Datenstrukturen

Überblick

- Was sind Datenstrukturen?
- Grundlegende Datenstrukturen:
  - Arrays
  - Queue (Warteschlange)
  - Stack (Stapel)
  - Heap
  - Baum
  - Graph
- Vergleich und Anwendungen

Was sind Datenstrukturen?

- **Definition:** Spezielle Formate zur Organisation, Verwaltung und Speicherung von Daten
- **Zweck:** Effiziente Zugriffs- und Änderungsoperationen
- **Bedeutung:** Grundlage für effiziente Algorithmen und Softwareentwicklung
- **Auswahlkriterien:** Zeit- und Speicherkomplexität, Anwendungsfall

Big O-Notation

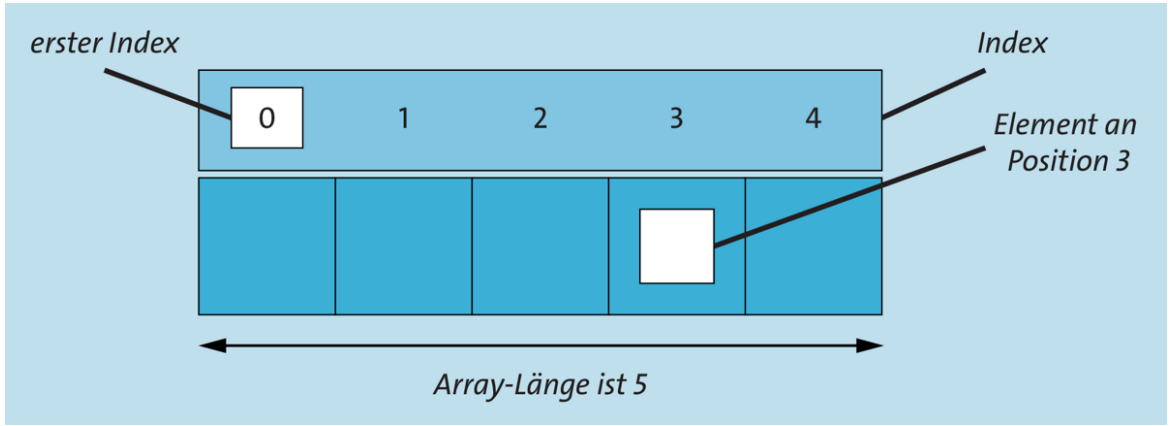
- Definition: Diese Notationen beschreiben die Zeitkomplexität von Algorithmen oder Operationen auf Datenstrukturen
- Bedeutung: Effizienzmaß für Algorithmen und Datenstrukturen
- Häufige Komplexitätsklassen:
  - O(1) - Konstante Zeit: Unabhängig von der Datenmenge
  - O(log n) - Logarithmische Zeit: Sehr effizient für große Datenmengen
  - O(n) - Lineare Zeit: Proportional zur Datenmenge
  - O(n log n) - Linearithmische Zeit: Effiziente Sortierverfahren
  - $O(n^2)$ - Quadratische Zeit: Ineffizient für große Datenmengen
  - $O(2^n)$ - Exponentielle Zeit: Praktisch nur für kleine Eingaben nutzbar
  - O(V+E) - Graphen-Komplexität: Abhängig von Anzahl der Knoten (V) und Kanten (E)

Array(Statisch)

Beschreibung: Sammlung von Elementen gleichen Typs, die sequentiell im Speicher angeordnet sind

- Eigenschaften:
  - Feste Größe (in vielen Sprachen)
  - Indexbasierter Zugriff (O(1))
  - Homogene Daten
- Operationen:
  - Lesen: O(1)
  - Suchen: O(n)
  - Einfügen/Löschen: O(n)
- Anwendungen: Tabellen, Matrizen, Listen mit konstanter Größe

