Algorithms and Data Structures
- Lesson 1 -

Michael Schwarzkopf
https://www.uni-weimar.de/de/medien/professuren/medieninformatik/grafische-datenverarbeitung

Bauhaus-University Weimar

May 2, 2018

Overview

...of things you should definetely know about if you want a very good grade

- Definition of Algorithms & Data Structures
- Arrays
- Linked Lists
- Stacks
- Queues
- Graphs
- Trees
- Complexity
- Sorting ← Let's talk about this next time.
- ...

Exact definitions are important!

Algorithms & Data Structures

"A finite sequence of well defined instructions which, given an input, will produce in a finite time a required output"

"Structured memory space and a set of operations, or actions, for accessing and manipulating such memory space"

Exact definitions are important!

Algorithms

- Finitely many Instructions but also finitely many steps!

"A finite sequence of well defined instructions which, given an input, will produce in a finite time a required output"

Exact definitions are important!

Algorithms

"A finite sequence of well defined instructions which, given an input, will produce in a finite time a required output"

- Finitely many Instructions but also finitely many steps

- No ambiguous sequences

Exact definitions are important!

Algorithms

"A finite sequence of well defined instructions which, given an input, will produce in a finite time a required output"

- Finitely many Instructions but also finitely many steps

- No ambiguous sequences

- Input and Output

Exact definitions are important!

Algorithms

"A finite sequence of well defined instructions which, given an input, will produce in a finite time a required output"

- Finitely many Instructions but also finitely many steps

- No ambiguous sequences

- Input and Output

- Should be correct → does what we want

Exact definitions are important!

- Information which is located somewhere and connected somehow

Data Structures

"Structured memory space and a set of operations, or actions, for accessing and manipulating such memory space"

Exact definitions are important!

- Information which is located somewhere and connected somehow

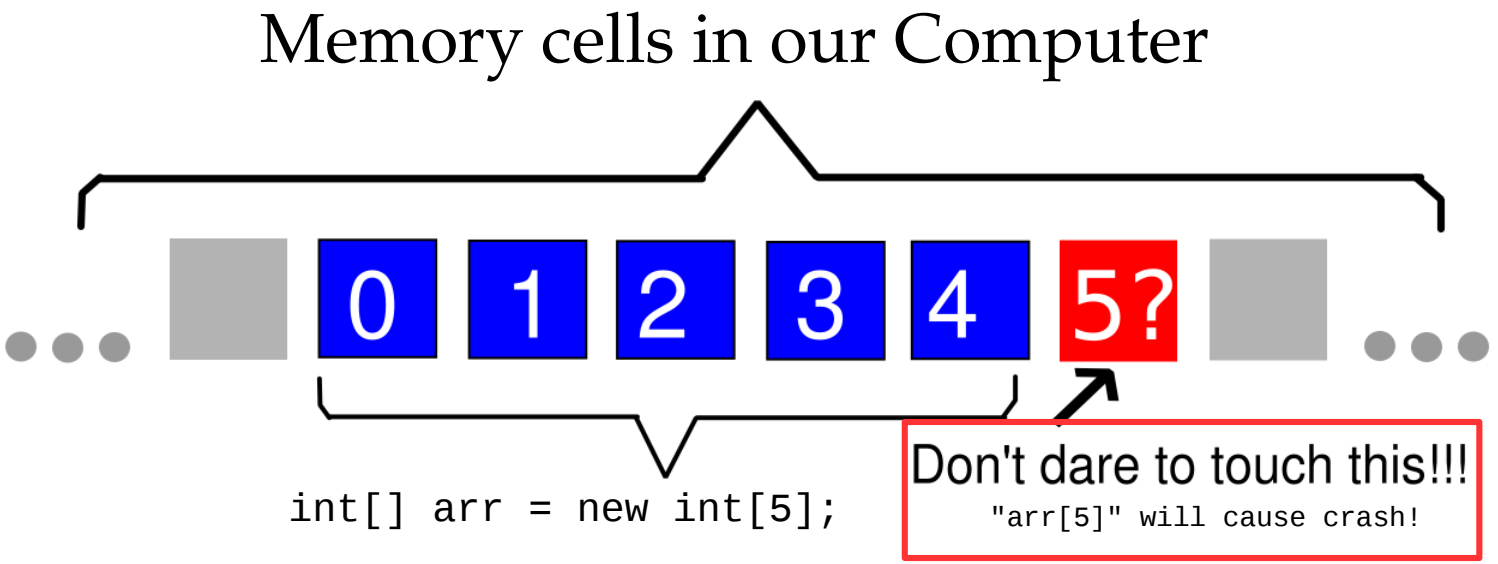
- Operations to locate ceriain information to read / overwrite / delete

Data Structures

"Structured memory space and a set of operations, or actions, for accessing and manipulating such memory space"

Arrays

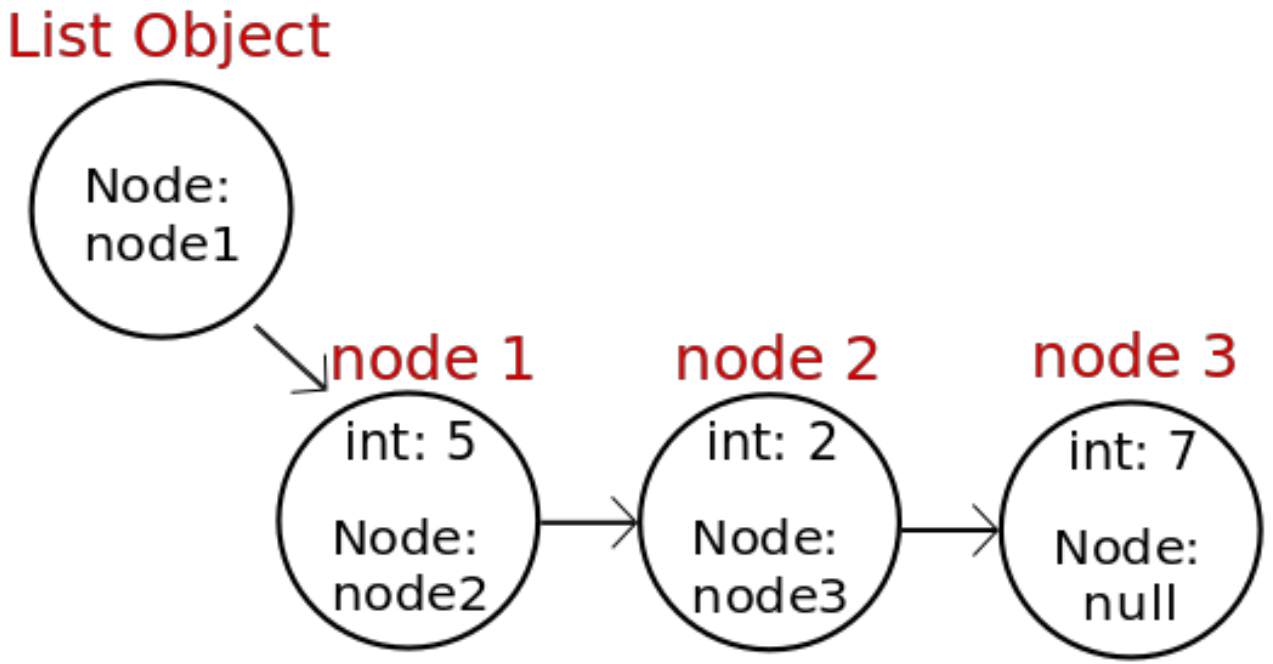
- Most simple structure but with little feedom
  → Static size once daclared
  → Only one datatype



