3.2 Graph mit Adjazenzmatrix

Nachdem wir uns mit der Grundlegenden Struktur des Graphen beschäftigt haben, stellt sich die Frage, wie wir dies mit dem Computer umsetzen können.

Die zentrale Frage stellt sich darin, wie die „Nachbarn“ eines KNOTEN im GRAPH gespeichert werden können. Dabei gibt es zwei zentrale Ansätze.

Der erste (vielleicht naheliegende) Ansatz ist, dass man in jedem KNOTEN eine Liste der Nachbarn abspeichert mit dementsprechenden Gewichtungen der KANTEN. Dafür braucht man eine Klasse KANTE, die dem Nachbarn eine Gewichtung zuordnen kann.

Der zweite Ansatz ist, eine zentrale „Tabelle“ zu erstellen, in der die Verbindungen von einem KNOTEN zu einem anderen vermerkt werden.

Beide Ansätze haben ihre eigenen Anwendungsbereiche. Vergleichen wir die beiden Ansätze an einem Beispiel:

1.Liste:

Mü: -(163)→ Nü -(117)→ Re

Nü: -(80)→ Re -(163)→ Mü

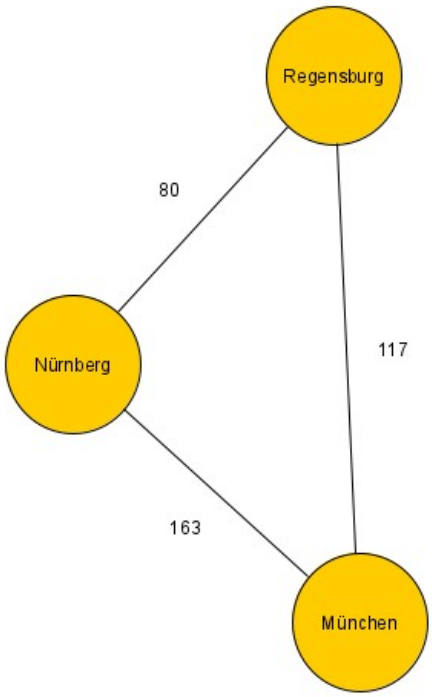
Re: -(80)→ Nü -(117)→ Mü

(dynamisch)

2. Tabelle:

	München	Nürnberg	Regensburg
München	0	163	117
Nürnberg	163	0	80
Regensburg	117	80	0

(statisch)

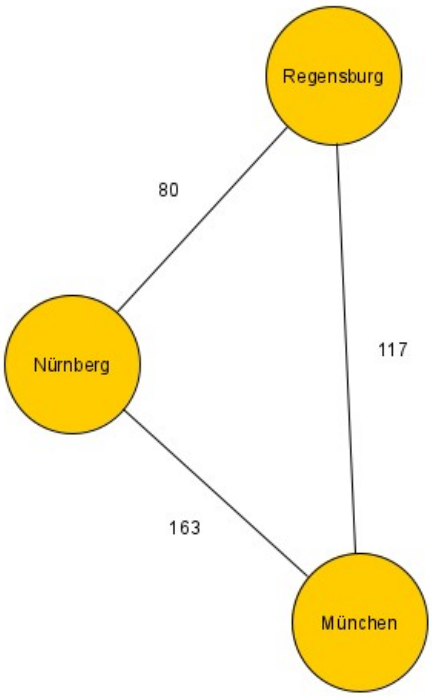


Ist die Anzahl der KNOTEN in einem Graph bekannt (z.B. Straßennetz), so kann man den GRAPH mit einer Tabelle beschreiben. Die Tabelle lässt sich am besten mit Feldern umsetzen. Jede Zeile der Tabelle entspricht einem Feld des Datentyps Integer. Die Felder können dann in einem Feld zusammengefasst werden und man erhält ein zweidimensionales Feld (`int[][] matrix`) für die Kanten des Graphen. Die KNOTEN werden in einem Feld der Klasse KNOTEN gespeichert.