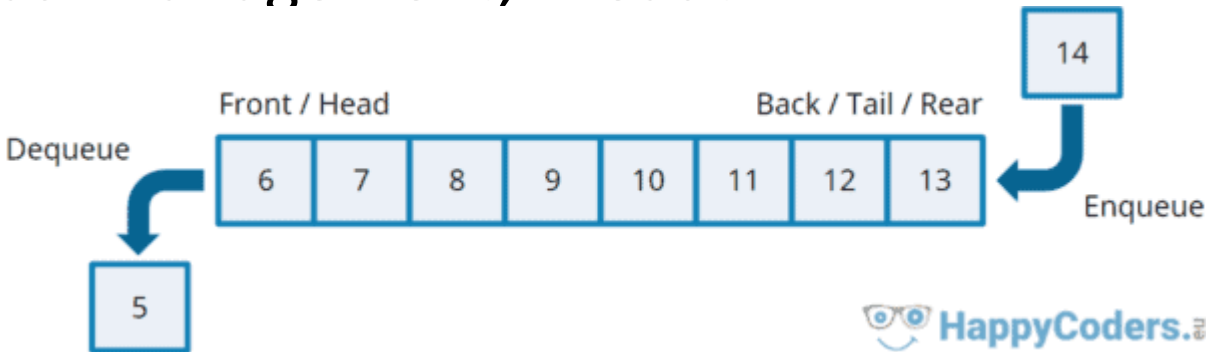
https://www.youtube.com/watch?v=SRJZ1XmqHfA



Queue (Warteschlange)

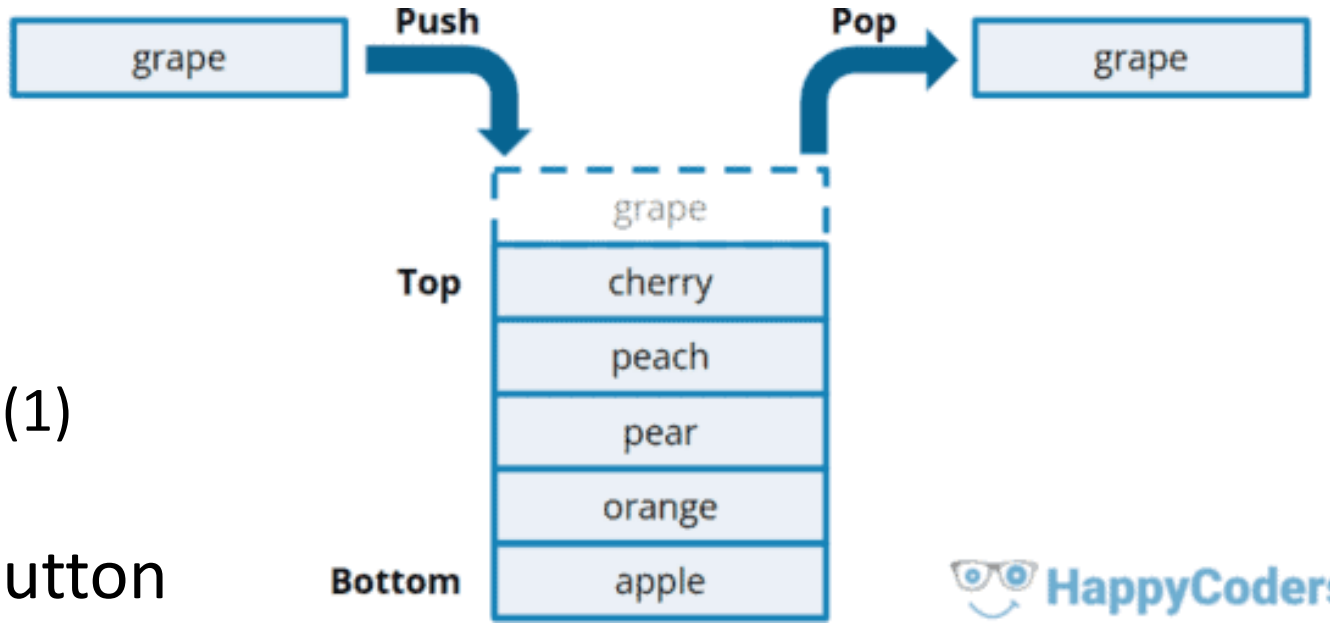
- Beschreibung: FIFO-Prinzip (First In, First Out)
- Eigenschaften:
  - Elemente werden am Ende hinzugefügt
  - Elemente werden vom Anfang entfernt
- Operationen:
  - Enqueue (Hinzufügen): O(1)
  - Dequeue (Entfernen): O(1)
  - Peek (Erstes Element anschauen): O(1)
- Anwendungen: Druckerwarteschlangen, Taskmanagement, Breadth-First-Search

https://www.youtube.com/watch?v=iMs2Fq8O9T4



Stack (Stapel)

