

# Kapitel ADS:IV

## IV. Datenstrukturen

- ☐ Record
- ☐ Container
- ☐ List
- ☐ Linked List
- ☐ Stack
- ☐ Queue
- ☐ Priority Queue
- ☐ Dictionary
- ☐ Direct-address Table
- ☐ Hash Table
- ☐ Hash Function

# Record

## Definition

Ein Record (*Verbund*) fasst einen oder mehrere primitive Datentypen und/oder Records zusammen. Die Bestandteile eines Records können durch eindeutige Schlüsselwörter benannt werden.

# Record

## Definition

Ein Record (*Verbund*) fasst einen oder mehrere primitive Datentypen und/oder Records zusammen. Die Bestandteile eines Records können durch eindeutige Schlüsselwörter benannt werden.

## Manipulation

- ❑ **Komponente lesen**  
Den Wert einer Komponente des Records lesen und anderswo verarbeiten.
- ❑ **Komponente modifizieren**  
Den Wert einer Komponente des Records neu zuweisen.
- ❑ **Records vergleichen**  
Auf Gleichheit prüfen: Zwei Records sind gleich, wenn sie komponentenweise gleich sind.

# Record

## Definition

Ein Record (*Verbund*) fasst einen oder mehrere primitive Datentypen und/oder Records zusammen. Die Bestandteile eines Records können durch eindeutige Schlüsselwörter benannt werden.

## Manipulation

- ❑ **Komponente lesen**  
Den Wert einer Komponente des Records lesen und anderswo verarbeiten.
- ❑ **Komponente modifizieren**  
Den Wert einer Komponente des Records neu zuweisen.
- ❑ **Records vergleichen**  
Auf Gleichheit prüfen: Zwei Records sind gleich, wenn sie komponentenweise gleich sind.

## Implementierung

- ❑ Repräsentation mehrerer Records in parallelen Arrays.
- ❑ Ablegen der Komponenten in eindeutiger Reihenfolge im Speicher;  
Verweis auf Speicherposition mit Pointer.

