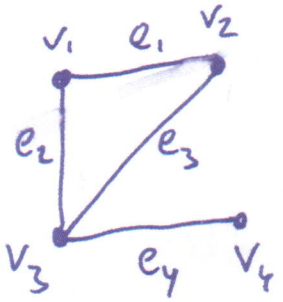
24.11.09

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 herraussuchen.

Länge: Jede Kante taucht doppelt auf, einmal für jeden Knoten

So also: $2n + 4m + n \log_{10} n + 2m \log_{10} n$

- d.h. $\Theta(n \log n + m \log n)$.

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

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

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

Also:

Liste: $v_2, v_3; v_1, v_3; \underline{v_1, v_2, v_3}; v_4$ (↓ auf v_3)

Zugriff auf Nachbarn von v_3 ab zweitem Semikolon!

Algorithmisch: Gehe Liste durch, zähle Semikolons → das kann dauern!

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

↳ Zeiger (oder auch „Pointer")

Unterschied bei Webseiten:

Speicherinhalt: z.B. Video auf YouTube (etliche Megabyte)

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
der Liste

So ein Zeiger braucht also selber

$$\log_2\left(2n + 4m + n\log_{10} n + 2m\log_{10} n\right)$$

$$\leq \log_2\left(9m\log_{10} n\right) \leq \log_2 9m + \log_2\log_{10} n$$

für $n \geq 10$, $m \geq n$

$$\leq \log_2 9m + \log_2 \log_{10} n$$

$$\leq \log_2 9 + \log_2 m + \log_2 \log_{10} n$$

$$\leq 2 \log_2 m \quad \text{Bits}$$

- also insgesamt

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

Jetzt ist

$$m \leq n^2,$$

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

d.h. $n \log_2 m \leq 2n \log_2 n$,

und der insgesamt benötigte Speicherplatz ist

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

3.5 Wachstum von Funktionen

Im letzten Abschnitt haben wir Funktionen abgeschätzt und auf ihre „wesentlichen" Bestandteile reduziert, um Größenordnungen und Wachstumsverhalten zu beschreiben.

Ein bisschen formaler:

Definition 3.9 (Θ-Notation)

Seien $f, g : \mathbb{N} \to \mathbb{R}$ Funktionen.

Dann gilt

$f \in \Theta(g) :\Leftrightarrow$ Es gibt positive Konstanten c_1, c_2, n_0 mit

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ für alle $n \geq n_0$.

Man sagt: f wächst asymptotisch in derselben Größenordnung wie g.

Beispiel: $2n^2 - 1 \in \Theta(n^2)$

$\frac{n^3}{1000} + n^2 + n \log n \in \Theta(n^3)$

Manchmal hat man nur obere oder untere Abschätzungen:

Definition 3.10 (O-Notation)

Seien $f, g : \mathbb{N} \to \mathbb{R}$ Funktionen

Dann gilt

$f \in O(g) :\Leftrightarrow$ Es gibt positive Konstanten c, n_0 mit

$0 \leq f(n) \leq c g(n)$ für alle $n \geq n_0$.

Man sagt: f wächst asymptotisch höchstens in derselben Größenordnung wie g.