Info2 Uebung vom 27.5.02

Gruppe F, Thomas Briner, http://n.ethz.ch/student/brinerth/info2

27. Mai 2002

1 Vorbesprechung Uebung 7 - Sortieren

1.1 Masse für Vorsortiertheit

0	1	2	3	4	5	6	7	8	9
2	1	3	5	4	8	2	3	9	5

las: Longest ascending subsequence, muss nicht am Stück sein, zB:

rem: remove, alle Elemente, die nicht zur las gehören, zB

runs: aufsteigende Teilfolgen am Stück. Anzahl runs im Beispiel:

inversionen: Anzahl Elemente die in Folge hinten an Element i stehen, aber kleiner sind als Element i, aufsummiert über alle Elemente, im Beispiel:

Je nach Art und Grad der Vorsortiertheit gibt es Sortieralgorithmen, die sich besonders gut dafür eignen und in diesem Spezialfall eine besonders gute Laufzeit haben.

1.2 Ueberblick über Sortierverfahren

1.2.1 Bubble Sort

Immer zwei nebeneinander liegende Elemente werden verglichen und falls notwendig vertauscht. So wird sooft durch das Array durchgegeangen, bis keine Vertauschungen mehr stattfinden.

Laufzeit:

1.2.2 Insertion Sort

1.2.3 Selection Sort

1.2.4 Merge Sort

1.2.5 Heap Sort

1.2.6 Quick Sort

1.3 Bucket Sort

Bucket (engl.: Eimer, Kübel)

Beispiel:
3-stellige Dezimalzahlen sortieren, unsortierte Liste:
176, 321, 107, 981, 277, 345, 316, 276

Es wird immer nur eine Stelle angeschaut. Es stehen 10 Buckets (die Ziffern 0 bis 9) zur Verfügung. Jede Zahl wird gemäss ihrer Ziffer an der entsprechenden Stelle in den richtigen Bucket eingefüllt. Anschliessend werden die Buckets in aufsteigender Reihenfolge wieder geleert und der Inhalt zurück in die Liste gefüllt.

zB nach der zweiten Stelle in die Buckets eingefüllt:

0	1	2	3	4	5	6	7	8	9
107	316	321		345			176, 277, 276	981	

Nachher wieder rausgeholt:
107, 316, 321, 345, 176, 277, 276, 981

Frage: In welcher Reihenfolge müssen die Stellen abgearbeitet werden, damit am Schluss eine korrekt sortierte Liste entsteht?

Am Beispiel durchspielen:

Erster Durchgang:

Zweiter Durchgang:

Dritter Durchgang:

Implementation:

Buckets: Array, pro Feld 2 POINTER (first, last). Daran angehängt werden alle Elemente, die in den entsprechenden Bucket gehören.

- In die Buckets hinein: hinten in die Liste
- Aus den Buckets heraus: vorne aus der Liste

Laufzeit:

- Für Verteilen auf die Buckets:
- Für Aufsammeln aus den Buckets:
- Wie oft muss dieses Einfügen/Einsammeln gemacht werden?

Umsetzung auf Strings:

- Stringvergleich: Was stimmt? $abc > abca$ oder $abca > abd$? $\Rightarrow$hinten mit 0x auffüllen
- Anzahl benötigte Buckets?

2 Nachbesprechung Uebung 5

2.1 Schöner Programmieren

```
PROCEDURE lookup*(n:Node;key:INTEGER):Node;
        BEGIN
                IF (n=NIL) THEN RETURN NIL;
                END;
                IF (n^.key=key) THEN
                        RETURN n;
                ELSIF (n^.key>key) & (n^.left#NIL) THEN
                        RETURN (lookup(n^.left,key));
                ELSIF (n^.key<key) & (n^.right#NIL) THEN
                        RETURN (lookup(n^.right,key));
                END;
        END lookup
```

Schreib diese Prozedur lookup so um, dass möglichst wenige Tests notwendig sind und der Kontrollfluss so einfach wie möglich verläuft.

2.2 Mehrfach gleiche Elemente in einem Baum

Je nach Anwendung sind verschiedene Strategien sinnvoll.

Jeder Schlüssel kommt nur einmal vor

Wenn ein zweites Mal der gleiche Schlüssel eingefügt werden soll, so wird der Inhalt des Elements mit dem gleichen Schlüssel mit dem neuen Inhalt überschrieben.

Bei AVL Bäumen kommt jedes Element nur 1 Mal vor (Ab drei Vorkommen der gleichen Zahl kommt es ansonsten zu seltsamen Effekten: Füge in einen leeren AVL Baum dreimal den Schlüssel 1 ein).

Applet zu AVL Bäumen: http://www.ibr.cs.tu-bs.de/lehre/ss98/audii/applets/avlbaum/

Jeder Schlüssel kann beliebig oft vorkommen

Wenn ein zweites Element mit dem gleichen Schlüssel eingehängt werden soll, so kommt es einfach an die korrekte Stelle und wird dort eingefügt.

Dieses Vorgehen macht zB bei einem elektronischen Telefonbuch mit Nachnamen als Schlüssel Sinn.

Problem: Ein Lookup ist nun nicht mehr so einfach wie zB oben ausprogrammiert. Es muss nach dem Finden des Elements mit dem korrekten Schlüssel auch noch den Inhalt überprüfen oder einfach ohnehin alle Elemente finden.

Bsp:

Folgende Zahlen werden in einen normalen binären Suchbaum eingefügt, in dem gleiche Schlüssel mehrmals vorkommen können: 8, 3, 1, 14, 6, 5, 7, 8, 6. Zeichne den entstehenden Baum!

3 Aufgabe aus altem Vordiplom: Gleiche Summe

Die Ausgangslage für ein Spiel ist eine Folge von höchstens 50 ganzen Zahlen. Der Spieler macht einen Zug, indem er entweder die erste oder die letzte Zahl der Folge entfernt und - streng abwechselnd - in einen von zwei Beuteln steckt; er gewinnt, wenn am Ende die Summen in beiden Beuteln gleich sind.

Beispiel:

1 4 6 2 3 5 7 Stecke 1 in Beutel A => 1
4 6 2 3 5 7 Stecke 7 in Beutel B => 7
4 6 2 3 5 Stecke 4 in Beutel A => 5
6 2 3 5 Stecke 5 in Beutel B => 12
6 2 3 Stecke 6 in Beutel A => 11
2 3 Stecke 2 in Beutel B => 14
3 Stecke 3 in Beutel A => 14

Summe in Beutel A: 14
Summe in Beutel B: 14

Schreibe ein Oberon-Programmstück, welches für eine beliebige Ausgangslage entscheidet, ob der Spieler so spielen kann, dass er gewinnt. Was ist die Laufzeit?