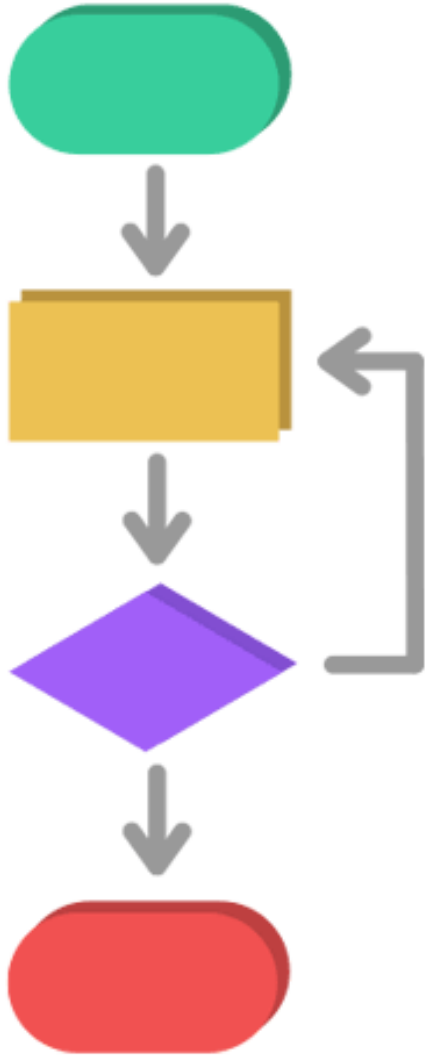
Algorithmen Tutorium

Beginn: 16:15



03.11.2021

Organisation

- Nur noch kurze Besprechung des UB
  - Was ist wichtig & meine Bewertung
  - Vorrechnen am Montag 16-18 Uhr
- Abgabe: Zip-Archiv mit allen Dateien (oder nur ein PDF)
  - Quellcode der Programme abgeben (.c/.cpp und .h Dateien)
  - Programm muss nicht gebuildet werden (keine .exe)
  - Kompilierte/Gebuildete Programme werden NICHT bewertet!!
    - Nicht nur eine .exe oder die kompilierte Datei abgeben!

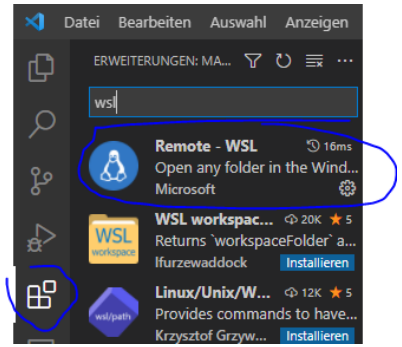
Programmieren C I (Windows + Linux)

- Virtualisierung im BIOS aktivieren (Google nach Mainboard/PC-Hersteller)
- Terminal: `wsl --install`
- In der WSL-Shell: <Nutzername & Passwort festlegen>
- WSL-Shell: `sudo apt update`
- WSL-Shell: `sudo apt install build-essentials`

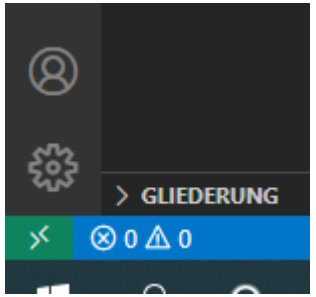
Programmieren C II (Windows + Linux)

- Visual Studio Code installieren
- Installiere Extension: „Remote – WSL“
- Öffne Ordner in WSL (unten links)
- Installiere (in WSL) Extension: „C/C++“ UND „Code Runner“
- <Programmieren> (und alle Files speichern)
- „Main.c“ auswählen kompilieren

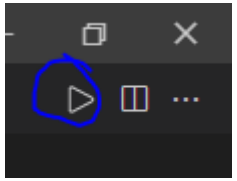
1.



2.



3.



Programmieren C (Mac)

- Angaben ohne Gewähr
- Terminal: `xcode-select --install`
- Terminal: `cd <my-dir>`
- Terminal: `cc <my-file.c>` bzw. `gcc <my-file.c>`
- Nach xCode-Installation evtl. analog zu VSCode

Logarithmus

- Problem: Welche Basis *log(n)* ? *10*, *2* oder *e*?
- Aufteilung meistens nach „Fach“:
  - Informatik: 2
  - Mathematik: e (teilweise auch 10)
  - Alltag: 10
- In dieser Vorlesung: 2

Arrays und Listen

ARRAYS

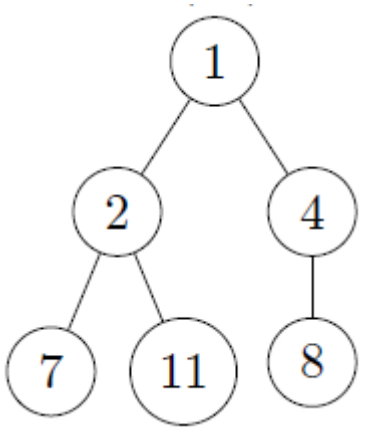
- Feste Größe
- Zugriff auf jedes Element
  - Direkter Zugriff => O(1)

LISTE

- Variable Größe/Länge
- Zugriff nur auf das „nächste“ Element
  - Liste komplett durchlaufen => O(n)
- 1-fach verkettet:
  - Zugriff in eine Richtung (vor)
- 2-fach verkettet:
  - Zugriff in beide Richtungen (vor & zurück)

Heaps

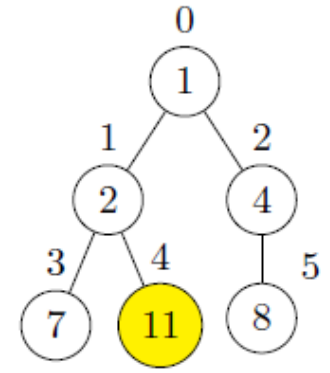
- Vollständiger Binärer Baum
- Heapeigenschaft:
  - Kindsknoten sind größer/gleich als Elternknoten
  - Kleinstes Element ist die Wurzel
  - (Analog mit kleiner/gleich und dem größten Element)
- Operation: *heapify* = Herstellen der Heapeigenschaft



Heaps

- Random Fact:

Heaps sind vollständig, somit können sie ebenweise, von links nach rechts durchlaufen werden, ohne ein Loch zu finden.
Außderdem hält jede volle Ebene auf mit Tiefe h 2^h Elemente (Wurzel hat Tiefe 0). Das x-te (von 0) Element auf Ebene h hat bekommt somit den Index $2^h + x$.
Der Heap kann so durch ein Array realisiert werden, wobei jedes Element wie eben beschrieben addressiert wird.



1	2	4	7	11	8

Index $= 2^2 + 0 = 4$✓

Heaps

- Beispiel mit dem Array:

7	1	8	2	11	4