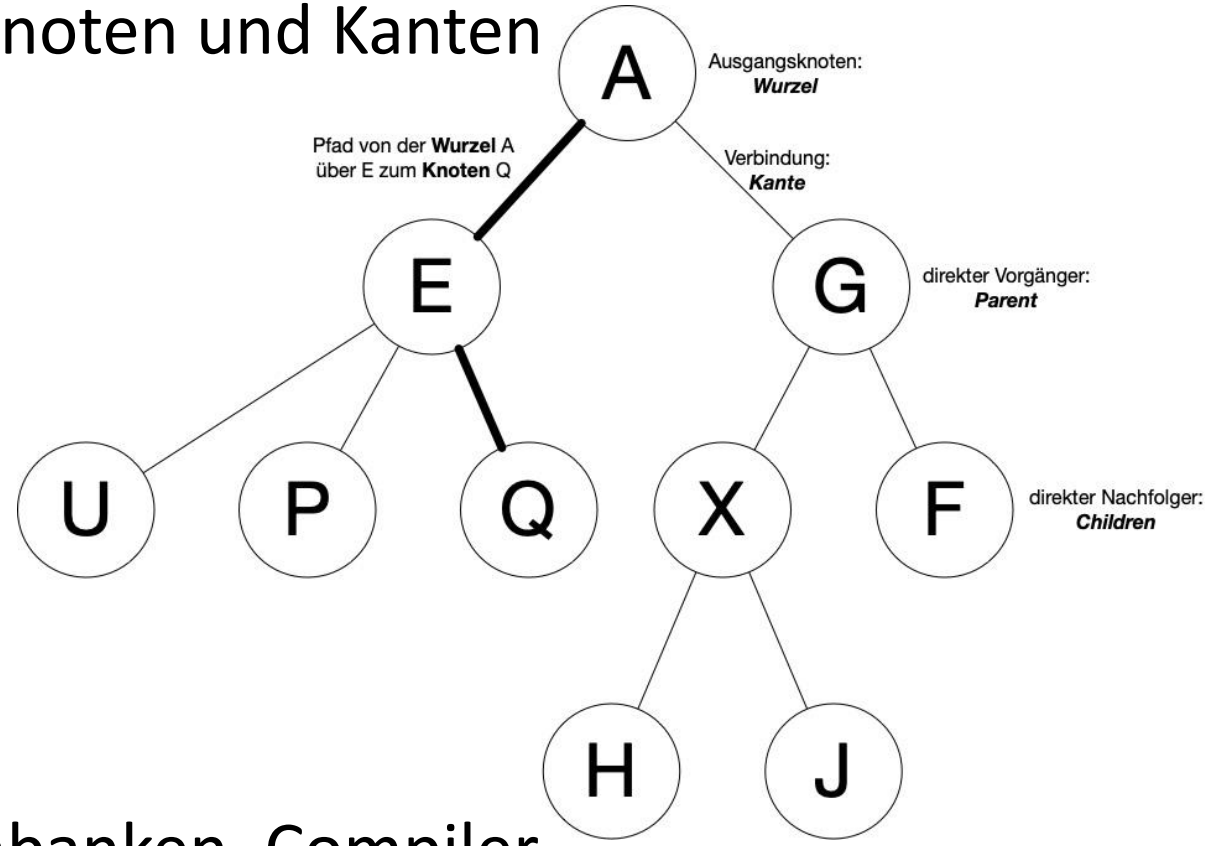
- Beschreibung: LIFO-Prinzip (Last In, First Out)
- Eigenschaften:
  - Elemente werden oben hinzugefügt
  - Elemente werden von oben entfernt
- Operationen:
  - Push (Hinzufügen): O(1)
  - Pop (Entfernen): O(1)
  - Peek (Oberstes Element anschauen): O(1)
- Anwendungen: Funktionsaufrufe, Undo-Funktionen, Browser Zurück-Button



