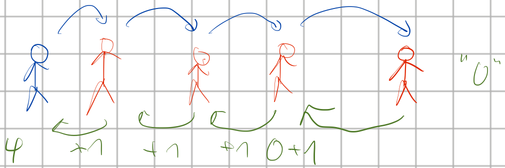
- A main clause consists of a noun phrase and a verb phrase (base case)
  - E.g., “Maria schläft”
- A sentence consists of two main clauses, joined by “denn” (recursion step)
  - E.g., “Maria schläft denn Hans isst denn der Pizzabote war da.”

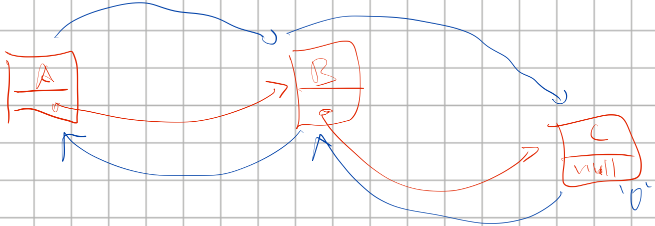
Recursion

- Two components
  - Recursion step: How to make one additional step
  - Base case(s): When and how to stop doing additional steps

Example

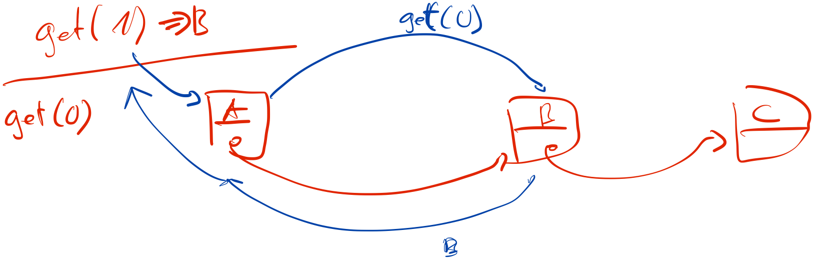
- Recursion step (for person A)
  - Ask the next person (B) how long this queue is
  - The queue length for A is one more than for B
- Base case
  - The first person knows how long the queue is


"0"
4
+1
+1
+1
0+1


A
0
B
C
null
'0'

demo

Implementation of `get(int)` in linked list



Recursion

- Two relevant areas in programming
  - Recursive data structures – how we store things
  - Recursive algorithms – how we process things
- Usually, one needs recursive algorithms to deal with recursive data structures

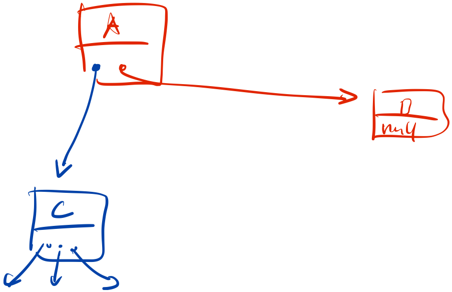
Section 1

Recursive Data Structures

Recursive Data Structures

- A new kind of data structure: Trees
- Represents hierarchical situations
  - File systems
  - HTML/XML nodes
  - Company hierarchies

Recursive Data Structures



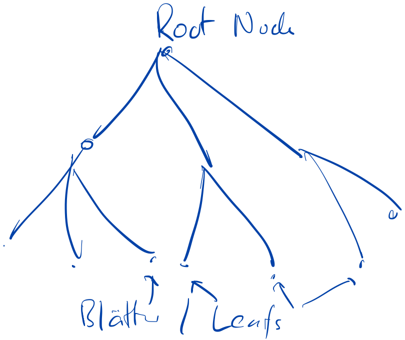
- A new kind of data structure: Trees
- Represents hierarchical situations
  - File systems
  - HTML/XML nodes
  - Company hierarchies

Recursive Definition of a Tree

A tree is a **pair** consisting of some value and a **set** of children, which are **tree**s.

