Datenstrukturen (DS)

Sommersemester 2015

Dipl-Inf. Bert Besser
Prof. Dr. Georg Schnitger
Hannes Seiwert, M.Sc.

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Institut für Informatik
AG Theoretische Informatik

Übung 3

Ausgabe: 19.05.2015
Abgabe: 02.06.2015

Auf diesem Blatt können Sie 6 **Extrapunkte** erreichen! Ein Extrapunkt zählt zur **erreichten**, nicht aber zur **erreichbaren** Punktzahl.

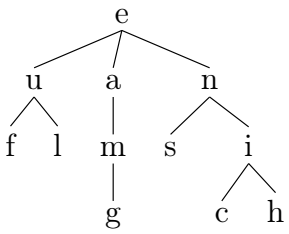
Aufgabe 3.1. *Baum-Traversierungen* (2+2+2 Punkte)

(Vorlesung am 12.05.2015, Abschnitt 4.3 im Skript)

Betrachten Sie den rechts dargestellten geordneten Baum B. Geben Sie an, in welcher Reihenfolge die Knoten in B in einem

a) Präorder-Durchlauf
b) Inorder-Durchlauf
c) Postorder-Durchlauf

besucht werden.



Aufgabe 3.2. *Spielbäume* (8 Punkte)

(Vorlesung am 12.05.2015, Abschnitt 4.3 im Skript)

Alice und Bob spielen gegeneinander. Rundenweise wechseln sich die beiden ab. Alice beginnt. Wir modellieren alle möglichen Spielverläufe durch einen Spielbaum T in Kind-Geschwister-Darstellung:

```
enum Spieler {ALICE, BOB};
enum Gewinner {A, B, NICHT_INITIALISIERT};

typedef struct Knoten {
   Spieler s;
   Gewinner g;
   Knoten *LKind, *RGeschwister; }
```

Die Wurzel von T entspricht dem Spielbeginn: Alice beginnt das Spiel, d.h. für den Zeiger `root` auf die Struktur der Wurzel gilt `root->s==ALICE`. Allgemein gilt: Wenn `p` ein Zeiger auf die Struktur des Knotens v ist und Alice in v am Zug ist, d.h. `p->s==ALICE` gilt, dann hat v für jeden möglichen Zug von Alice ein Kind v' und Bob ist im Kind v' am Zug. Natürlich gilt analoges, sollte Bob in v am Zug sein. Ist kein Zug für den in v ziehenden Spieler möglich, dann ist v ein Blatt und das Spiel ist zu Ende. Der Gewinner der Spiels ergibt sich aus der Markierung des Blattes, in dem das Spiel geendet hat.

In der Eingabe sind alle Blätter mit Gewinnern markiert: Für einen Zeiger `p` auf ein Blatt gilt `p->g==A` genau dann, wenn Alice in diesem Blatt gewonnen hat. Analog gilt `p->g==B`, wenn Bob gewonnen hat. Für alle inneren Knoten ist der Gewinner `g` anfänglich auf `NICHT_INITIALISIERT` gesetzt.

Bitte wenden!

Wir erhalten den Zeiger `root` und sollen entscheiden, ob Alice eine Gewinnstrategie hat, d.h. ob Alice den Sieg selbst dann erzwingen kann, wenn Bob den jeweils besten Gegenzug auswählt.

Entwerfen Sie einen möglichst schnellen Algorithmus, der entscheidet, ob Alice eine Gewinnstrategie hat, und analysieren Sie seine Laufzeit in Abhängigkeit von der Anzahl der Knoten in T.

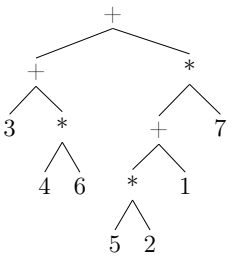
Aufgabe 3.3. *Syntaxbäume* (2+6 Punkte)

(Vorlesung am 12.05.2015, Abschnitt 4.3 im Skript)

Wir betrachten arithmetische Ausdrücke auf den zweistelligen Operationen + und $*$. Ein Beispiel für einen solchen Ausdruck ist $(3 + (4 * 6)) + (((5 * 2) + 1) * 7)$.

Wir lernen in dieser Aufgabe verschiedene Darstellungen arithmetischer Ausdrücke kennen: Syntaxbaum, Präfix-, Infix- und Postfix-Notation.

Ein Syntaxbaum wird durch die folgende rekursive Regel definiert (der Syntaxbaum für das Beispiel ist rechts dargestellt):



- Ein einziger Knoten, mit einer Zahl x markiert, ist ein Syntaxbaum mit Ergebnis x.
- Wir erhalten aus zwei Syntaxbäumen T_1 und T_2 mit Ergebnissen x_1 und x_2 einen neuen Syntaxbaum, indem wir eine neue Wurzel, markiert mit dem Operator + oder $*$, einfügen und T_1 und T_2 zum linken bzw. rechten Teilbaum machen. Das neue Ergebnis ist $x_1 + x_2$ bzw. $x_1 * x_2$.

