## 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.

Beweis: Übung!

Für BFS zeigen wir, dass der zugehörige Baum tatsächlich kürzeste Wege im Graphen liefert. Für leichtere Argumentation gebrauchen wir dabei folgende leichte Modifikation von Algorithmus 3.7:

**Algorithmus 3.17**

Input: Graph $G=(V,E)$, Knoten $s$

Output:
- Knotenmenge $R \subseteq V$, die von $s$ aus erreichbar ist;
- für jeden Knoten $v \in R$ die Länge $d(s,v)$ eines kürzesten Weges von $s$ nach $v$;
- eine Kantenmenge $T \subseteq E$, die die ~~Erreichbarkeit~~ kürzestmöglichen Wege zu den Knoten von $R$ sicherstellt, d.h. einen die Zusammenhangskomponente von $s$ aufspannenden Baum $(R,T)$, der kürzeste Wege von $s$ zu allen Knoten in $R$ liefert.

(1) Sei $R := \{s\}$, $Q := \{s\}$, $T := \emptyset$, $\ell(s) := 0$,

(2) WHILE $(Q \neq \emptyset)$ DO {

wähle erstes Element $v \in Q$

(3) IF (es gibt kein $w \in V \setminus R$ mit $e = \{v, w\} \in E$ THEN

$Q := Q \setminus \{v\}$

(4) ELSE {

wähle ein $w \in V \setminus R$ mit $e = \{v, w\} \in E$;

setze $R := R \cup \{w\}$, $T := T \cup \{e\}$, hänge $w$ an $Q$ an;

setze $\ell(w) := \ell(v) + 1$.

}

}

(5) STOP

## Satz 3.18

(i) Verfahren 3.17 ist ein Algorithmus.

(ii) Die Laufzeit ist $O(n+m)$.

(iii) Am Ende ist für jeden erreichbaren Knoten $v \in R$ die Länge eines kürzesten Weges von $s$ nach $v$ im Baum $(R,T)$ durch $\ell(v)$ gegeben.

(iv) Am Ende ist für jeden erreichbaren Knoten $v \in R$ die Länge eines kürzesten Weges von $s$ nach $v$ im Graphen $(V,E)$ durch $\ell(v)$ gegeben.

## Beweis:

(i) Wie für Algorithmus 3.7 gelten alle Eigenschaften; zusätzlich ist für jeden Knoten $w \in Q$ per Induktion der Wert $\ell(v)$ tatsächlich definiert.

(ii) Die Laufzeit bleibt von Algorithmus 3.7 erhalten.

(iii) Sei $d_{(R,T)}(s,v)$ die Länge eines kürzesten Weges von $s$ nach $v$ in $(R,T)$. Dann zeigt man ~~über~~ durch Induktion über $d_{(R,T)}(s,v)$, dass für alle Knoten $d_{(R,T)}(s,v) = \ell(v)$ gilt:

Induktionsanfang:

$$d_{(R,T)}(s,v) = 0 \quad \text{gilt genau für } v = s,$$

und $\ell(s) = 0$.

Induktionsannahme:

Sei $d_{(R,T)}(s,v) = \ell(v)$ für alle

$v \in V$ mit $d_{(R,T)}(s,v) \leq k-1$.

Induktionsschritt:

Sei $w \in V$ ein Knoten mit

$$d_{(R,T)}(s,w) = k.$$

Dann gibt es im Baum $(R,T)$ einen eindeutigen Weg von $s$ zu $w$; sei $v$ der Vorgänger von $w$ in diesem Weg, also $\{v,w\} \in T$.

Nach Induktionsannahme gilt

$$d_{(R,T)}(s,v) = \ell(v);$$

außerdem ist

$$d_{(R,T)}(s,w) = d_{(R,T)}(s,v) + 1$$

und $\ell(w) = \ell(v) + 1$,

also $d_{(R,T)}(s,w) = \ell(w)$, und die Behauptung gilt.