Linked Lists

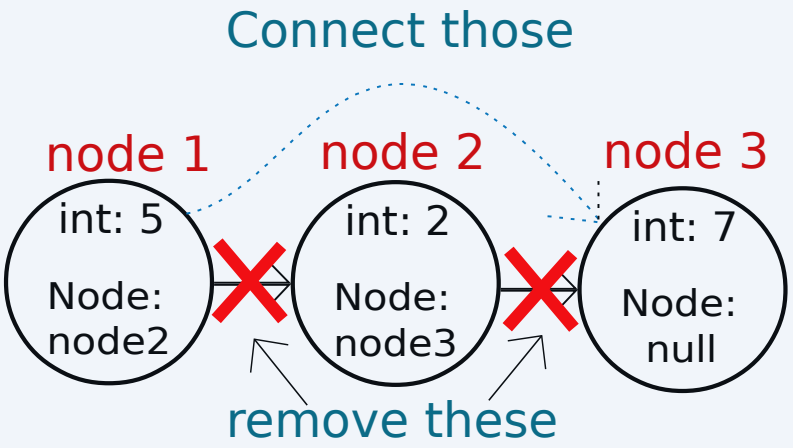
- Consist of "Nodes" which hold data + location of next Node
  → Can be anywhere in Memory
  → Nodes are easy to add / remove

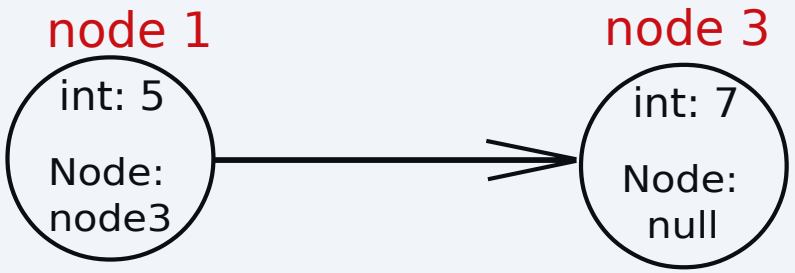
  Compare: removing Element in Array → cell of unusable Memory,
  adding an Element in Array → need to reallocate Memory.



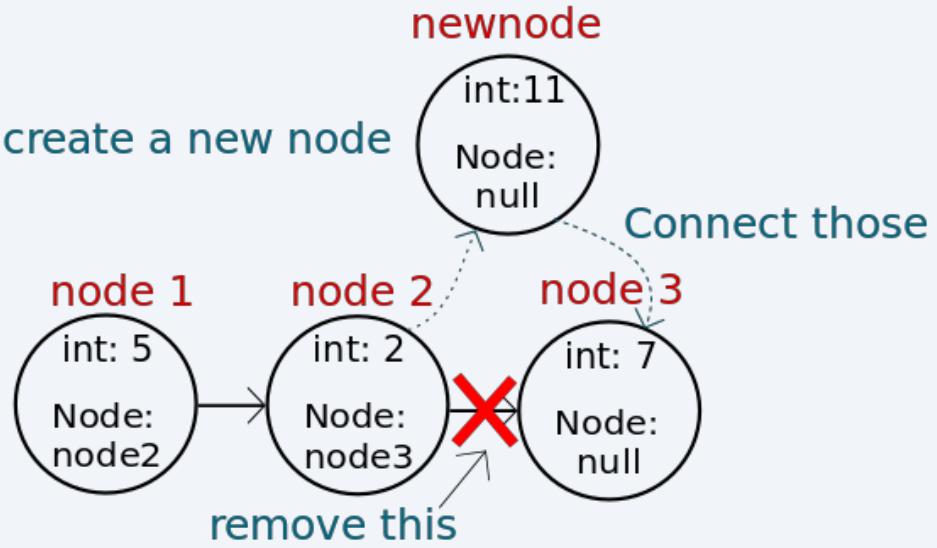
Linked Lists

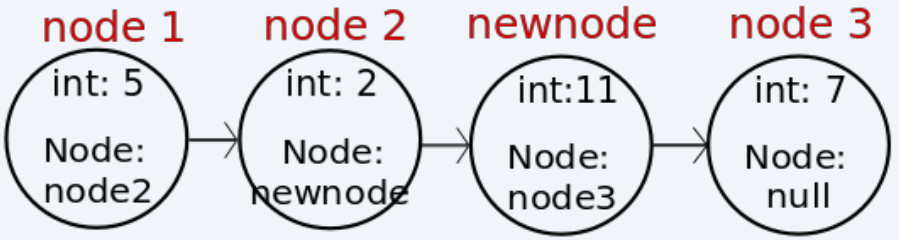
Removing nodes:



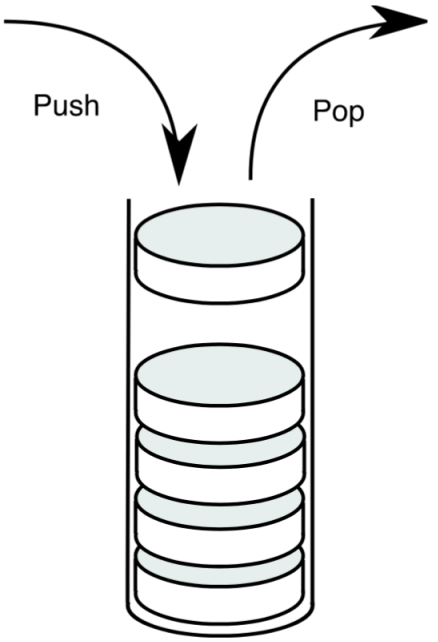


Adding nodes:





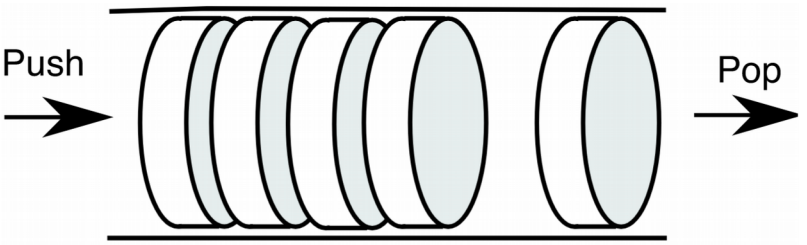
Stacks



- Last – in – First – out (LiFo)
- Basically a List with constraints:
  - Only first element can be read
  - The same is true for adding and removing nodes

- But what can they be usefull for?