https://www.youtube.com/watch?v=K2Lm8UFS9_8

Baum

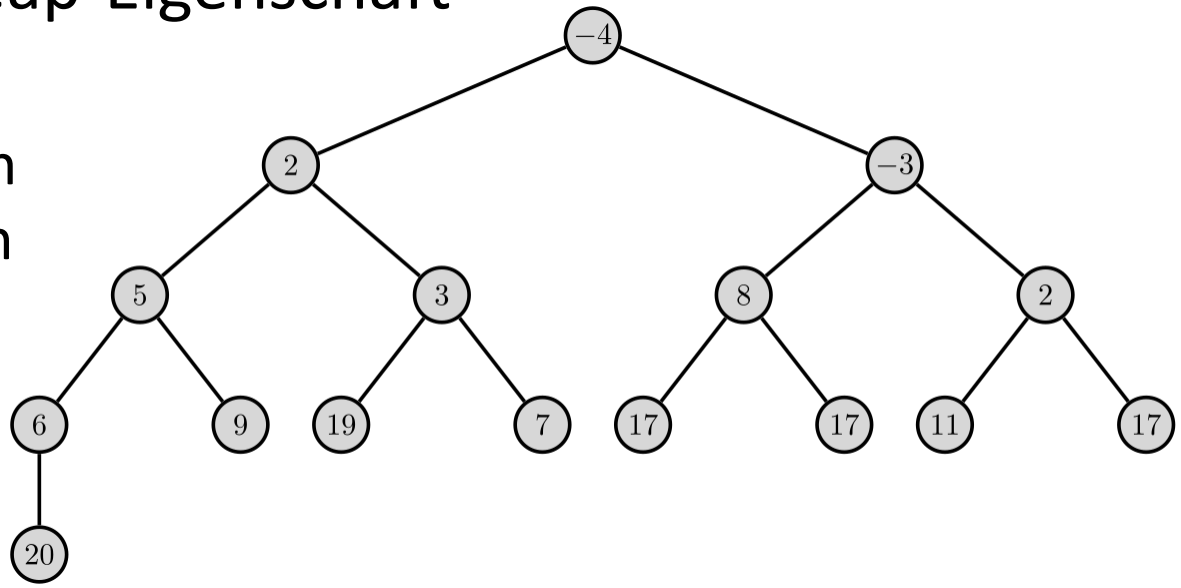
- Beschreibung: Hierarchische Struktur mit Knoten und Kanten
- Varianten:
  - Binärbaum
  - Suchbaum (BST)
  - AVL-Baum
  - B-Baum
- Operationen:
  - Suchen: O(log n) im ausgeglichenen Fall
  - Einfügen: O(log n) im ausgeglichenen Fall
  - Löschen: O(log n) im ausgeglichenen Fall
- Anwendungen: Hierarchische Daten, Datenbanken, Compiler