# Record

Beispiel: Records sortieren

## Records

	$R_1$	$R_2$	$R_3$
$A$	89	37	41
$B$	R	S	O
$\vdots$	$\vdots$	$\vdots$	$\vdots$

# Record

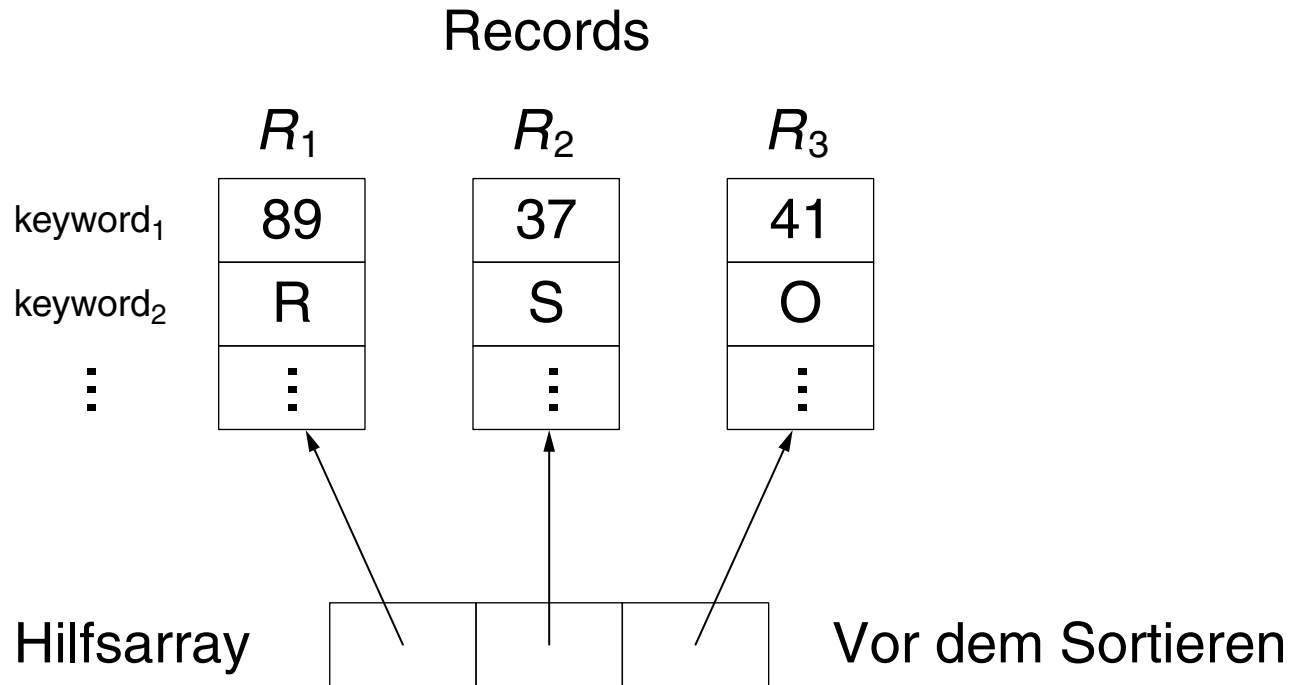
Beispiel: Records sortieren

## Records

	$R_1$	$R_2$	$R_3$
keyword <sub>1</sub>	89	37	41
keyword <sub>2</sub>	R	S	O
⋮	⋮	⋮	⋮