Queues



- First – in – First – out (FiFo)
- Basically a List with constraints:
  - Only first element can be read and removed
  - Adding nodes only possible at the very end*
- And what can they be usefull for?

*Just in theory as you had to manipulate the last element to connect it to a new one

Graphs

- Def: Ordered pair of two sets:

  $$(V , E)$$

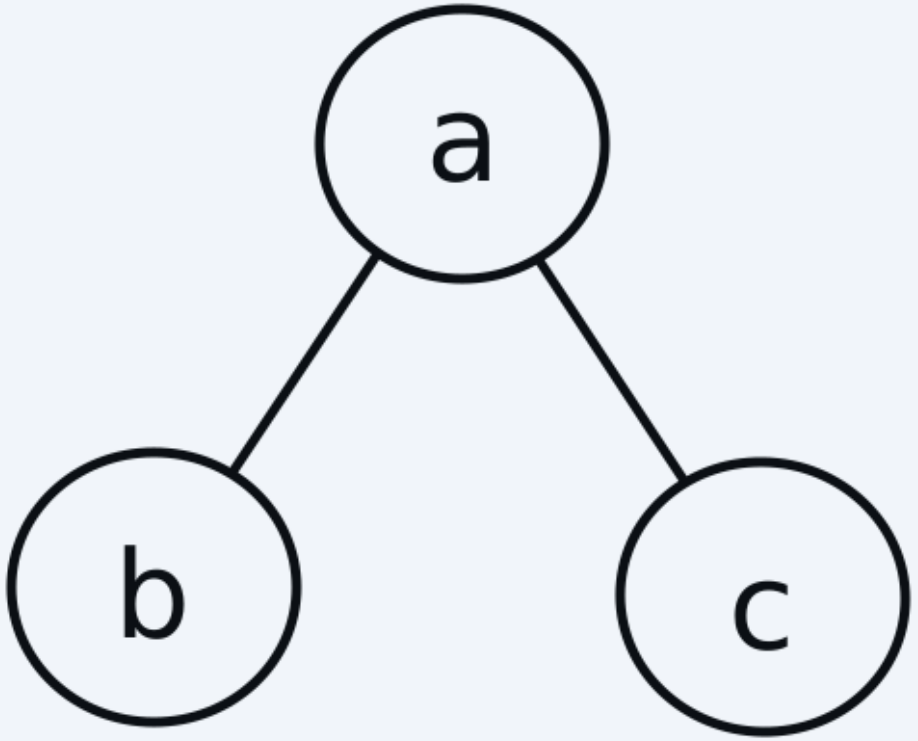
  Where V is the set of Verices while E denotes the set of Edges connecting two Vertices (In general).

- So what does such a Graph look like?

Graphs

For example: G = ({a,b,c} , {(a,b) , (a,c)})

Visual representation:

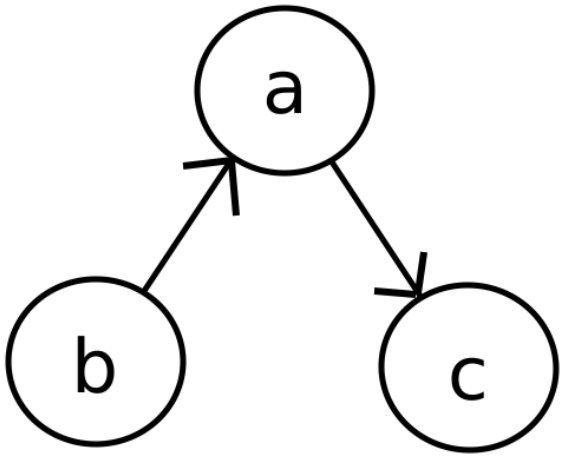


Adjacency Matrix:

	a	b	c
a	0	1	1
b	1	0	0
c	1	0	0

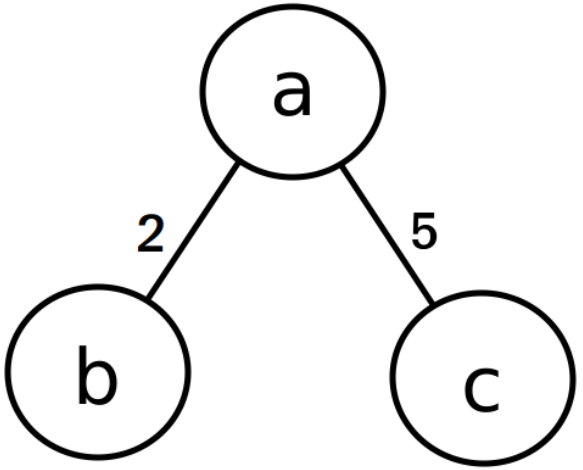
Graphs

Directed Graph:



	a	b	c
a	0	0	1
b	1	0	0
c	0	0	0

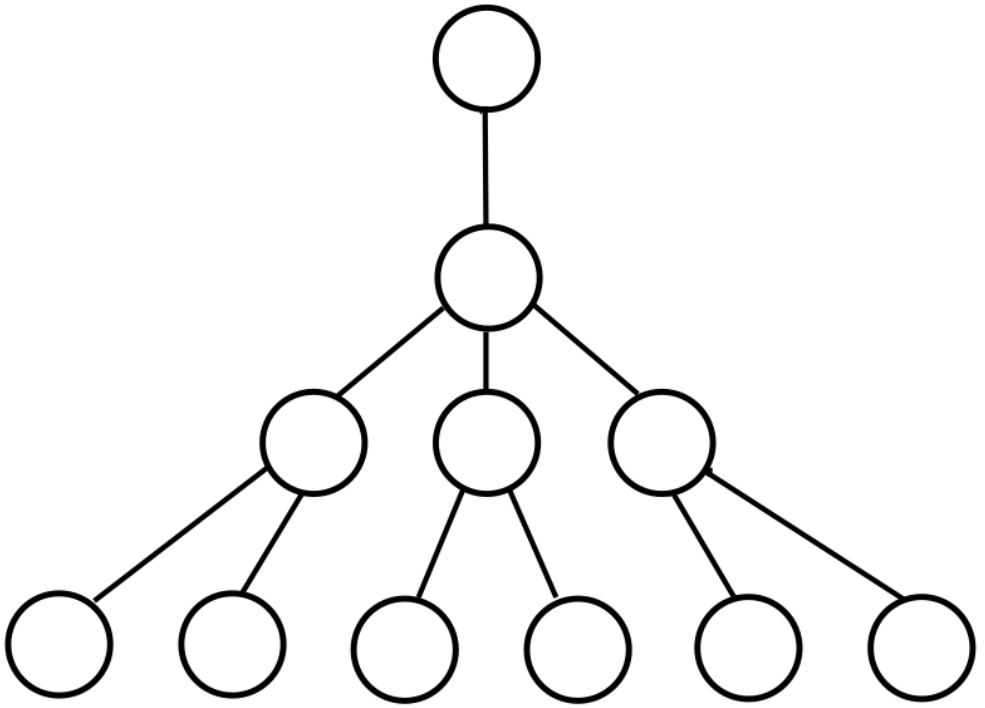
Weighted Graph:



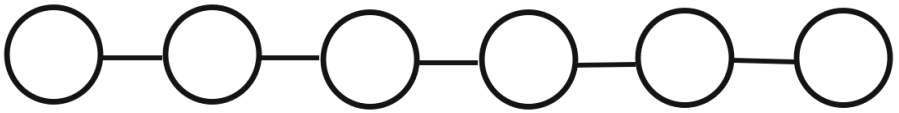
	a	b	c
a	0	2	0
b	2	0	5
c	0	5	0

Trees

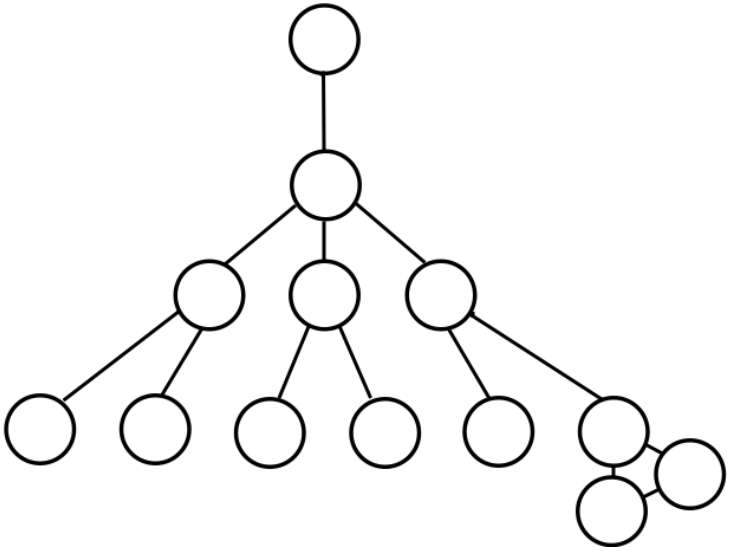
Remember: Graphs without Circles



Still a tree by definition:

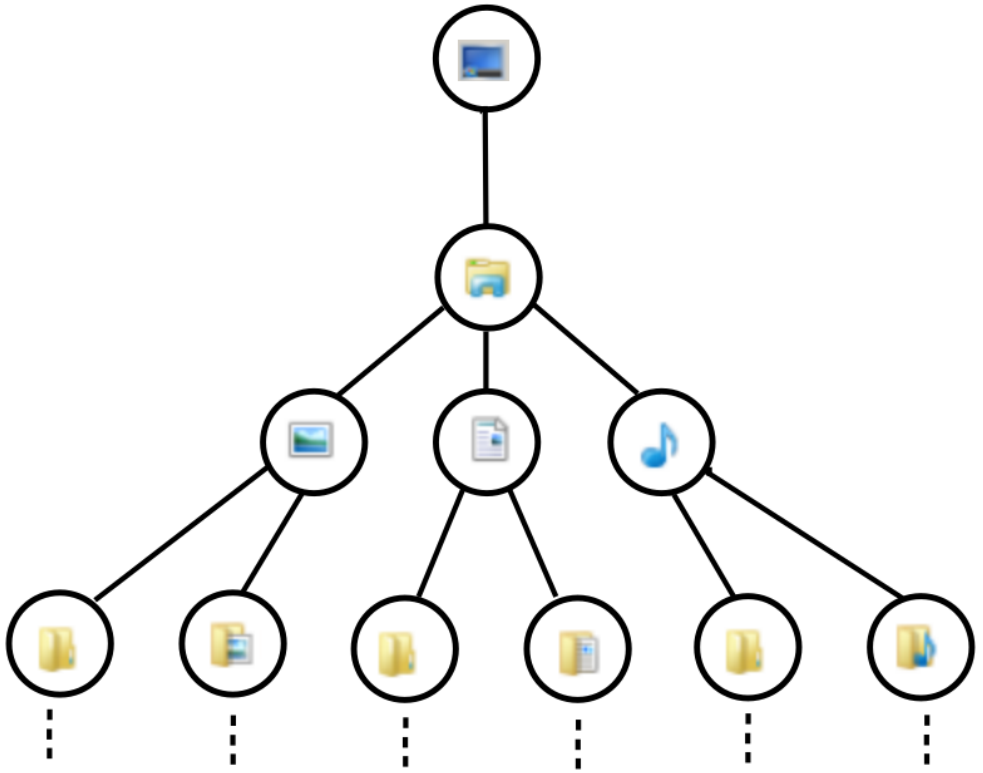


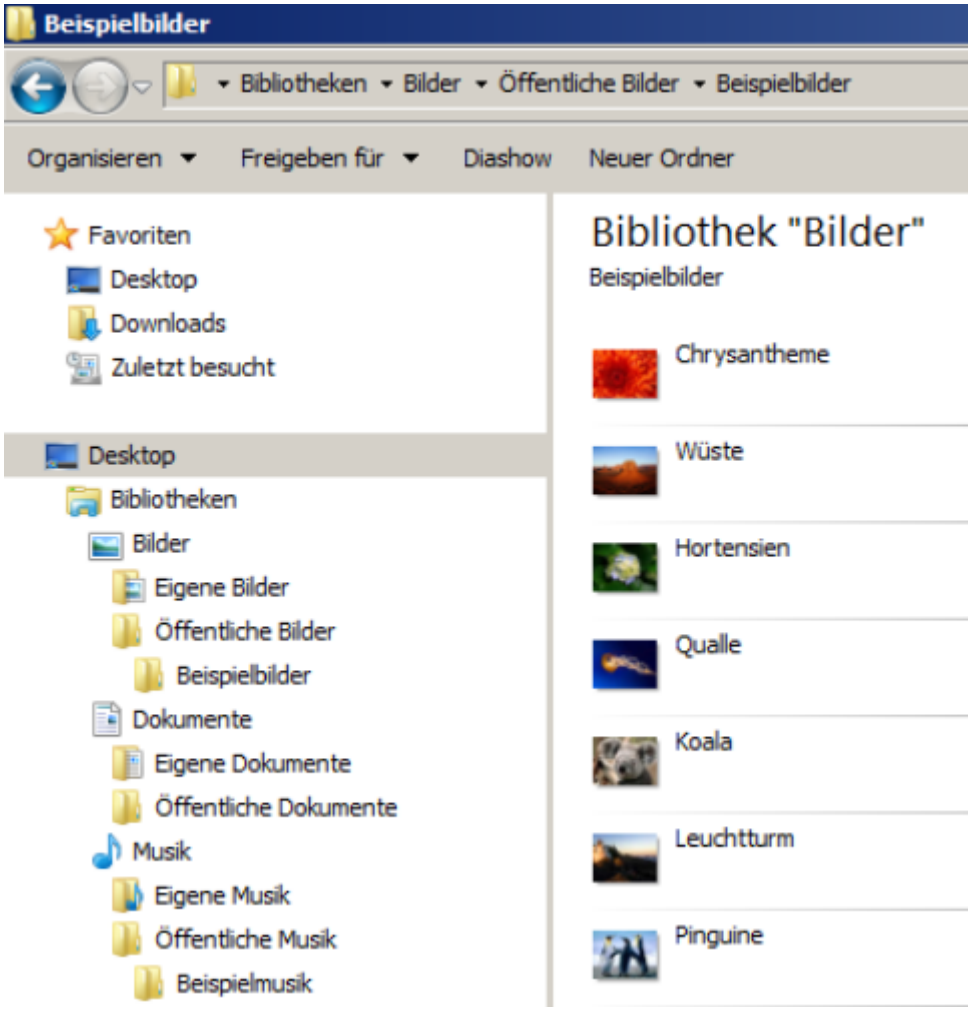
Not a tree anymore:



Trees

Practical example: Data in Operating Systems





Trees

Based on our example:
What if you search for certain documents?

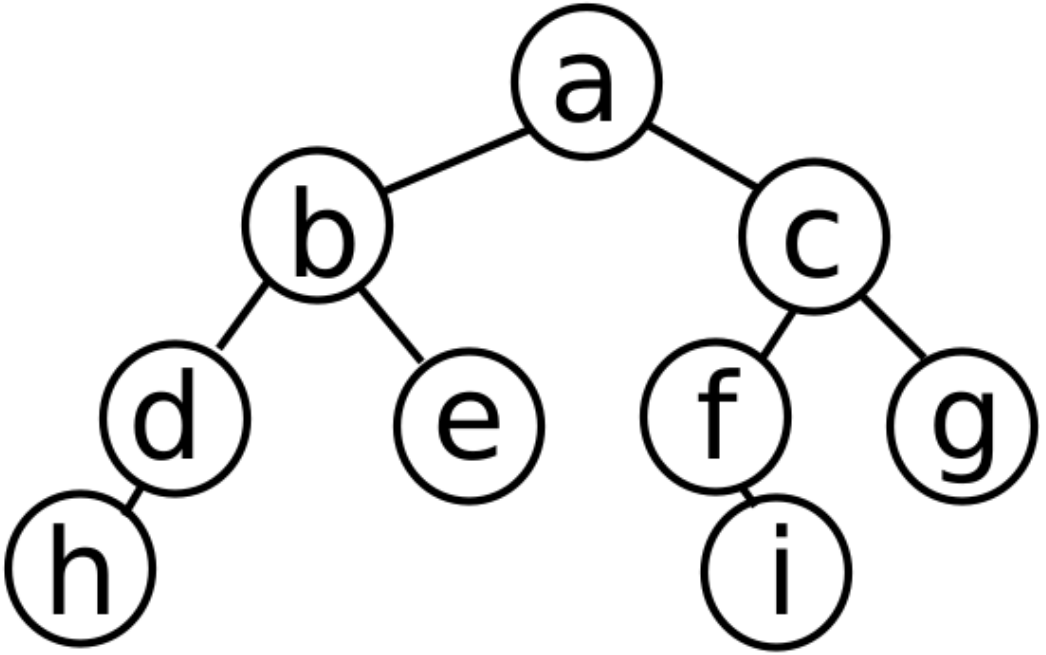
→ You need to *traverse* the tree:

- Pre – Order
- Post – Order
- In – Order
- Level – Order

Very good explanation if you need to repeat:

https://en.wikipedia.org/wiki/Tree_traversal

Traversal of Trees



- Pre – Order a, b, d, h, e, c, f, i, g
- Post – Order h, d, e, b, i, f, g, c, a
- In – Order h, d, b, e, a, f, i, c, g
- Level – Order a, b, c, d, e, f, g, h, i

Some use cases

List: Representing and adding polynomials

Stack: Recognizing palindromes

Queue: Orders from a fast Device to a slow one like a printer. → Data has to „wait“

Just think about some more!

Assignment

Huffman Code

- Lossless Data Compression
- Still relevant in later courses like Coding Theory
- Uses Trees
- Well documented on the web

Assignment

By the way, how would you encode

- Lossless? Data Compression
- Still relevant in later courses like Coding Theory
- Uses Trees
- Well documented on the web

Assignment

By the way, how would you encode

Lossless?

$L \rightarrow 00, O \rightarrow 01, S \rightarrow 10, E \rightarrow 11$

00 01 10 10 00 11 10 10

16 Bit.

Assignment

You can do it this way but you will

$$L \rightarrow 00, O \rightarrow 01, S \rightarrow 10, E \rightarrow 11$$

more space in Memory than necessary

Assignment

Frequently represented letter gets shorter Code

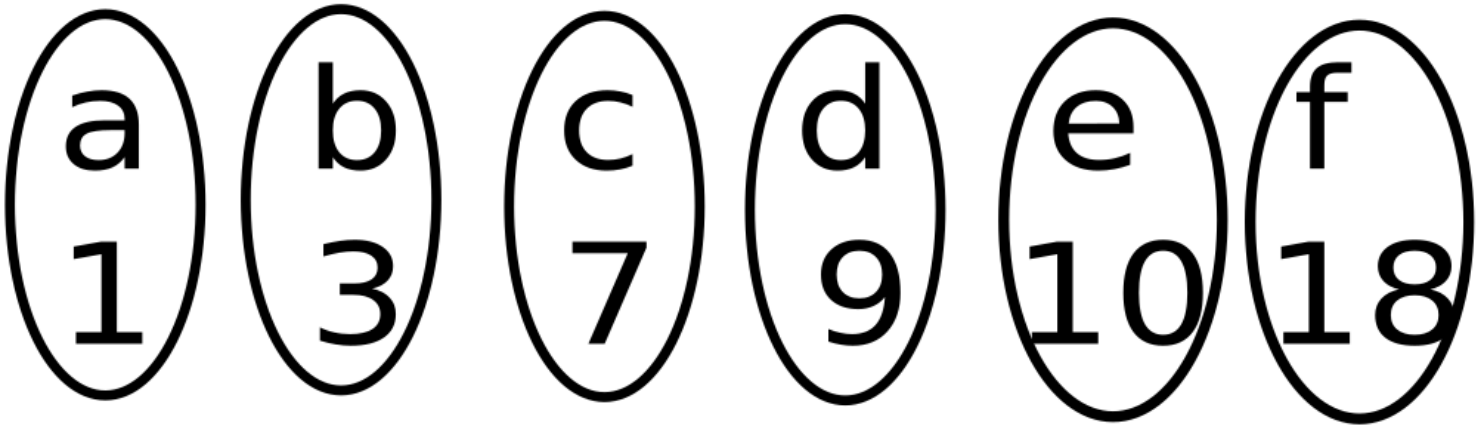
Letter	Frequency	Code
s	4	1
l	2	01
o	1	001
e	1	000

→ 0100 1110 1000 11 – only 14 Bit!

