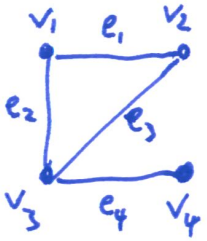
(4) Adjazenzliste



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

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 heraussuchen.

Länge:

Jede Kante taucht doppelt auf, einmal für jeden Knoten.

So also:

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

– 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 $\Theta(m \log n)$.

Jetzt wollen wir noch etwas mehr: den direkten Zugriff auf die Nachbarn der Knoten, wenn wir sie brauchen!

Also:

Liste (!) $\quad \underset{\downarrow}{v_2}, \underset{\downarrow}{v_3}; \underset{\downarrow}{v_1}, \underset{\downarrow}{v_3}; \underset{\downarrow}{v_1}, \underset{\downarrow}{v_2}, \underset{\downarrow}{v_4}; \underset{\downarrow}{v_3}$

Zugriff auf Nachbarn jeweils ab Semikolon.

Algorithmisch: Gehe Liste durch, zähle Semikolons → das dauert

Datenstruktur: Speichere die Stelle ab, an der die Nachbarn von v_3 zu finden sind.

→ Zeiger (engl. "Pointer")

Bekannt bei Webseiten:

Speicherinhalt: z.B. YouTube-Video (etliche MB)

Zeiger: URL (einige Byte)

„Man muss nicht alles wissen, man muss nur wissen wo's steht!"

Für den direkten Zugriff brauchen wir n Zeiger; jeder codiert eine Speicherzelle, d.h. die Nummer eines Bits in der Liste.

So ein Zeiger braucht also selber

$$\log_2 \left\lfloor \left(2n + 4m + n\left(\lfloor \log_2 n \rfloor + 1\right) + 2m\left(\lfloor \log_2 n \rfloor + 1\right)\right)\right\rfloor + 1$$

für $n \geq 10$, $m \geq n$

$$\leq \log_2 \left(9m\left(\log_2 n + 1\right) + 1\right)$$

$$\leq \log_2 9 + \log_2 m + \log_2\left(\log_2 n + 1\right) + 1 \leq 2\log_2 m$$

Bits

Insgesamt benötigen wir also

$$\Theta(n \log_2 m) \quad \text{Bits.}$$

Jetzt ist $m \leq n^2$,

also $\log_2 m \leq \log_2 n^2 = 2 \log_2 n$,

d.h. $n \log m \leq 2n \log n$,

und der insgesamt verbrauchte Speicherplatz ist

$$\Theta(n \log m + m \log n) = \Theta(m \log n).$$