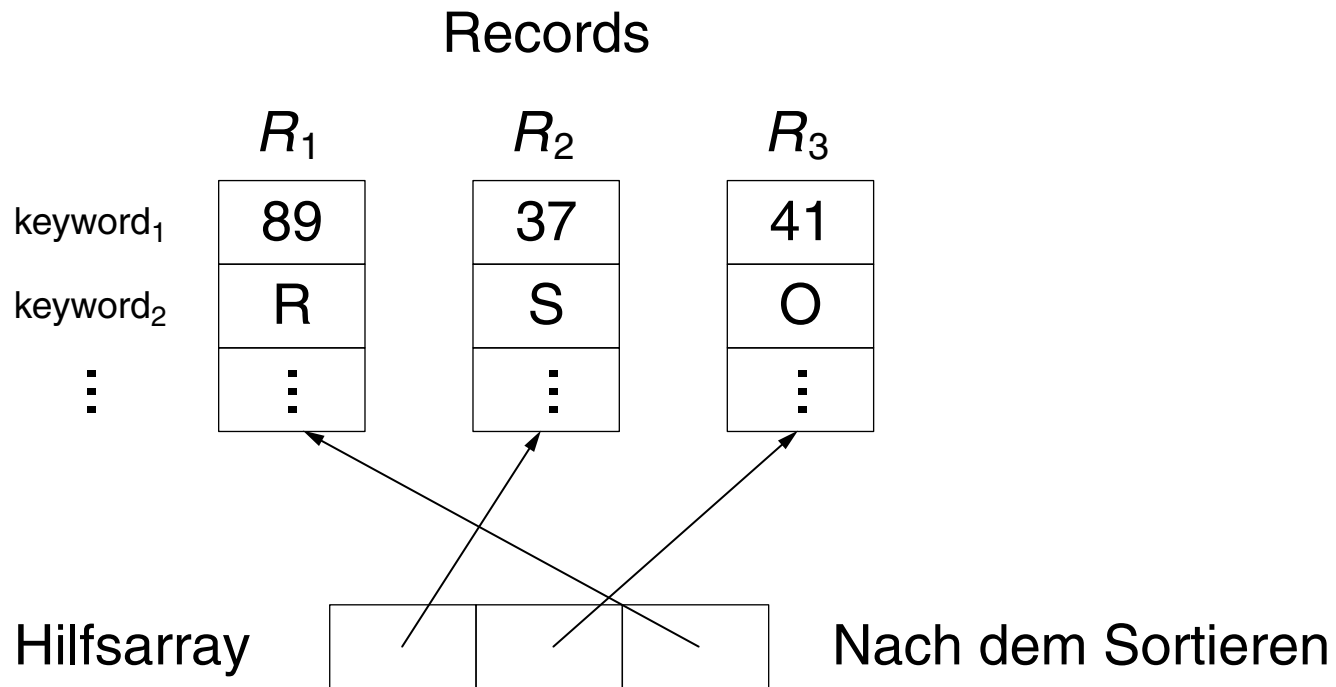
# Record

Beispiel: Records sortieren



# Record

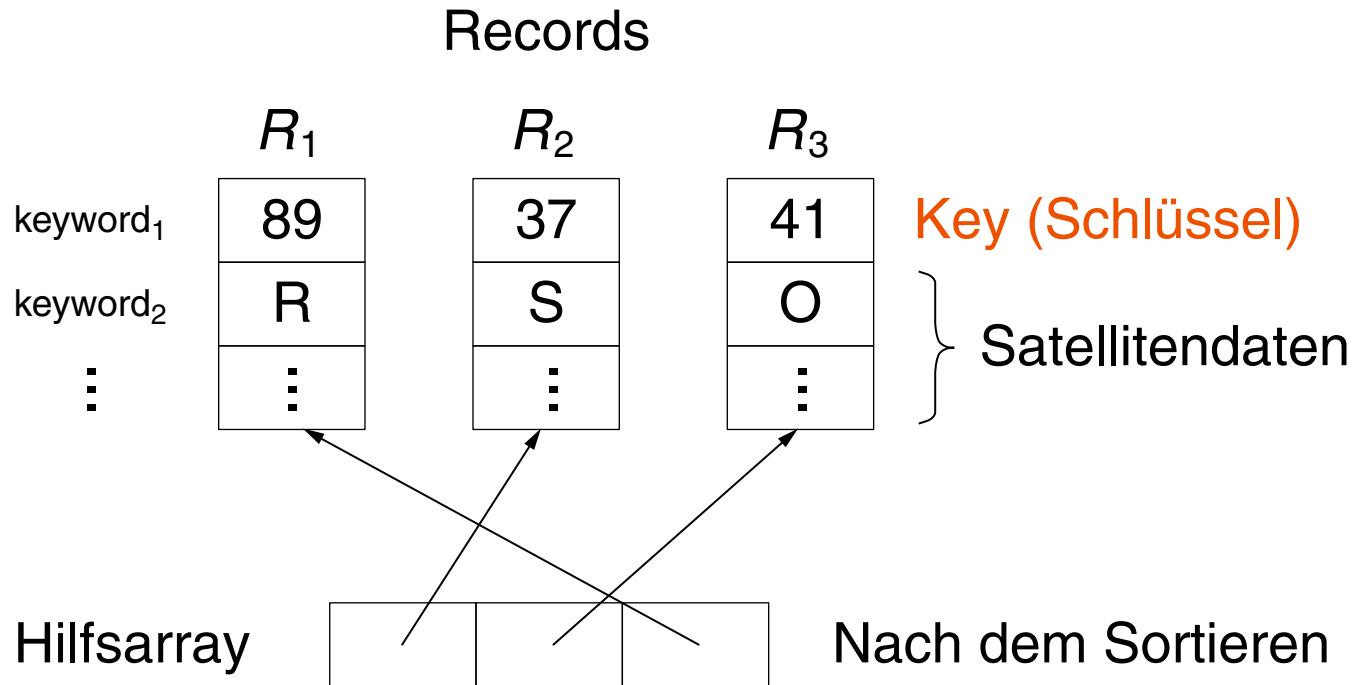
## Beispiel: Records sortieren





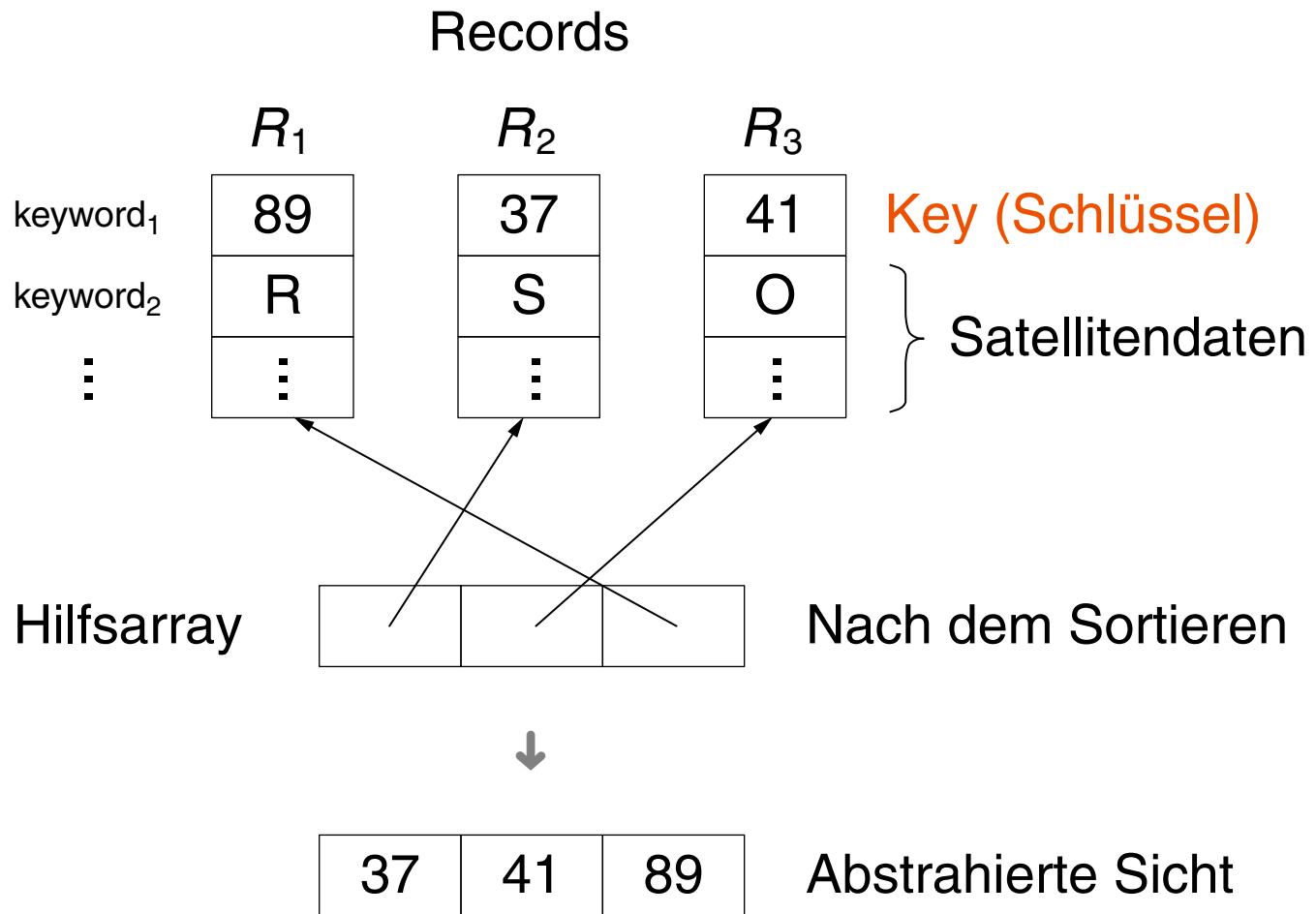
# Record

Beispiel: Records sortieren



# Record

Beispiel: Records sortieren



# Container

## Definition

Ein Container ist eine abstrakte Datenstruktur für eine endliche Anzahl an Elementen, zwischen denen eine **binäre Relation** aufreht erhalten wird.

# Container

## Definition

Ein Container ist eine abstrakte Datenstruktur für eine endliche Anzahl an Elementen, zwischen denen eine **binäre Relation** aufrecht erhalten wird.

Mögliche Relationen:

- ❑ **Keine Relation**

Jedes Element ist unabhängig von allen anderen.

- ❑ **Adjazenzrelation**

Jedes Element hat beliebig viele Nachbarelemente.

- ❑ **Äquivalenzrelation**

Jedes Element ist selbstäquivalent; paarweise Äquivalenzen sind symmetrisch und transitiv.

- ❑ **Partielle Ordnung**

Jedes Element hat beliebig viele Nachfolger, ohne dass dabei Zyklen gebildet werden.

- ❑ **Hierarchische Ordnung**

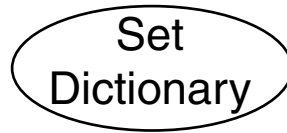
Jedes Element hat ein Elternelement, mit Ausnahme eines sogenannten Wurzelements.

- ❑ **Totale Ordnung**

Jedes Element hat ein Vorgänger- und ein Nachfolgerelement, mit Ausnahme eines sogenannten Kopfelements ohne Vorgänger und eines Endelements ohne Nachfolger.