Assignment

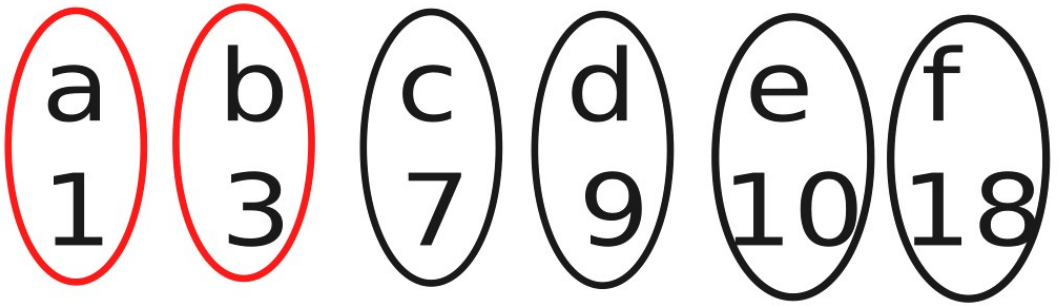
How do we achieve this Coding scheme?

- Input:
  - Letter
  - Corresponding Frequency
- Treat every node as a Subtree:

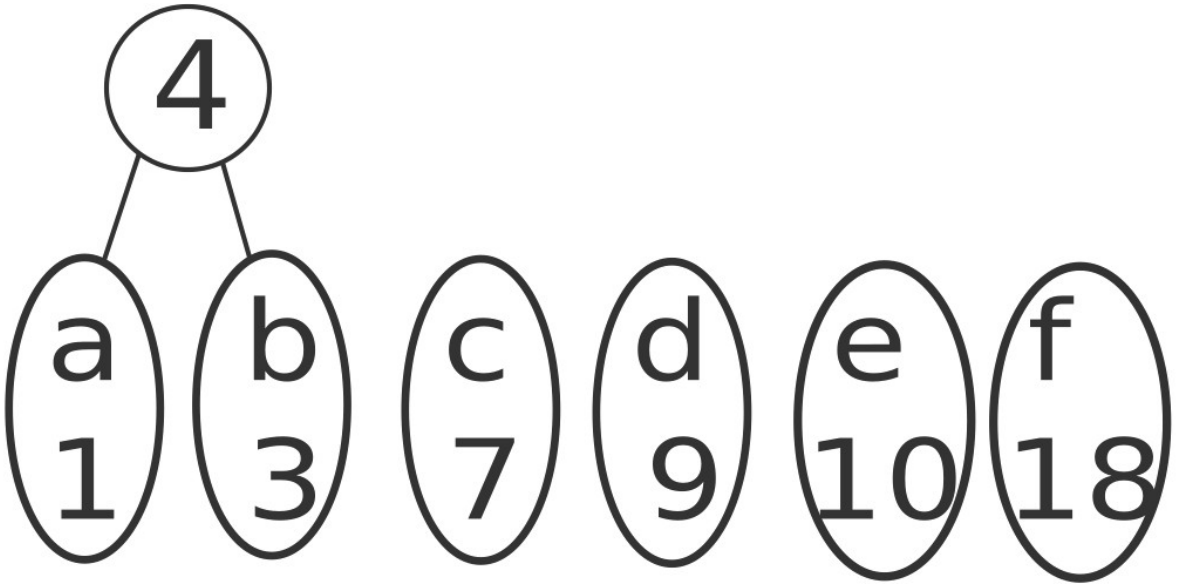


Assignment

Search the two smallest roots

