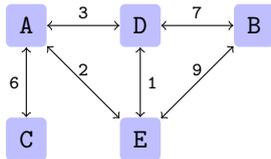


1 Graphen

Def.: Ein Graph besteht aus einer Menge von Knoten und Kanten, wovon jede Kante genau zwei Knoten verbindet. Diese können allgemein Netzwerke darstellen und beschreiben (Straßennetze, Computernetze, Soziale Netze, ...)



1.1 Einteilungen

Ein Graph ist **gerichtet**, wenn zwischen Start- und Zielknoten unterschieden wird, **gewichtet**, wenn jede Kante eine Bewertung hat (bswp. bei Straßennetz, Abstand).

Der obige Graph ist daher ungerichtet, aber gewichtet.

1.2 Implementierung

Mit einer **Adjazenzmatrix** kann dieser Graph tabellarisch dargestellt werden:

	A	B	C	D	E
A	0	-1	6	3	2
B	-1	0	-1	7	9
C	6	-1	0	-1	-1
D	3	7	-1	0	1
E	2	9	-1	1	0

In jeder Zelle wird die Bewertung der Kante hineingeschrieben, welche die Zwei Knoten verbindet, 0 wenn der Knoten „zu sich selbst“ zeigt, -1 wenn unverbunden.

Dieses kann daher mit einem Zweidimensionalen Feld (`int [][]`) dargestellt werden.

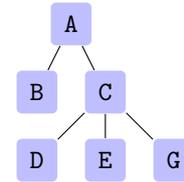
Wenn die Matrix vertikal symmetrisch ist, deutet es darauf, dass der Graph ungerichtet ist.

2 Bäume

Def.: Ein Baum ist ein besonderer Graph, mit folgenden Eigenschaften:

1. Es enthält ein „Wurzel“ Knoten
2. Jeder Knoten ist verbunden, mit 0 oder mehr „Kindern“, wovon jeder der Wurzel eines weiteren Baumes ist.
3. Jeder Knoten hat genau einen Vorgänger

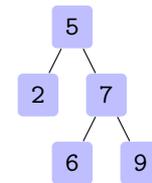
Ein Baum kann also so aussehen:



Bäume können genutzt werden um hierarchische Verhältnisse darzustellen, wie in der Gesellschaft oder einer Ordner-Struktur am Computer.

2.1 (Sortierte) Binärbäume

Def.: Ein Binärbaum, ist ein solcher Baum, wo jeder Knoten höchstens zwei Nachfolger hat. Bei einem **sortierten** Binärbaum, wird immer (bswp.) der linke Knoten einen kleineren Wert, und der rechte einen größeren habe als der jetzige Knoten, wie hier gesehen:



2.1.1 Einfügen

Um in einen sortierten BB. einzufügen, muss der Wert (bzw. der Schlüssel) erst mit dem

Wurzelknoten verglichen werden, und dann je nach Wert nach rechts oder links gegeben werden, bis eine freie Stelle gefunden wurde.

Aus diesem Grund kann ein BB. kein Schlüssel zweimal enthalten.

2.1.2 Suche (oder „binary search“)

Das Suchverhalten bei einem BB kann mit $\log_2 n$ beschrieben werden, wobei n die Anzahl an Knoten ist.

Hier wird also der gesuchte Schlüssel s erst mit dem des Wurzelknoten k verglichen, und dann wenn

$s = k$ Der Wert des Knotens wird zurückgegeben; die Suche ist vorbei

$s < k$ Der linke Teilbaum wird mit s abgesucht

$k > s$ Der rechte Teilbaum wird mit s abgesucht

Wenn k keine Kinder hat, wird die Suche abgebrochen.

2.1.3 Löschen

Um einen Knoten zu löschen, muss dieser erst im Baum gefunden werden. Beim löschen eines Knoten, werden verschiedene Fälle berücksichtigt: Der Knoten hat ...

1. **keine Kinder:** Der Knoten kann einfach entfernt werden
2. **nur ein Kind:** Der Knoten wird mit dem Kind ersetzt
3. **beide Kinder:** Es wird zunächst der kleinste Knoten m im rechten Teilbaum R gesucht, und mit dem jetzigen Knoten ersetzt. Nun versucht man m in R mit den vorherigen schritten zu löschen.

2.1.4 Traversierung

Für die Herausgabe aller Knoten in einem Baum, gibt es folgende Methoden/Reihenfolgen:

Inorder

linker Baum, Wurzel, rechter Baum

Im vorherigen Baum: 2, 5, 6, 7, 9
→ die Ausgabe ist sortiert

Postorder

linker Baum, rechter Baum, Wurzel

Im vorherigen Baum: 2, 5, 6, 7, 9
→ kann bswp. für Rechenbäume genutzt werden.

Preorder

Wurzel, linker Baum, rechter Baum

Im vorherigen Baum: 2, 5, 6, 7, 9
→ die Ausgabe kann genutzt werden um den Baum identisch wieder aufzubauen.

2.1.5 Zählen

Die Anzahl der Knoten eines Baumes, wird so bestimmt:

1. Für eine Knoten k , nimmt Man die Zahl $z := 1$
2. Wenn k einen Linken Teilbaum L hat, so wiederholt man dieses ganze für L , und addiert dessen z_L zu z
3. Wenn k einen Linken Teilbaum L hat, so wiederholt man dieses ganze für L , und addiert dessen z_R zu z
4. Für einen leeren Ast, wird zu z kein Wert addiert.
→ z ist damit die Anzahl der Knoten für diesen Baum.

Der erste Knoten der benutzt wird, ist der Wurzelknoten des Baumes.