17.11.09

① Sei $R := \{s\}$, $Q := \{s\}$, $T := \emptyset$.

② WHILE $(Q \neq \emptyset)$ DO {

wähle $v \in Q$

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

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

④ ELSE {

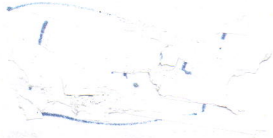
wähle ein $w \in V \setminus R$ mit $e = \{v,w\} \in E$

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

}

}

⑤ STOP.



Satz 3.8

(i) Das Verfahren 3.7 ist endlich.

(ii) Das Verfahren 3.7 funktioniert korrekt.

Beweis:

(i) Bei jedem Durchlauf der Schleife ② wird entweder in ③ ein Element aus Q entfernt, oder in ④ ein Element zu R hinzugefügt und ein Element zu Q hinzugefügt. Offenbar kann man also ④ nur $(n-1)$-mal durchlaufen, also auch nur Q $(n-1)$-mal erweitern, also Q höchstens n-mal verkleinern. ② kann also höchstens $(2n-1)$-mal durchlaufen werden.

(ii) Zu jedem Zeitpunkt ist (R,T) ein s enthaltender Baum, denn (a) alle Knoten in R sind von s aus erreichbar (und umgekehrt)

(b) neu eingefügte Kanten verbinden die bisherige Knotenmenge R nur mit bislang nicht erreichbaren Knoten, können also keinen Kreis schließen.

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

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

Da x zu R gehört, wurde x auch zu Q hinzugefügt. Der Algorithmus stoppt aber nicht, bevor x aus Q entfernt wurde. Dies wird aber in (3) nur vorgenommen, wenn es keine Kante $\{x,y\}$ mit $y \notin R$ gibt

– im Widerspruch zur Annahme. □

3.3 Tiefensuche, Breitensuche; Stapel, Warteschlange

Wie wird in (2) die Auswahl eines Knotens $v \in Q$ vorgenommen?

Beobachtung: Die Art, wie Q abgespeichert und abgearbeitet wird, hat eine unmittelbare Auswirkung auf den Ablauf des Algorithmus!

Zwei häufigste Möglichkeiten:

(B) Die zuerst aufgenommenen Knoten werden zuerst abgearbeitet, d.h. die Suche geht jeweils erst einmal in die Breite, bevor weiter entfernte Knoten erledigt werden.

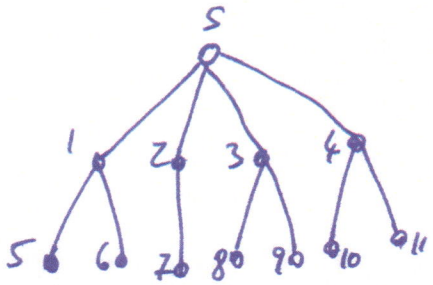
→ „Breitensuche" oder Breadth-First Search (BFS)

(D) Die zuletzt aufgenommenen Knoten werden zuerst abgearbeitet, d.h. die Suche geht jeweils erst einmal in die Tiefe, bevor weitere Knoten und deren Nachfolger erledigt werden.

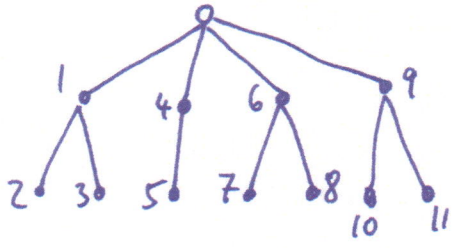
→ „Tiefensuche" oder Depth-First Search (DFS)

Bildlich: Selbst derselbe Baum wird unterschiedlich abgearbeitet:

(B)



(D)



Für Kompliziertere Graphen ist das noch deutlicher

- ein BFS-Baum ist tendenziell „breit und flach",
  ein DFS-Baum dagegen „schmal und tief".

Anwendungen:

(B) Breitensuche liefert kürzeste Wege von einer Quelle aus (→ später mehr)

(D) Tiefensuche liefert eine zusammenhängende Suche nach einem versteckten Objekt (→ z.B. Weg aus Labyrinth)

Etwas pointierter (und nicht zu ernst zu nehmen):

(B) Parallele, „kooperative" Strategie (viele Sammler bzw. Frauen)

(D) Individuelle Strategie (ein Jäger bzw. Mann)