Beispiele:

$$2n^2-1 \in O(n^2)$$
$$2n^2-1 \in O(n^3)$$
$$n \log n \in O(n^2)$$

Definition 3.11 (Ω-Notation)

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

Dann gilt $f \in \Omega(g) :\Longleftrightarrow$ Es gibt positive Konstanten c, n_0 mit
$0 \leq c\,g(n) \leq f(n)$ für alle $n \geq n_0$.

Einige einfache Eigenschaften:

Satz 3.12

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

Dann gilt

(i) $f \in \Theta(g) \Longleftrightarrow g \in \Theta(f)$

(ii) $f \in \Theta(g) \Longleftrightarrow f \in O(g)$ und $f \in \Omega(g)$

(iii) $f \in O(g) \Longleftrightarrow g \in \Omega(f)$

Beweis:

Übung!

3.6 Laufzeit von BFS und DFS

Wenn man einen Graphen durchsucht, sollte man schon alle Knoten und alle Kanten ansehen; also lässt sich eine untere Schranke von $\Omega(n+m)$ nicht unterbieten.

Tatsächlich wird diese Schranke auch erreicht:

Satz 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Beweis:

Wir nehmen an, dass G durch eine Adjazenzliste gegeben ist.

Für jeden Knoten x verwenden wir einen Zeiger, der auf die „aktuelle" Kante für diesen Knoten in der Liste zeigt (d.h. auf den „aktuellen" Nachbarn). Anfangs zeigt $akt(x)$ auf das erste Element in der Liste.

In (4) wird ~~der~~ aktuelle Nachbar ausgewählt und der Zeiger weiterbewegt; wird das Listenende erreicht, wird x aus Q entfernt und nicht mehr eingefügt.

Also ergibt sich eine Gesamtlaufzeit von $O(n+m)$. □

Korollar 3.14

Mit Algorithmus 3.7 kann man alle Zusammenhangskomponenten eines Graphen bestimmen.

Beweis:

Wende 3.7 einmal an und überprüfe, ob $R = V$ ist.
Falls ja, ist der Graph zusammenhängend.
Falls nein, haben wir eine Zusammenhangskomponente identifiziert; wir lassen den Algorithmus erneut für einen beliebigen Knoten $s' \in V \setminus R$ laufen usw.

Wieder wird keine Kante doppelt angefasst, also bleibt die Gesamtzeit linear, d.h. $O(n+m)$

$\square$

Korollar 3.15

BFS und DFS haben Laufzeit $O(n+m)$.

Beweis:

Einfügen in Q lässt sich jeweils in konstanter Zeit vornehmen, der Rest überträgt sich von Satz 3.13

$\square$

3.7 Besondere Eigenschaften von DFS und BFS

Einfach gesagt:

- DFS ist eine bestmögliche, individuelle Suchstrategie mit lokaler Information.

- BFS ist eine bestmögliche, kooperative Suchstrategie mit globaler Information.

Konkret:

- DFS ist gut geeignet für die Suche nach einem Ausweg aus einem Labyrinth.

- BFS ist gut geeignet für die Suche nach kürzesten Wegen in einem Graphen

Satz 3.16 (Lokale Suche mit DFS)

DFS ist eine optimale lokale Suchstrategie in folgendem Sinne:

(1) DFS findet in jedem Graphen mit n Knoten einen Weg der Länge höchstens $2n-1$, der alle Knoten besucht.

(2) Für jede lokale Suchstrategie gibt es einen Graphen mit n Knoten, so dass der letzte Knoten erst nach einer Weglänge von $2n-1$ besucht wird.