Universität Konstanz
Lehrstuhl für Praktische Informatik
Prof. Dr. U. Brandes / J. Lerner / B. Schlieper

Entwurf und Analyse von Algorithmen
WS 2005/2006

4. Übungsblatt

Ausgabe: 11.11.2005 **Abgabe:** 18.11.2005, 10 Uhr
Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Aufgabe 10: 3 Punkte

Geben Sie den Pseudocode für Find mit Pfadkompression an. (Orientieren Sie sich z. B. am Pseudocode für Find ohne Pfadkompression aus der Vorlesung.)

Aufgabe 11: 6 Punkte

Eine Folge von n Operationen Makeset, Find (mit Pfadkompression) und Union (gewichtet) habe die spezielle Eigenschaft, dass alle Union-Operationen vor der ersten Find-Operation ausgeführt werden.

(a) Zeigen Sie, dass der amortisierte Gesamtaufwand für die n Operationen in $\mathcal{O}(n)$ liegt.

(b) Ist (a) korrekt, wenn zwar die Find-Operationen weiterhin mit Pfadkompression, die Union-Operationen aber ungewichtet ausgeführt werden?

(c) Ist (a) korrekt, wenn die Union-Operationen wieder gewichtet, die Find-Operationen jetzt aber ohne Pfadkompression ausgeführt werden?

Aufgabe 12: 5 Punkte

Eine alternative Pfadkompressions-Heuristik sei wie folgt definiert: Nachdem der Pfad von einem Element bis zu seiner Wurzel durchlaufen ist, erhält jedes Element auf diesem Pfad den Vorgänger seines Vorgängers als Vorgänger. Ist z. B.

$$\cdots \to i \to j \to k \to l \to \cdots$$

ein Teil des Pfades vor der Pfadkompression, so wird bei der alternativen Pfadkompression k der Vorgänger von i und l der Vorgänger von j. Die direkten Nachfolger der Wurzel behalten die Wurzel als Vorgänger.

(a) Gehen Sie den Beweis des Satzes von Hopcroft & Ullman durch. An welchen Stellen geht ein, dass Find mit Pfadkompression verwendet wird? Entscheiden Sie: Ist der Beweis immer noch gültig, wenn Find mit der alternativen Pfadkompression durchgeführt wird?

[Bitte wenden]

(b) Zeigen oder widerlegen Sie ob Aufgabe 11(a) weiterhin korrekt ist, wenn die alternative Pfadkompression verwendet wird.

Aufgabe 13: **6 Punkte**

Ergänzen Sie das folgende Code-Fragment zu einem Algorithmus der – bei Eingabe eines Graphen – eine Union-Find Datenstruktur aufbaut, deren Elemente die Knoten des Graphen sind und in der zwei Knoten u und v genau dann in der selben Menge sind, wenn u und v durch einen Pfad im Graphen verbunden sind (d. h. wenn u und v in der selben Zusammenhangskomponente des Graphen sind).

```
Input: Ein ungerichteter Graph G = (V, E)
begin
    foreach v ∈ V do
        ...
    foreach {u, v} ∈ E do
        ...
        ...
end
```

Beweisen Sie die Korrektheit Ihres Algorithmus indem Sie eine geeignete Invariante angeben.

Ein Graph $G = (V, E)$ habe n Knoten, m Kanten und k Zusammenhangskomponenten. Wie oft (als Funktionen von n, m und k) werden in Ihrem Algorithmus bei Eingabe von G die Methoden MAKESET, UNION und FIND aufgerufen? Hier sind ausnahmsweise einmal die exakten Anzahlen (d. h. nicht in asymptotischer Notation) gefragt.