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

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

3.7 Wachstum von Funktionen

Im letzten Abschnitt haben wir Funktionen abgeschätzt und auf die „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

$$f \in \Theta(g) :\Leftrightarrow \text{Es gibt positive Konstanten } c_1, c_2, n_0 \text{ mit } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ für alle } n \geq n_0.$$

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

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

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

Manchmal hat man nur untere oder obere Abschätzungen:

DEFINITION 3.10 (O-Notation)

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

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

$$0 \leq f(n) \leq c\, g(n) \quad \text{für alle } n \geq n_0.$$

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

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

$$0 \leq c\, g(n) \leq f(n) \quad \text{für alle } n \geq n_0.$$

Beispiele: $2n^2-1 \in \Omega(n^2)$
$2n^2-1 \in \Omega(n^3)$

Einige einfache Eigenschaften:

SATZ 3.12

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

Dann gilt
(1) $f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$
(2) $f \in \Theta(g) \Leftrightarrow f \in O(g)$ und $f \in \Omega(g)$.
(3) $f \in O(g) \Leftrightarrow g \in \Omega(f)$

Beweis: Übung!

3.8 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 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 2.3.1 wird der aktuelle Nachbar ausgewählt und der Zeiger weiterbewegt; wird das Listenende erreicht, wird x aus R entfernt und nicht mehr eingeführt.
Also ergibt sich eine Gesamtlaufzeit von $O(n+m)$. $\square$

KOROLLAR 3.14

Mit Algorithmus 3.7 kann man alle Zusammenhangskomponenten eines Graphen berechnen.

Beweis:

Wende 3.7 an und überprüfe, ob $Y=V$ ist. Falls ja, ist der Graph zshgd.
Falls nein, haben wir eine Zusammenhangskomponente berechnet; wir lassen den Algorithmus erneut für einen Knoten $s' \in V \setminus Y$ laufen, usw.
Wieder wird keine Kante von einem Knoten aus doppelt angefasst, also bleibt die Laufzeit linear, d.h. $O(n+m)$. $\square$

Korollar 3.15
BFS und DFS haben Laufzeit $O(n+m)$.

Beweis:
Einfügen in R (Warteschlange bzw. Stapel) lässt sich jeweils in konstanter Zeit vornehmen, der Rest überträgt sich von Satz 3.13 $\square$

3.9 Besondere Eigenschaften von BFS und DFS

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 ~~kürzesten~~

Satz 3.16 (Lokale Suche mit DFS)
DFS ist eine optimale lokale Suchstrategie in folgendem Sinne:

(1) DFS findet in jedem zusammenhängenden 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.

Beweis: Übung!