Übungen zur Vorlesung Einführung in die Informatik II

Ausgabe: 28. Mai 2002 Abgabe: 3. Juni 2002, 12.00 Uhr

Die Abgabe erfolgt ausschließlich über das System eClaus! Siehe
http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ss02/info_II_02.html

Aufgabe 1: (Suchen in Bäumen) **zum Votieren**

Gegeben sei ein 2-gleichverzweigter, geordneter und markierter Baum der Höhe h. Die Markierungen seien ganze Zahlen. $M(n)$ bezeichnet die Markierung des Knotens n. Es sei die folgende Ordnung "<" definiert: Für jeden Knoten l des linken Teilbaums eines jeden Knoten n im Baum gilt $M(l) < M(n)$. Für jeden Knoten r des rechten Teilbaums eines jeden Knoten n im Baum gilt $M(r) > M(n)$.

a. (0.5 Punkte) Wieviele Schritte sind maximal nötig, um in einen Baum mit n Knoten einen vorhandenen Knoten von der Wurzel aus zu finden? Wie ändert sich der Aufwand im schlechtesten Fall, wenn der Baum vollständig ist?

b. (1 Punkt) Implementieren Sie die Funktion `Tree_Search` in Ada95, die die Suche nach einem Element im Baum durchführt. Die Signatur sieht folgendermaßen aus:

```
function Tree_Search (Key : Integer ; A_Tree : Tree) return Tree;
```

Verwenden Sie bei der Implementierung die folgenden Datentypen:

```
type Tree;
type Tree_Ptr is access Tree;
type Tree is record
  Contents : Integer;
  Left     : Tree_Ptr;
  Right    : Tree_Ptr;
end record;
```

c. (0.5 Punkte) Geben Sie ein Traversierungsschema an, das die Knoten im Baum in absteigend sortierter Reihenfolge ausgibt.

Aufgabe 2: (Eigenschaften von Bäumen) **zum Votieren**

(1 Punkt) Was ist die minimale und die maximale Anzahl von Knoten, die ein k-beschränkter Baum der Höhe h haben kann?

In eClaus gibt es eine Multiple Choice Aufgabe, in der Sie angeben können, ob Sie wenigstens eine Aufgabe votieren wollen. Diese Aufgabe wird dann mit der in der Übung erreichten votierten Punkteanzahl bewertet.

Alle weiteren Informationen zur Vorlesung finden Sie unter

`http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ss02/info_II_02.html`

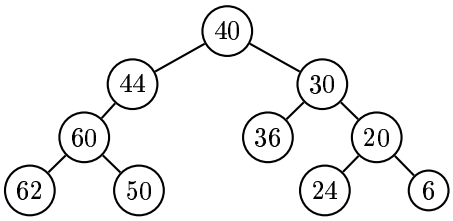
Fragen zur Vorlesung und den Übungen, sowie Anregungen und Kritik können auf dem Schwarzen Brett

`http://fachschaft.informatik.uni-stuttgart.de/forum/`

diskutiert werden.

Aufgabe 3: (Eigenschaften von Bäumen) **zum Votieren**

a. (1.5 Punkte) Geben Sie für den untenstehenden Baum an, ob er ausgeglichen, vollständig ausgeglichen oder nicht ausgeglichen ist, sowie die Ausgabe seiner Preorder-, Inorder- und Postorder-Traversierung. Würden Sie den Baum als binären Suchbaum klassifizieren?



b. (1.5 Punkte) Gegeben seien die Postorder- und die Inorder-Traversierung eines Binärbaumes.

Postorder D–B–G–E–F–C–A

Inorder D–B–A–E–G–C–F

Geben Sie den zugehörigen Binärbaum an.

c. (1 Punkt) Zeigen Sie anhand eines Beispiels, daß die Angabe der Preorder- und der Postorder-Traversierung nicht ausreicht, um einen Binärbaum eindeutig zu charakterisieren. (Hinweis: Verwenden Sie als Beispiel einen Baum mit möglichst wenig Knoten.)

d. Im Skript (Kaptiel 3.5) wurde eine iterative Implementierung der Inorder-Traversierung für einen Binärbaum vorgestellt, wobei ein Stack als Datenstruktur benutzt wurde.

- (1 Punkt) Wie sieht die iterative Postorder-Implementierung für Binärbäume aus?
- (1 Punkt) Wie sieht die iterative Postorder-Implementierung für normierte Binärbäume (siehe Skript, Kapitel 3.3) aus?

Aufgabe 4: (Baumimplementierungen) **schriftlich**

In der Vorlesung wurden verschiedene Varianten von Baumdarstellungen vorgestellt. Zwei Varianten sollen Sie im folgenden programmieren. Alle Vorgaben finden Sie auf der Seite zur Vorlesung

`http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ss02/info_II_02.html`

Holen Sie sich die Vorgaben von der Web-Seite und ergänzen Sie die Pakete.

a. (6 Punkte) Das Paket `tree` soll so implementiert sein, dass jeder Knoten beliebig viele Kinder hat. Sie finden in den Dateien zu tree weitere Hinweise, was zu tun ist. Schreiben Sie sich ein kurzes Testprogramm, mit dessen Hilfe Sie die Implementierung testen. Versuchen Sie sich sinnvolle Testfälle auszudenken! Geben Sie `tree.adb` und `tree_test.adb` in eClaus ab.

b. (6 Punkte) Holen Sie sich das Paket tree2 von der oben angegebenen Adresse; Ergänzen Sie das Packet `tree2`, so dass es beliebige Bäume mittels eines *normierten Binärbaums* (siehe Skript) implementiert. Sie finden in den Dateien zu `tree2` weitere Hinweise, was zu tun ist. Schreiben Sie sich ein kurzes Testprogramm `tree2_test`, mit dessen Hilfe Sie die Implementierung testen. Sie dürfen die gleichen Testfälle wie bei Teil a verwenden. Geben Sie `tree2.adb` und `tree2_test.adb` in eClaus ab.