Das zweidimensionale Array wird ohne die Bezeichner der KNOTEN (hier: Städtenamen) aufgeschrieben. Man nennt dies die Adjazenzmatrix des Graphen.

In unserem Beispiel würde die Matrix wie folgt aussehen:
(Indexe der KNOTEN Mü=0, Nü=1, Re=2)

	nach 0	nach 1	nach 2
von 0	-1	163	117
von 1	163	-1	80
von 2	117	80	-1

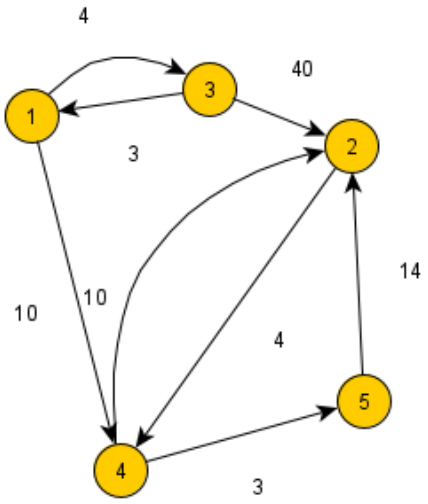


Besonderheiten:

- Das Gewicht -1 gibt oft an, dass keine Verbindung besteht. (Je nach Sachzusammenhang kann es auch sinnvoll sein dies mit dem Gewicht 0 auszudrücken.)
- Beim ungerichteten Graph ist die Adjazenzmatrix symmetrisch zur Diagonalen.
- Beim ungewichteten Graph wird jede KANTE die existiert mit Gewicht 1 angegeben
- Beim gerichteten Graph kann es die Verbindung von 0 zu 1 geben (z.B. mit Gewicht 15) aber keine Verbindung von 1 zu 0 (somit Gewicht -1).
- Jedes Element der Matrix wird mit einem Indexpaar identifiziert. In unserem Beispiel wäre von München(0) nach Regensburg(2) der Matrixeintrag in der rechten oberen Ecke und hätte den Bezeichner m_{02}.

Aufgaben:
1) Erstellen Sie den Graph zur gegebenen Adjazenzmatrix (links) und die Adjazenzmatrix zum Graphen auf der rechten Seite.

$$\begin{pmatrix} -1 & -1 & 5 & -1 \\ 3 & -1 & 3 & -1 \\ 5 & 5 & 2 & 3 \\ 1 & 2 & 2 & -1 \end{pmatrix}$$



Woran erkennt man, ob der Graph gerichtet oder gewichtet ist?

2) Kopieren Sie die Vorlage Graph_mit_Adjazenzmatrix_1 aus dem Klassenordner.

3) Implementieren Sie einen Graphen mit Adjazenzmatrix anhand folgender Checkliste: (Testen Sie ausführlich)

Implementierung eines Graphen mit einer Adjazenzmatrix (Checkliste)

Klasse KNOTEN:
- DATENELEMENT daten;
- daten im Konstruktor setzen
- evtl. Methode zum Ändern der Daten

Interface DATENELEMENT:
- Methode: String DatenGeben()
- Methode: void InformationAusgeben()

Testklasse STADT:
- Attribut name
- Konstruktor bei dem der name gleich gesetzt werden kann
- Umsetzung des Interfaces DATENELEMENT (inklusive Methoden implementieren)

Klasse GRAPH:
- Feld mit Knoten (Bezeichnung: knoten)
- zweidimensionales Feld matrix
- Methode: KnotenAnzeigen() (Konsolenausgabe)
- Methode: MatrixAnzeigen() (Konsolenausgabe
  Evtl. Zusatz: Schöne Anzeige, auch wenn die Gewichtung aus zwei Ziffern besteht

4) Überlegen Sie, wie eine Methode KnotenEinfügen und KanteEinfügen im GRAPH umgesetzt werden könnte. Lösen Sie zuerst das Problem der Vergrößerung der Matrix und des KNOTEN Feldes. Denken Sie daran, dass neue Knoten beim Einfügen erst mal keine Verbindung zu anderen KNOTEN haben sollten.