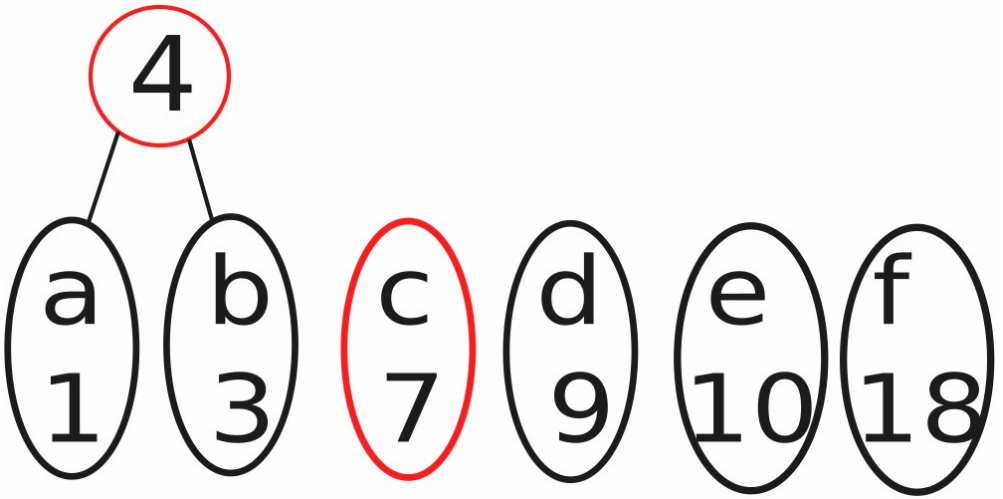


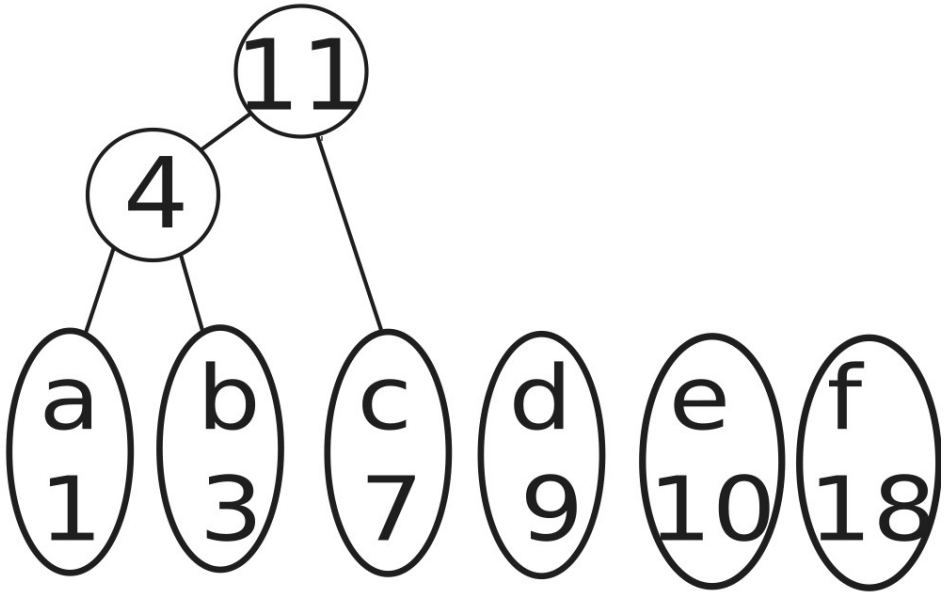
Connect to a new Tree and add frequencies



Assignment

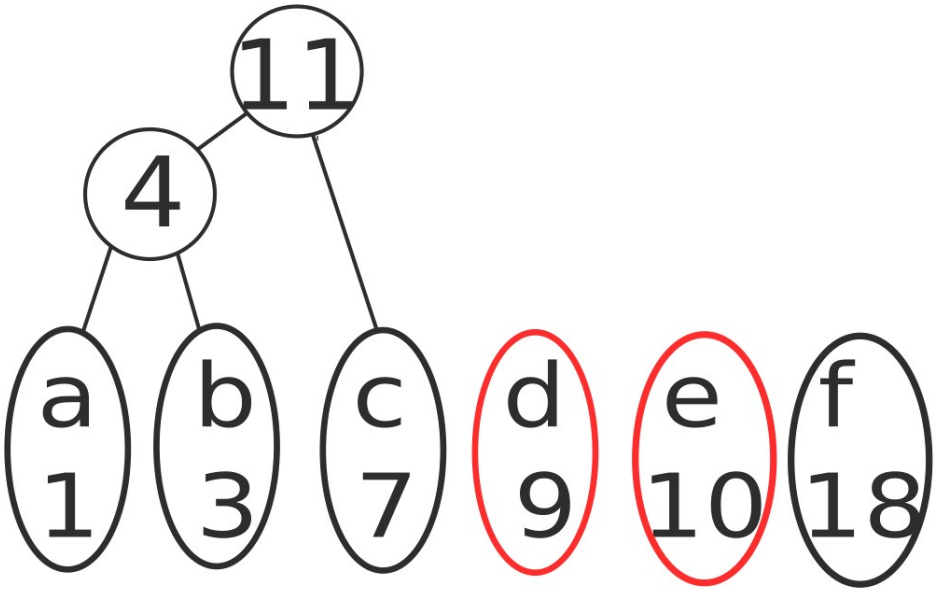
Do it rekursively on the new Data

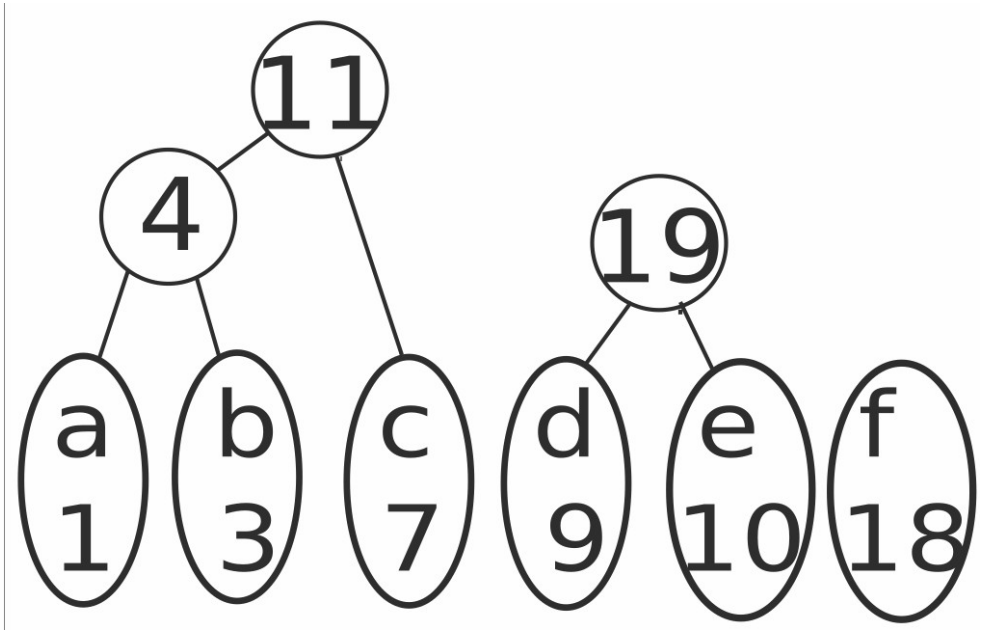




Assignment

And so on...



