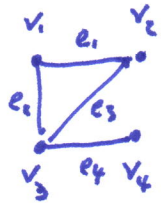
3.6 Datenstrukturen für Graphen

Wie beschreibt man einen Graphen?

(1) Inzidenzmatrix



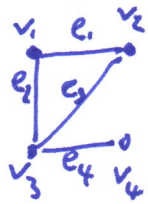
$$\begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline v_1 & 1 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 0 \\ v_3 & 0 & 1 & 1 & 1 \\ v_4 & 0 & 0 & 0 & 1 \end{array}$$

(„inzident": zusammentreffend)

Also: $A \in \{0,1\}^{n \times m}$ mit $a_{v,e} := \begin{cases} 1 & \text{für } v \in e \\ 0 & \text{sonst} \end{cases}$

Größe: nm für einen Graphen mit n Knoten, m Kanten. (Viele Nullen!)

(2) Adjazenzmatrix



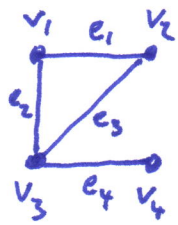
$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 1 & 0 \\ v_2 & 1 & 0 & 1 & 0 \\ v_3 & 1 & 1 & 0 & 1 \\ v_4 & 0 & 0 & 1 & 0 \end{array}$$

(„adjazent": verbunden)

Also: $A \in \{0,1\}^{n \times n}$ mit $a_{v,w} := \begin{cases} 1 & \text{für } \{v,w\} \in E \\ 0 & \text{sonst} \end{cases}$

Größe: n^2 für einen Graphen mit n Knoten.

(3) Kantenliste



$\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_3, v_4\}$

Benötigt wird eine Kantennummerierung!

→ Jeder Index ist eine Binärzahl mit $(\log_2 n + 1)$ Bits, bzw. eine Dezimalzahl mit $(\log_{10} n + 1)$ Stellen.

Unterschied in Codierungslänge: Ein Faktor $\log_2 10 = 3{,}3219\ldots$, denn $\log_2 n = \log_2 10 \cdot \log_{10} n$

Für einen Graphen mit m Kanten und n Knoten ergibt sich in obiger (ausführlicher) Codierung ein Platzbedarf von

$$(6m-1) + 2m(\log_{10} n + 1).$$

Dabei kann man an ein paar Stellen sparen (z.B. „v" oder „{", „}" weglassen) aber auch mehr Platz investieren (z.B. in ASCII codieren bzw. binär statt dezimal codieren). So wäre auch

$$(2m-1) + 2m(\log_2 n + 1) \qquad \text{denkbar.}$$

Was ist wirklich wichtig dabei?!

(i) Die Kantenliste ist sparsamer als die Inzidenzmatrix, denn wenn n nicht zu klein ist, dann ist

$$n \geq 2 + 2(\log_2 n + 1).$$

(was heißt „nicht zu klein"? $n \geq 8$ reicht!)

Also ist auch

$$mn > (2m-1) + 2m(\log_2 n + 1).$$

(ii) Unabhängig von der Codierung ist für die Größe des Speicherplatzes der zweite Ausdruck wichtig, denn

$$2m(\log_2 n + 1) \leq (2m-1) + 2m(\log_2 n + 1)$$
$$\leq 4m(\log_2 n + 1) \qquad \text{für } n \geq 2.$$

(iii) Letztlich kommt es also gar nicht so sehr auf die Vorfaktoren an (die sind codierungsabhängig!), sondern auf den Ausdruck

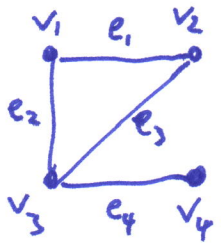
$$m \log n.$$

(iv) In $m \log n$ steckt das Wesen der Kantenliste:
Zähle für alle m Kanten die Nummern der beteiligten Knoten auf!

Wie wir im nächsten Abschnitt sehen werden, lohnt sich dafür eine eigene Notation:

Die Kantenliste benötigt $\Theta(m \log n)$ Speicherplatz.

(4) Adjazenzliste



v_1 : v_2, v_3 ;
v_2 : v_1, v_3 ;
v_3 : v_1, v_2, v_3 ;
v_4 : v_4 ;

Das ist etwas praktischer als die Kantenliste, wenn man für Graphenalgorithmen direkten Zugriff auf die Nachbarn eines Knotens benötigt! Man muss nicht die Nachbarn erst mühsam aus einer Liste herausschreiben.

Länge:

Jede Kante taucht doppelt auf, einmal für jeden Knoten.

So also:

$$2n + 4m + n(\log_{10} n + 1) + 2m(\log_{10} n + 1)$$

– d.h. $\Theta(n \log n + m \log n)$.

Im allgemeinen sind Graphen mit vielen isolierten Knoten (ohne Kanten!) uninteressant, d.h. z.B. $m \geq \frac{n}{2}$

$m \geq n$ o.ä.

Also wieder $\Theta(m \log n)$.