9.3 Graphen im Rechner

LK

Ein Bild ist für Menschen gemacht. Ein Programm braucht eine Datenstruktur. Für Graphen gibt es zwei gebräuchliche, und die Wahl zwischen ihnen ist eine echte Entwurfsentscheidung.

9.3.1 Die Adjazenzmatrix

Man nummeriert die Knoten durch und legt eine quadratische Tabelle an. In Zeile i, Spalte j steht, ob — und bei gewichteten Graphen: wie teuer — die Kante von Knoten i nach Knoten j ist.

	SB	NK	OTW	WND	HOM	MZG
SB	–	24	–	40	31	45
NK	24	–	8	–	15	–
OTW	–	8	–	10	–	–
WND	40	–	10	–	28	48
HOM	31	15	–	28	–	–
MZG	45	–	–	48	–	–

Zwei Dinge lassen sich hier sofort ablesen. Erstens ist die Matrix **symmetrisch zur Hauptdiagonalen** — genau das heißt „ungerichtet". Zweitens ist die Zahl der Einträge in einer Zeile der Knotengrad.

Stolperstein

Im Programm schreibt man statt des Strichs meist eine 0. Das geht so lange gut, wie kein Gewicht 0 vorkommt — sonst kann man „keine Straße" nicht mehr von „Straße der Länge 0" unterscheiden. Sauberer ist ein großer Platzhalterwert wie `UNENDLICH = 9999` oder eine zweite, rein logische Matrix daneben. Es ist dieselbe Falle wie bei den fehlenden Einträgen in der Übergangstabelle aus Abschnitt 5.6.

9.3.2 Die Adjazenzliste

Statt einer Tabelle führt man zu jedem Knoten eine Liste seiner Nachbarn:

```
  SB: NK (24), WND (40), HOM (31), MZG (45)
  NK: SB (24), OTW (8), HOM (15)
 OTW: NK (8), WND (10)
 WND: SB (40), OTW (10), HOM (28), MZG (48)
 HOM: SB (31), NK (15), WND (28)
 MZG: SB (45), WND (48)
```

In Python ist das ein Wörterbuch von Wörterbüchern — Knotenname auf Nachbarn, Nachbar auf Gewicht:

```python
GRAPH = {
    "SB":  {"NK": 24, "WND": 40, "HOM": 31, "MZG": 45},
    "NK":  {"SB": 24, "OTW":  8, "HOM": 15},
    "OTW": {"NK":  8, "WND": 10},
    "WND": {"SB": 40, "OTW": 10, "HOM": 28, "MZG": 48},
    "HOM": {"SB": 31, "NK": 15, "WND": 28},
    "MZG": {"SB": 45, "WND": 48},
}
```

In Pascal nimmt man die Matrix, weil sie sich als `array` direkt hinschreiben lässt:

```
const
  N = 6;
  UNENDLICH = 9999;
  ORTE: array [1..N] of String = ('SB', 'NK', 'OTW', 'WND', 'HOM', 'MZG');
  ENTFERNUNG: array [1..N, 1..N] of Integer = (
    {        SB   NK  OTW  WND  HOM  MZG }
    { SB }  ( 0,  24,   0,  40,  31,  45),
    { NK }  (24,   0,   8,   0,  15,   0),
    { OTW } ( 0,   8,   0,  10,   0,   0),
    { WND } (40,   0,  10,   0,  28,  48),
    { HOM } (31,  15,   0,  28,   0,   0),
    { MZG } (45,   0,   0,  48,   0,   0));
```

9.3.3 Welche Darstellung wann

Die Entscheidung hängt daran, wie **dicht** der Graph ist — also wie nah die Kantenzahl an der maximal möglichen liegt.

	Adjazenzmatrix	**Adjazenzliste**
Speicher	n^2 Plätze, unabhängig von der Kantenzahl	$n + 2m$ Plätze bei m Kanten
„Gibt es Kante $a \to b$?"	ein Zugriff	Liste von a durchsuchen
„Alle Nachbarn von a"	ganze Zeile prüfen, n Schritte	direkt da, $\deg(a)$ Schritte
geeignet für	dichte Graphen	dünne Graphen

Straßennetze, soziale Netze und das Internet sind extrem **dünn**: Kein Ort hat tausend Nachbarorte, kein Mensch Millionen Freunde. Für einen Graphen mit einer Million Knoten bräuchte die Matrix 10^{12} Plätze — undenkbar. Deshalb arbeitet man in der Praxis fast immer mit Adjazenzlisten.

Arbeitsauftrag 3

11. Geben Sie die Adjazenzmatrix des Königsberger Graphen an. Was steht dort, wo zwei Brücken dasselbe Paar verbinden?
12. Zeichnen Sie den Graphen zu dieser Adjazenzmatrix. Ist er gerichtet oder ungerichtet?

```
    A  B  C  D
A   0  1  1  0
B   0  0  1  1
C   0  0  0  1
D   0  0  0  0
```

13. Schreiben Sie eine Funktion, die zu einer Adjazenzmatrix prüft, ob der Graph ungerichtet ist.
14. Schreiben Sie eine Funktion, die aus einer Adjazenzmatrix die Adjazenzlisten erzeugt.
15. Ein sozialer Dienst hat 50 Millionen Mitglieder mit durchschnittlich 300 Kontakten. Rechnen Sie den Speicherbedarf beider Darstellungen aus, wenn ein Eintrag 4 Byte belegt.
16. Begründen Sie, warum die Adjazenzmatrix eines schlichten Graphen auf der Hauptdiagonalen nur Nullen enthält.