Tree Terminology

- Parent/child: The super- or subordinate tree
  - Each tree has 0 or 1 parents, and 0 or more children
- Root tree: The tree with 0 parents
- Leaf tree: Any tree that has 0 children



Tree Terminology

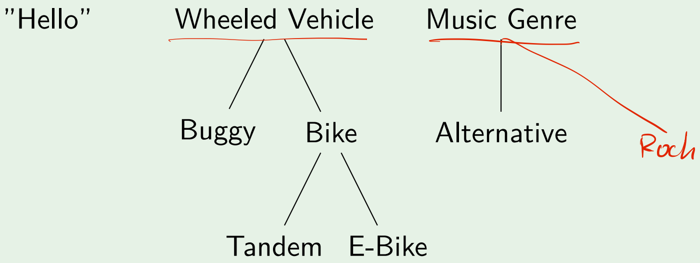
- Parent/child: The super- or subordinate tree
  - Each tree has 0 or 1 parents, and 0 or more children
- Root tree: The tree with 0 parents
- Leaf tree: Any tree that has 0 children
- Metrics
  - Depth: The maximal number of steps between root and a leaf
  - Size: Number of trees

Recursive Data Structures

Trees

Examples

All these are trees:



demo

Creation of a data structure `Tree<T>`

Recursive Algorithms

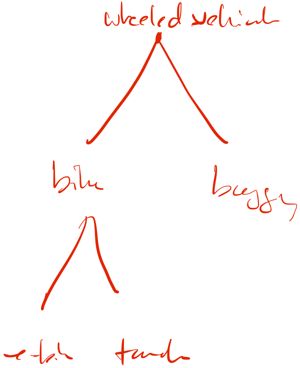
- Recursive algorithms to take recursive data structure into account
- Linked list context
  - `size()`
    - Single base case
    - During return, size is calculated

Recursive Algorithms

- Recursive algorithms to take recursive data structure into account
- Linked list context
  - `size()`
    - Single base case
    - During return, size is calculated
  - `get(int)`
    - Two base cases: End of list reached and n equals 0
    - Return value is passed through unchanged

Recursive Algorithms

- Recursive algorithms to take recursive data structure into account
- Linked list context
  - `size()`
    - Single base case
    - During return, size is calculated
  - `get(int)`
    - Two base cases: End of list reached and n equals 0
    - Return value is passed through unchanged
- Oerations for the tree
  - Size: Total number of trees
  - Depth: Maximal number of trees between root and one leaf
  - Both require “visiting” each tree and doing something – a “walk”



demo

Visit each item in the tree and print it

wheeled vehicle

bike

tandem

e-bike

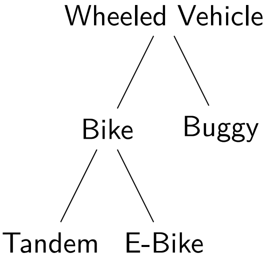
buggy

Depth-First Search vs. Breadth-First Search

- Two strategies of iterating over all elements of a tree
  - Concerns the order in which elements are visited
- Depth-first search: Descend first before going to a sibling
- Breadth-first search: First go over all siblings, then descend

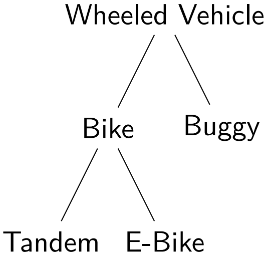
Depth-First Search vs. Breadth-First Search

- Two strategies of iterating over all elements of a tree
    - Concerns the order in which elements are visited
- Depth-first search: Descend first before going to a sibling
- Breadth-first search: First go over all siblings, then descend

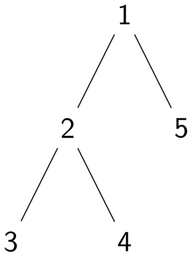


Depth-First Search vs. Breadth-First Search

- Two strategies of iterating over all elements of a tree
  - Concerns the order in which elements are visited
- Depth-first search: Descend first before going to a sibling
- Breadth-first search: First go over all siblings, then descend

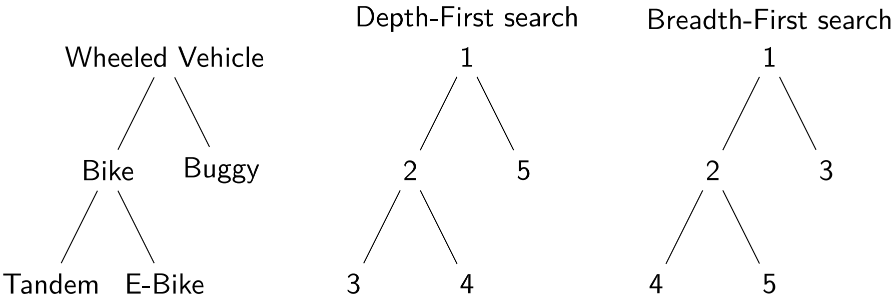


Depth-First search



Depth-First Search vs. Breadth-First Search

- Two strategies of iterating over all elements of a tree
  - Concerns the order in which elements are visited
- Depth-first search: Descend first before going to a sibling
- Breadth-first search: First go over all siblings, then descend



Exercise

https://github.com/idh-cologne-java-2/exercise-08