Fachbereich IV - Abteilung Informatik
Algorithmen und Datenstrukturen
Universität Trier

Prof. Dr. S. Näher, Tel.: 0651 201-3275

2. Übung zur Vorlesung:

Algorithmen und Komplexität

Wintersemester 2009/10 13. November 2009

Aufgabe 2.1:

Zeigen Sie, dass für den in der Vorlesung behandelten Algorithmus zur Berechnung der starken Zusammenhangskomponenten stets folgende Invarianten erfüllt sind:

a) Es existiert keine Kante $(v, E) \in E$ mit v in abgeschlossener SZK und w in nicht-abgeschlossenener SZK.

b) Alle nicht-abgeschlossenen SZK liegen auf einem Pfad, genauer: alle Wurzeln von nicht abgeschlossenenen SZK liegen auf einem Baumpfad.

c) Die Knoten jeder nicht-abgeschlossenen SZK bilden ein Intervall der Folge $unfertig$.

Aufgabe 2.2:

Sei $G = (V, E)$ ein gerichteter Graph, betrachten Sie folgenden Algorithmus.

a) Berechne eine completion-Nummerierung (*compnum*) von G mit DFS.

b) Drehe alle Kanten in G um.

c) Markiere alle Knoten als nicht besucht.

d) Durchlaufe die Knoten in absteigender *compnum*-Reihenfolge und rufe für jeden noch nicht besuchten Knoten v die rekursive Funktion ($dfs(v)$) auf.

Zeigen Sie, dass jeder Aufruf von dfs in Zeile 3 des Algorithmus genau eine starke Zusammenhangskomponente von G durchläuft.

Aufgabe 2.3:

Ein Graph $G = (V, E)$ heißt *bipartit*, wenn man V in zwei disjunkte Teilmengen A und B zerlegen kann, so dass $E \subseteq A \times B \cup B \times A$, d. h. jede Kante führt von einem Knoten aus A zu einem Knoten aus B oder umgekehrt.
Entwickeln Sie einen effizienten Algorithmus, der testet, ob ein gegebener Graph bipartit ist.
Hinweis: Verwenden Sie BFS (Breitensuche).