Die Präfixnotation eines arithmetischen Ausdrucks mit Syntaxbaum T entspricht der Knotenreihenfolge eines Präorderdurchlaufs von T. (Dem obigen Beispiel entspricht dann die Präfixnotation + + 3 * 4 6 * + * 5 2 1 7.) Analog entspricht die Postfixnotation der Postorder-, und die Infixnotation der Inorder-Traversierung von T, bei letzterer dürfen zusätzlich Klammern eingesetzt werden.

Wir sind die Infixnotation gewohnt, allerdings ist diese nur mit Bindungsregeln eindeutig. Die Evaluierung eines „Infix-Ausdrucks“ dauert zu lange, stattdessen transformiert man häufig einen Infix-Ausdruck in einen „Präfix-Ausdruck“ und evaluiert den Präfix-Ausdruck.

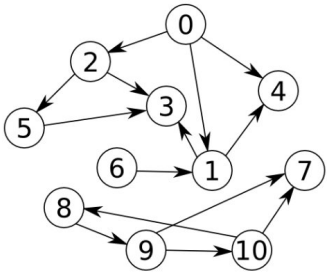
a) Evaluieren Sie den Ausdruck + * + * 2 3 1 3 + 7 * 3 3. Geben Sie auch den zugehörigen Syntaxbaum an.
b) Gegeben ist ein arithmetischer Ausdruck a in Präfix-Notation in einem Array A gespeichert. Sei $|A|$ die Länge des Ausdrucks. Entwerfen Sie einen möglichst schnellen Algorithmus, der den Ausdruck a evaluiert, und bestimmen Sie die Laufzeit in Abhängigkeit von $|A|$.

   Hinweis: Es ist nicht nötig, den Syntaxbaum zu erstellen, um ihn anschließend auszuwerten. Können Sie den Präfix-Ausdruck in einem Durchlauf von links nach rechts (oder rechts nach links) auswerten?

Aufgabe 3.4. *DFS, BFS und topologische Sortierung* (4+2+2 Punkte)

(Vorlesungen am 19.05.2015 und 26.05.2015, Abschnitt 4.4 im Skript)

Nehmen Sie an, dass die Adjazenzlisten des gegebenen Graphen aufsteigend sortiert sind (z.B. ist 1,2,4 die Adjazenzliste von Knoten 0).



a) Geben Sie den Tiefensuchwald und die Kantentypen aller Kanten des Graphen an. Wir starten die Tiefensuche in Knoten 0.
b) Vernachlässigen Sie die Richtung aller Kanten und geben Sie den Breitensuchwald an. Wieder soll in Knoten 0 gestartet werden.
c) Geben Sie eine topologische Sortierung der Knoten $0, \ldots, 6$ an.

Begründungen sind nicht notwendig.

Aufgabe 3.5. *Nachbarschaftsinitiative* (8 Punkte)

(Vorlesung am 26.05.2015, Abschnitte 4.4.3 oder 4.4.4 im Skript)

Karl Maeleon möchte einen Reptilien-Streichelzoo für die Kinder der Nachbarschaft eröffnen, in dem alle Haustiere der Nachbarkinder gestreichelt werden können. Leider hat Herr Maeleon nur zwei Gehege zur Verfügung, notwendigerweise müssen sich also mehrere Reptilien ein Gehege teilen. Natürlich sollen alle Kinder ihre Tiere wieder unversehrt mit nach Hause nehmen können – die Tiere in einem Gehege müssen sich also vertragen.

Deshalb erkundigt sich Herr Maeleon bei allen Kindern, welche zwei Tiere sich jeweils nicht vertragen. Beispielsweise wäre die Schlange von Anna Konda bedroht durch den Alligator von Kai Mahn – beide Tiere müssen also in unterschiedlichen Gehegen untergebracht werden.
Aus den Angaben der Kinder erstellt Herr Maeleon einen ungerichteten Konfliktgraphen $G{=}(V, E)$, wobei jedes Reptil durch einen Knoten in V und jeder Konflikt durch eine Kante in E repräsentiert wird. Herr Maeleon kann den Zoo nur dann eröffnen, wenn sich alle Tiere so auf zwei Gehege aufteilen lassen, dass innerhalb jedes Geheges kein Konflikt besteht.

Helfen Sie Herrn Maeleon, indem Sie einen Algorithmus entwerfen, der G darauf testet, ob eine derartige Aufteilung der Tiere möglich ist. Die Laufzeit soll linear in der Größe von G sein, also höchstens $\mathcal{O}(|V| + |E|)$ betragen. Nehmen Sie an, dass G in Adjazenzlisten-Darstellung vorliegt.