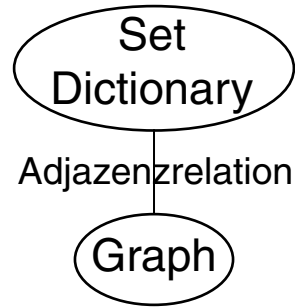
# Container

## Typen



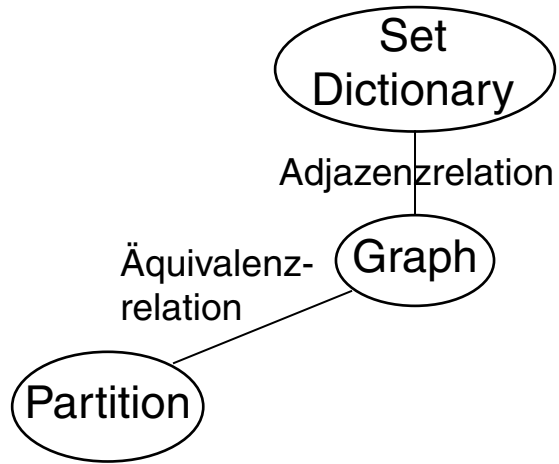
# Container

## Typen



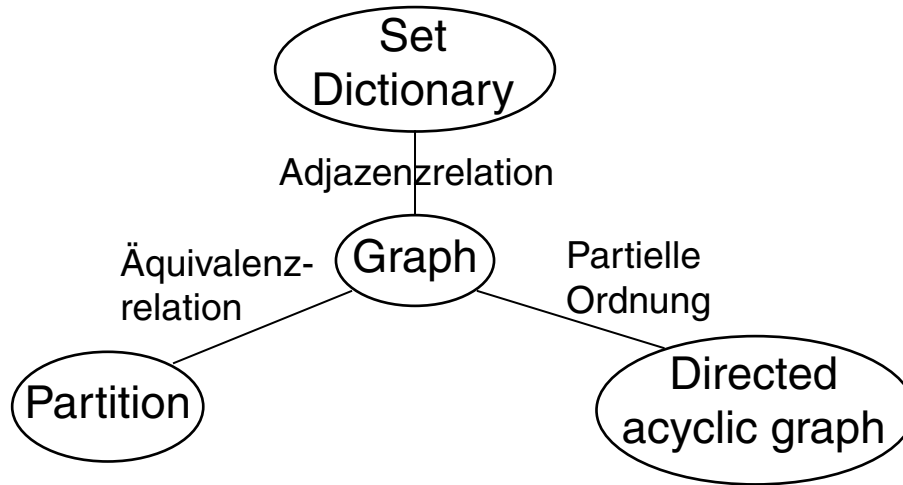
# Container

## Typen



# Container

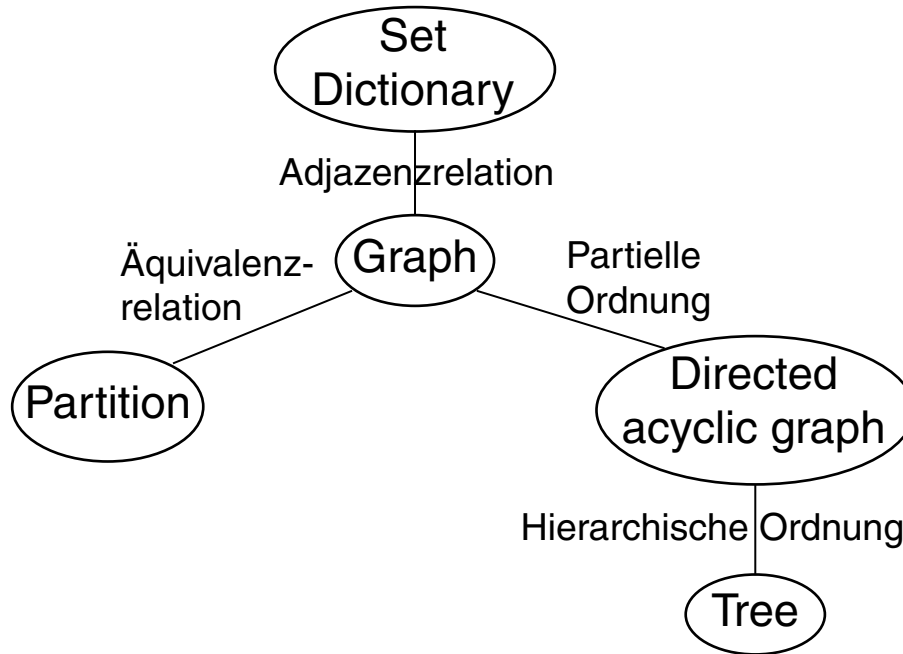
## Typen





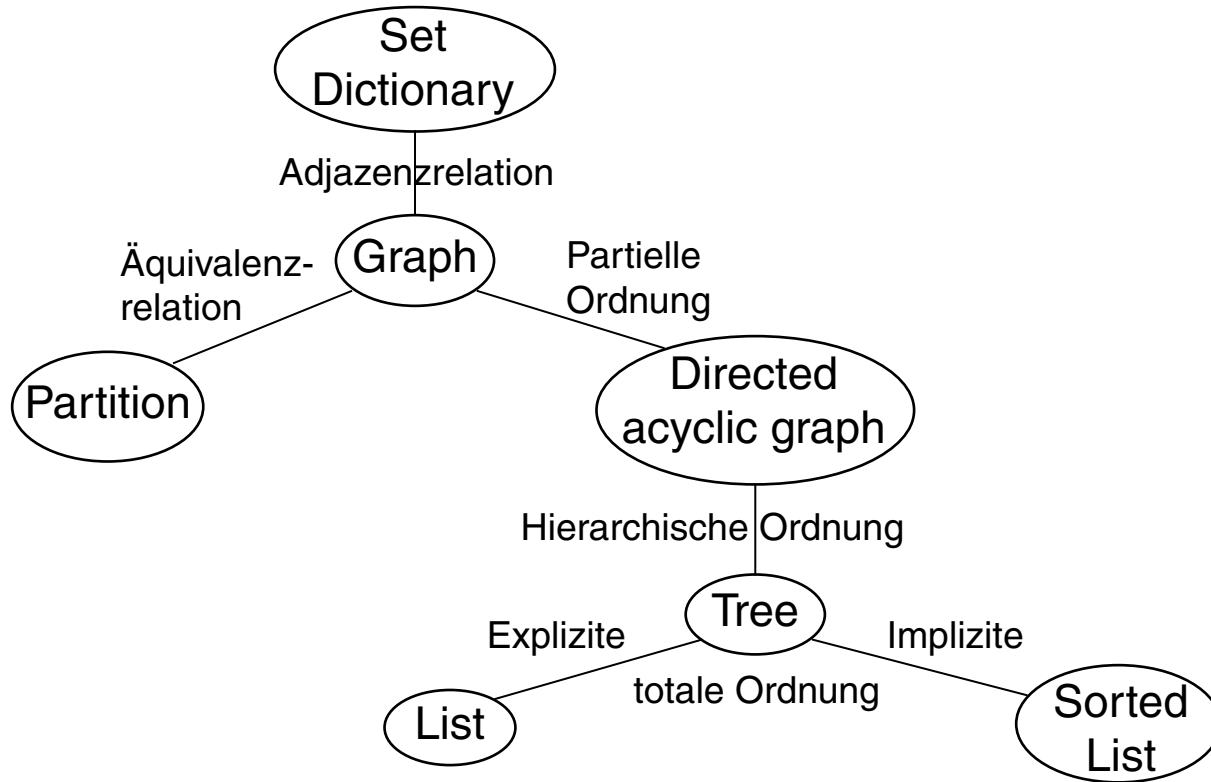
# Container

## Typen



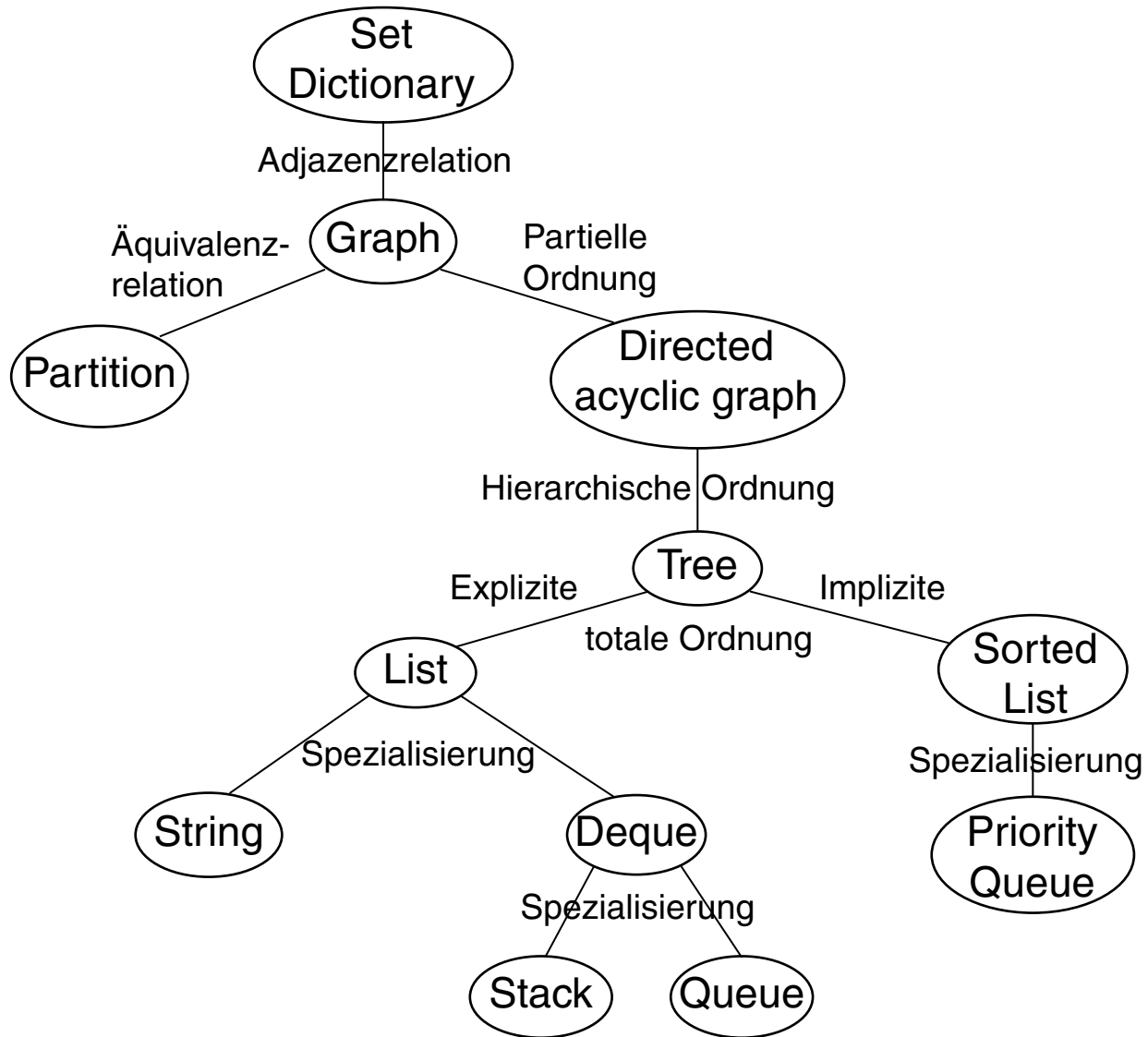
# Container

## Typen



# Container

## Typen



## Bemerkungen:

- ❑ Ein Containertyp zusammen mit für diesen Typ definierten Manipulationsoperationen wird *abstrakter Datentyp* genannt. Die meisten Programmiersprachen implementieren