Institut für Informatik I
Prof. Dr. Heiko Röglin
Tobias Brunsch

universitätbonn

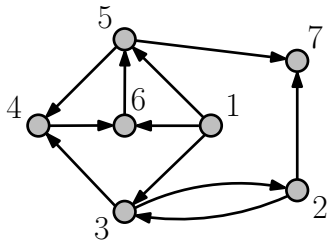
Algorithmen und Berechnungskomplexität I
Wintersemester 2010/11
Abgabe: 01.12.2010 in der Vorlesung

Übungsblatt 7

Aufgabe 7.1

3+4 Punkte

(a) Führen Sie eine Tiefensuche auf dem unten abgebildeten Graphen G durch, bei der Nachbarn mit kleinerer Nummer stets zuerst besucht werden sollen. Geben Sie dafür den ***D**epth-**F**irst-**S**earch*-Baum an, der entsteht, wenn Sie bei Knoten 1 beginnen. Ordnen Sie, wie in der Vorlesung beschrieben, jede Kante ihrer Klasse (T-, F-, C- oder B-Kante) zu.



(b) Eine Tiefensuche teilt die Kanten eines gerichteten Graphen in T-, F-, C- und B-Kanten ein. Überlegen Sie sich, wie man mit Hilfe dieser Einteilung entscheiden kann, ob der Graph einen gerichteten Kreis enthält. Entwickeln Sie daraus einen Algorithmus, der in einem gegebenen Graphen einen Kreis bestimmt oder „kreisfrei“ ausgibt, falls der Graph keinen Kreis enthält. Ein Beweis ist nicht nötig.

Aufgabe 7.2

4+3+4 Punkte

Eine Funktion $t\colon V \longrightarrow \{1, \ldots, |V|\}$ auf den Knoten eines gerichteten Graphen $G = (V, E)$ heißt *topologische Sortierung*, falls für alle Kanten (v, w) von G gilt: $t(v) < t(w)$.

(a) Betrachten Sie den folgenden Algorithmus zur Ermittlung einer topologischen Sortierung.

```
count = 0
while G nicht leer do
  Finde einen Knoten v in G, dessen Ingrad 0 beträgt
  if v = nil then return
  count = count +1
  t(v) = count
  Entferne v und alle Kanten der Gestalt (v, w) aus G
end while
```

Beweisen Sie, dass obiger Algorithmus eine *topologische Sortierung* von G erzeugt, falls G keinen gerichteten Kreis besitzt.

Hinweis: Zeigen Sie zunächst, dass jeder kreisfreie Graph einen Knoten mit Ingrad 0 besitzt. Dazu können Sie die Kontraposition dieser Aussage zeigen: Wenn der Ingrad jedes Knotens mindestens 1 ist, dann besitzt der Graph einen gerichteten Kreis. Überlegen Sie sich einen konstruktiven Beweis dieser Aussage.

(b) Beweisen Sie, dass G genau dann eine topologische Sortierung besitzt, wenn G keinen gerichteten Kreis enthält.

(c) Geben Sie einen Algorithmus an, der in Zeit $O(|V| + |E|)$ eine topologische Sortierung von G bestimmt. Orientieren Sie sich dabei an Aufgabenteil (a). Der Graph sei in Adjazenzlistendarstellung gegeben und die Knoten seien mit $1, \ldots, n$ durchnummeriert.

Hinweis: Benutzen Sie ein Feld d, das zu jedem Zeitpunkt für jeden Knoten den aktuellen Ingrad speichert.

Aufgabe 7.3

6 Punkte

Wir betrachten die erste *Union-Find*-Datenstruktur aus der Vorlesung, d.h. ein n-elementiges Feld A, in dem für jedes Element $i \in \{1, \ldots, n\}$ gespeichert ist, zu welcher Menge es gehört. Zusätzlich ist für jede Menge ihre Größe und eine Liste der Elemente der Menge gegeben.

In der Vorlesung wurde gezeigt, dass die Laufzeit für jede Folge von *Union*-Befehlen auf n Elementen höchstens $O(n \log n)$ ist. Wie sieht eine Folge solcher *Union*-Befehle auf n Elementen aus, deren Laufzeit wirklich $\Theta(n \log n)$ beträgt? Beweisen Sie Ihre Aussage.

Aufgabe 7.4

6 Zusatzpunkte

Wir wollen für ein Feld A mit n Einträgen drei Funktionen `Init`, `PrintSmallerValues` und `PrintSmallest` bereitstellen. Die Funktion `Init` wird einmal am Anfang aufgerufen und dient als Vorverarbeitungsschritt. Das Ausführen dieser Funktion soll in Zeit $O(n)$ möglich sein. Ein Aufruf `PrintSmallerValues`(z) soll in Zeit $O(k)$ alle Einträge von A ausgeben, die kleiner oder gleich z sind. Die Zahl k sei dabei die Anzahl dieser Einträge (oder 1, falls alle Einträge größer als z sind). Ein Aufruf `PrintSmallest`(k) für $1 \leq k \leq n$ soll in Zeit $O(k)$ die k kleinsten Einträge von A ausgeben.

Stünde der `Init`-Funktion Zeit $O(n \log n)$ zur Verfügung, dann könnte man das Feld mit *Quicksort* und optimierter Pivotisierung sortieren. In `PrintSmallerValues` und `PrintSmallest` würde man dann das sortierte Feld von links nach rechts durchgehen und an geeigneter Stelle stoppen.

Implementieren Sie die `Init`-, die `PrintSmallerValues`- und die `PrintSmallest`-Funktion. Gehen Sie dabei ähnlich vor wie in obiger Idee. Begründen Sie die Korrektheit Ihrer Implementierungen und erklären Sie, warum die Laufzeitrestriktionen eingehalten werden.

Hinweis: Nutzen Sie die Tatsache, dass das Bestimmen des k-t kleinsten Eintrages eines Feldes in Linearzeit (in der Feldlänge) möglich ist, auch wenn k Teil der Eingabe ist.