SATZ 3.8

(1) Das Verfahren 3.7 ist endlich.
(2) Das Verfahren 3.7 funktioniert korrekt.

Beweis:

(1) Bei jedem Durchlauf der Schleife 2. wird entweder in 2.2.1 ein Element aus R entfernt, oder in 2.3.2 ein Element zu R hinzugefügt und auch zu Y hinzugefügt. Also kann man 2.3.2 nur (n-1)-mal durchlaufen, also auch nur R (n-1)-mal erweitern, also R höchstens n-mal verkleinern. 2. kann also höchstens (2n-1)-mal durchlaufen werden.

(2) Zu jedem Zeitpunkt ist $(Y;T)$ ein s enthaltender Baum, denn
(a) alle Knoten in G sind von s aus erreichbar (und umgekehrt)
(b) neu eingefügte Kanten verbinden die bisherige Knotenmenge Y nur mit bislang nicht erreichbaren Knoten, können also keinen Kreis schließen.

Jetzt müssen wir noch zeigen, dass alle erreichbaren Knoten auch korrekt identifiziert werden.

Angenommen, am Ende gibt es einen Knoten $w \in V \setminus Y$, der in G von s aus erreichbar ist.

Sei P ein s-w-Pfad in G, und sei $\{x,y\}$ eine Kante von P mit $x \in Y$, $y \notin Y$.

Da x zu Y gehört, wurde x auch zu R hinzugefügt. Der Algorithmus stoppt aber nicht, bevor x aus R entfernt wurde. Das wird in 2.2.1 nur vorgenommen, wenn es in 2.2 keine Kante $\{x,y\}$ mit $y \notin Y$ gibt – im Widerspruch zur Annahme. $\square$

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!