Varianten der List-Typen sowie das Set/Dictionary. Implementierungen von Graphen und Bäumen sind nicht notwendigerweise vorhanden.
- ❑ Jeder der Containertypen unterstützt eine Reihe von Manipulationsoperationen. Deren Effizienz hängt von den durch die jeweilige Relation geforderten Einschränkungen sowie von ihrer Implementierung ab.
- ❑ Die durch Spezialisierung abgeleiteten List-Typen haben sich in vielen Anwendungen bewährt.
- ❑ Eine total Ordnung kann implizit durch Spezifikation einer  $\leq$ -Relation auf den zu speichernden Records definiert werden, die dann durch die Datenstruktur hergestellt wird.
- ❑ Explizite totale Ordnungen sind unabhängig vom Typ der Elemente und werden von außen durch den Nutzer oder durch einen Prozess, der Elemente erzeugt, bestimmt. Beispielsweise werden Buchstaben von Menschen in eine Reihenfolge gebracht, die interpretiert werden kann oder Emails erreichen ihre Adressaten in einer bestimmten Reihenfolge.
- ❑ Mathematische Objekte wie Zahlen, Vektoren und Matrizen werden hier nicht berücksichtigt, da sie gegebenenfalls als Elemente eines Container fungieren. Datenstrukturen, die Matrizen implementieren, werden auch eingesetzt, um Graphen zu repräsentieren.