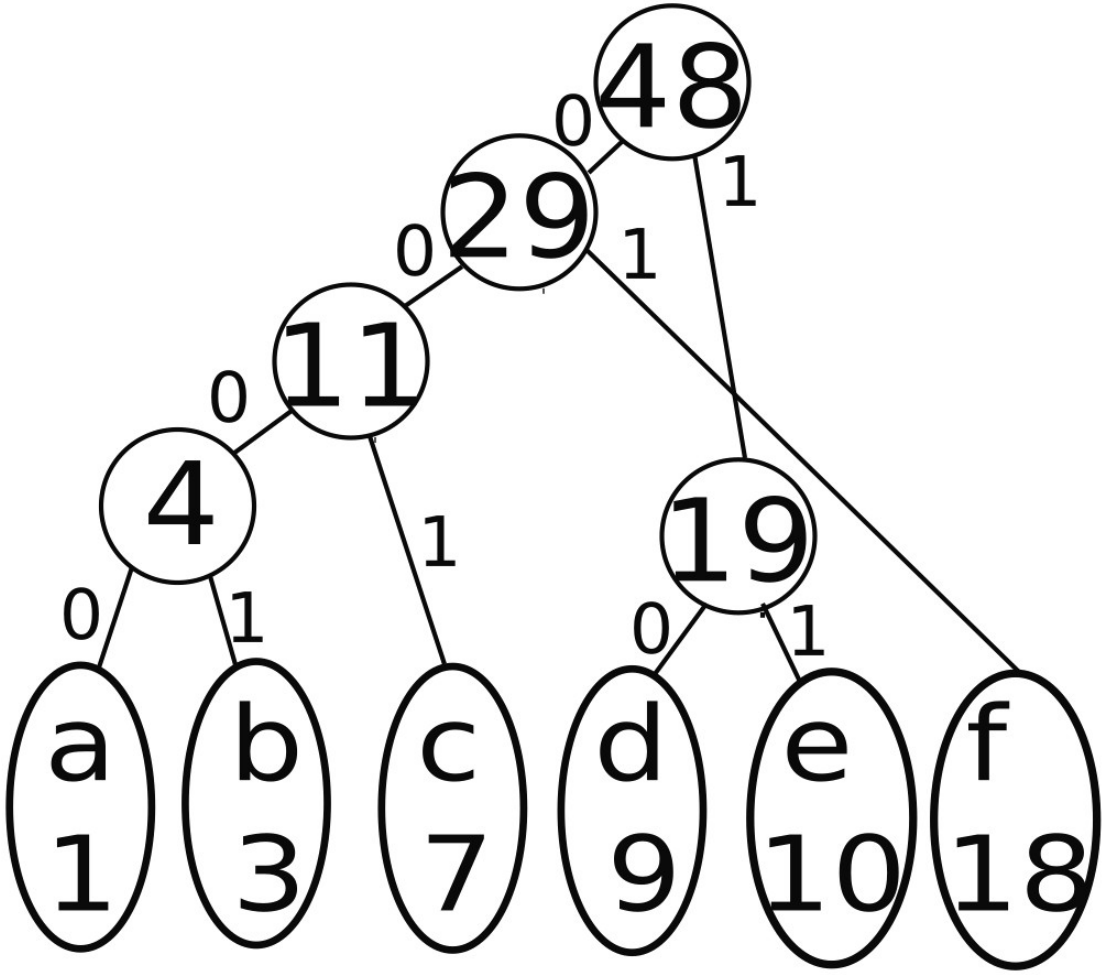


Assignment

You stop if there is only one node left.
For every node which is not a Leaf:

- Right edge: 1
- Left edge: 0

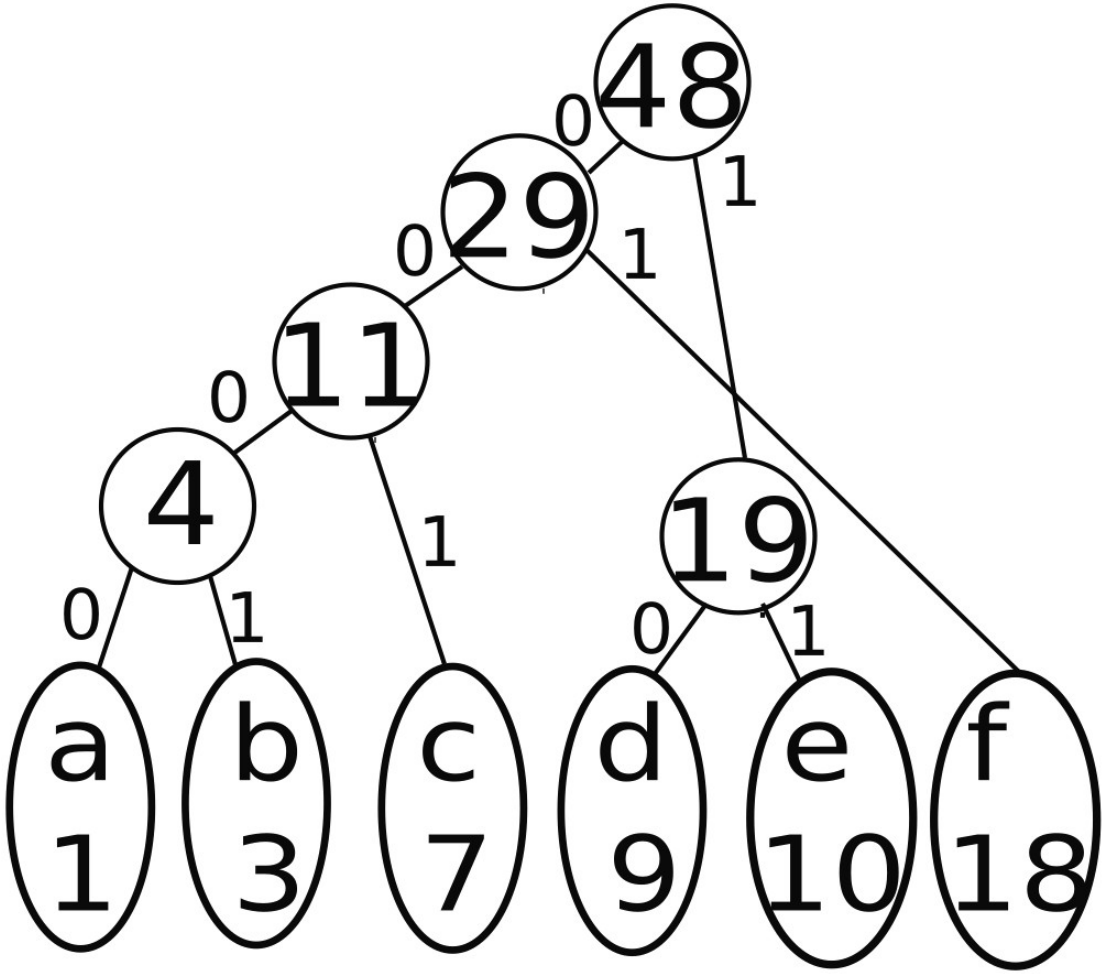
Huffman - tree



Assignment

Following the Edges from Root to Leaf you can just read the coding scheme:

Letter	Code
a	0000
b	0001
c	001
d	10
e	11
f	01



Assignment

Your job:
Write a program which takes the following frequencies as an input and gives out a coding scheme according to Huffman!
(Either take this one or the one shown in the Exercise. They are equally difficult)

Letter	a	b	c	d	e	f	g	h
Frequency	14	20	7	10	23	5	16	3

Assignment

It will be allright to hard – code the Input data.
Most important that the algorithm works like the shown way.
Output can look like this (for Example):

```
a - 11
b - 101
c - 100
d - 01
e - 001
f - 0001
g - 00001
h - 00000
```

Don‘t need to represent the tree graphically. It‘s handled internally

Assignment Requirements

- Use Java, C or C++
- Compilable source code (gcc or javac)
- No IDEs
- Readable source code
- Self-contained submission
- Sufficient comments (rule of thumb: another Developers can get program logic alone by reading the comments)

Assignment Requirements

- 5 Assignments
- In each!
  Data: Name, matriculation number, SS2018
- max 6 point for each assigment
- 30% exercise and 70% final exam

Assignment Requirements

- I suggest to write a „node“ - Class to build the tree.
- Again: Do not copy code. Not from the Internet. Not from your patner. We will find out.
- Deadline: 15. May 2018 23:59
- Mailto: michael.schwarzkopf@uni-weimar.de

Good luck!