https://www.youtube.com/watch?v=dqKoYntQHzc

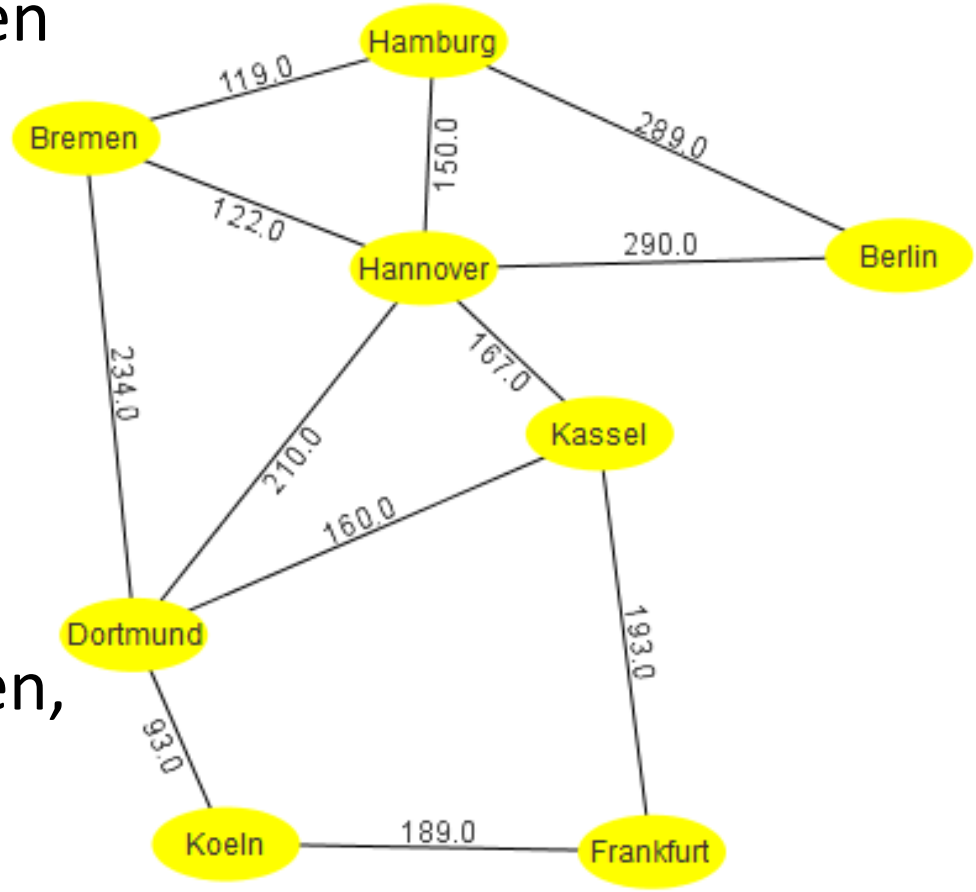
Heap

- Beschreibung: Spezieller Binärbaum mit Heap-Eigenschaft
- Varianten:
  - Max-Heap: Elternknoten größer als Kindknoten
  - Min-Heap: Elternknoten kleiner als Kindknoten
- Operationen:
  - Einfügen: O(log n)
  - ExtractMax/Min: O(log n)
  - Finden von Max/Min: O(1)
- Anwendungen: Prioritätswarteschlangen, Heap-Sort, Dijkstra's Algorithmus



Graph

- Beschreibung: Sammlung von Knoten und Kanten
- Varianten:
  - Gerichtet/Ungerichtet
  - Gewichtet/Ungewichtet
  - Zyklisch/Azyklisch
- Darstellungen:
  - Adjazenzmatrix
  - Adjazenzliste
- Anwendungen: Soziale Netzwerke, Straßenkarten, Webseiten-Verlinkungen



https://www.youtube.com/watch?v=frG2kBMZQDc

Vergleich der Datenstrukturen

Datenstruktur	Zugriff	Suche	Einfügen	Löschen	Speicherbedarf
Array	O(1)	O(n)	O(n)	O(n)	Niedrig
Queue	O(n)	O(n)	O(1)	O(1)	Mittel
Stack	O(n)	O(n)	O(1)	O(1)	Niedrig
Heap	O(n)	O(n)	O(log n)	O(log n)	Mittel
Baum (BST)	O(log n)	O(log n)	O(log n)	O(log n)	Hoch
Graph	O(1)	O(V+E)	O(1)	O(V+E)	Hoch

Wann welche Datenstruktur?

- Array: Wenn schneller Zugriff auf Elemente an bekannten Positionen benötigt wird
- Queue: Für FIFO-Verarbeitung von Daten
- Stack: Für LIFO-Verarbeitung und rekursive Algorithmen
- Heap: Wenn schneller Zugriff auf das Maximum/Minimum benötigt wird
- Baum: Für hierarchische Daten und effiziente Suche
- Graph: Für Beziehungsdaten und Netzwerke

Zusammenfassung

- Datenstrukturen sind essenziell für effiziente Algorithmen
- Die Wahl der richtigen Datenstruktur hängt vom Anwendungsfall ab
- Verschiedene Datenstrukturen bieten unterschiedliche Vor- und Nachteile
- Optimale Nutzung erfordert Kenntnis der Komplexität und Eigenschaften