Prof. Dr. Gerd Stumme, Folke Eisterlehner, Dominik Benz
Fachgebiet Wissensverarbeitung

9.06.2009

8. Präsenzübung „Algorithmen und Datenstrukturen“

Sommersemester 2009

1 Listen

StackList: Geben Sie eine Implementierung des `Stack`-Interfaces mit Hilfe der Klasse `List` aus der Vorlesung an. Auf die Behandlung von Ausnahmen kann dabei verzichtet werden.

Feld vs. Liste: Diskutieren Sie zu jeder in folgender Tabelle angegebenen Operationen die Laufzeit wenn diese *a*) auf einem Feld und *b*) auf einer Liste durchgeführt werden. Überlegen Sie sich vorher, welche Operationen für einen realistischen Vergleich Ergebnisse zu zählen sind. Tragen Sie jeweils eine obere Schranke für die Worst-Case-Laufzeit in O-Notation ein (Gehen Sie bei der Listen-Implementierung von einer doppelt verketteten Liste wie in der Vorlesung beschrieben aus).

Operation	**als Feld**	**als Liste**
Einfügen eines Elements in eine sortierte Folge		
Einfügen eines Elements in eine unsortierte Folge		
Löschen eines Elements in einer sortierten Folge		
Löschen eines Elements in einer unsortierten Folge		
Suchen eines Elements in einer sortierten Folge		
Suchen eines Elements in einer unsortierten Folge		
Konkatenieren zweier sortierten Folgen		
Konkatenieren zweier unsortierten Folgen		

Für die Löschoperationen kann davon ausgegangen werden, dass jeweils eine Referenz auf das betreffende Objekt bereits vorliegt und dieses nicht erst gesucht werden muss.

Suche in sortierten Listen: Überlegen Sie sich, wie man die Suche nach einem Schlüssel in einer sortierten Liste beschleunigen kann.

Türme von Hanoi: Die „Türme von Hanoi“ sind ein klassisches mathematisches Knobel- und Geduldsspiel, welche vermutlich Ende des 19. Jahrhunderts vom Mathematiker Édouard Lucas erfunden wurden.

Das Spiel besteht aus drei Stäben und n Scheiben ($n \geq 3$) mit paarweise verschiedenem Durchmesser. Zu Beginn des Spiels liegen alle Scheiben der Größe nach aufsteigend sortiert übereinandergestapelt auf dem ersten Stab. Am Ende des Spiels sollen alle Scheiben in gleicher Reihenfolge auf dem letzten Stab liegen.

Hierzu darf in jedem Schritt eine Scheibe versetzt werden, mit der Randbedingung, dass zu keinem Zeitpunkt eine größere Scheibe auf einer kleineren Scheibe liegt.

Folgender rekursiver Algorithmus löst das Problem für eine beliebige Zahl $n > 1$ von Scheiben:

```
algorithm moveTower(n, src, tmp, dst)
Eingabe: Anzahl n von Scheiben, Startstab src, Zielstab dst
         und Zwischenstab tmp jeweils als Stack

   if( n > 0 )
      moveTower(n − 1, src, dst, tmp);
      dst.push(src.top())
      src.pop()
      moveTower(n − 1, tmp, src, dst);
   fi
```

a) Erklären Sie intuitiv die Arbeitsweise des Algorithmus in eigenen Worten.

b) Stellen Sie eine Rekursionsgleichung für eine obere Schranke der Laufzeit $T(n)$ des Algorithmus auf.

c) Leiten Sie daraus eine explizite Formel in O-Notation her.