Universität Stuttgart
Fakultät 5, Institut FMI
Einf.i.d.Informatik II

Sommersemester 2003
Lewandowski, Claus
05. Juni 2003

**Aufgabenblatt 6** **Abgabe: 13. Juni, 20.00 Uhr**

---

1. **Zur Wiederholung: einfach verkettete Listen** (mittel) (schriftl., 3+2 Pkt.)
Der Inhalt einfach verketteter Listen lässt sich leicht ausgeben, in dem man beginnend mit dem Anker der Liste, diese sequentiell durchläuft, bis man auf einen `null` Zeiger trifft. Falls das letzte Element nicht `null` als Nachfolger hat, sondern ein beliebiges Element der Liste, so entsteht ein Zyklus. Die Ausgabe des Inhalts mit obigem Algorithmus führt dann in eine Endlosschleife.

   a) Schreiben Sie eine Ada 95 Funktion, die zu einer gegebenen einfach verketteten Liste erkennt, ob diese einen Zyklus hat oder nicht (schreiben Sie sich hierzu vorher eine Prozedur, die Listen mit Zufallszahlen erzeugt, die Länge der Listen soll ebenso zufällig gewählt sein, das letzte Element der Liste soll entweder auf `null` zeigen oder auf ein zufällig gewähltes Element der Liste).

   b) Schreiben Sie eine Ada 95 Funktion, die zu einer gegebenen einfach verketteten Liste, die einen Zyklus enthält, das letzte Element des Zyklus erkennt und den Nachfolger dieses letzten Elements auf `null` umsetzt (und somit den Zyklus auflöst).

   Schätzen Sie jeweils den Aufwand Ihrer Algorithmen ab. Kommentieren Sie ihre Abgaben bitte so, dass Ihr Tutor nachvollziehen kann, welche Ideen Sie beim Entwurf der Algorithmen hatten.

   Die Aufwandsabschätzung sollte ebenfalls als Kommentarzeilen im Programmcode erfolgen (schreiben Sie Sonderzeichen wie Summenzeichen einfach aus, also z.B. $\sum_{i=1}^{n}$ als "Summe(i=1 bis n)").

   **Hinweis:** Das Programm soll für die Liste keinerlei Zusatzinformation wie z.B. die Anzahl der Elemente verwalten, ebenso dürfen in den Listenelementen keinerlei zusätzliche Informationen gespeichert werden. Machen Sie sich klar, dass Sie direkt auf den Zeigern arbeiten müssen, da die Werte der Listenelemente nicht paarweise verschieden sein müssen.

   **Hinweis 2:** Es gibt Lösungen, die mit maximal drei Zeigern auskommen und nur linearen(!) Aufwand haben.

2. **Packungsalgorithmen** (mittel) (schriftlich, 5+2+2+1 Punkte)
Sie haben die Aufgabe, eine gegebene Menge von $n$ Objekten, deren Größe jeweils zwischen 0 und 100 liegt, in Behälter der Kapazität 100 zu füllen.

   a) Schreiben Sie ein Ada 95 Programm, das mit folgenden drei Methoden versucht, dabei mit möglichst wenig Behältern auszukommen:

      - Berechnen Sie mit *Backtracking* die minimal benötigte Behälteranzahl. Achten Sie darauf, dass Sie Berechnungen, die zu keiner Verbesserung mehr führen können, frühzeitig abbrechen.

- Die *"First-Fit"-Heuristik* legt ein Objekt in den erstmöglichen Behälter (d.h., sei $B(j)$ der Füllstand des Behälters $j$, so wähle zu einem Objekt der Größe $i$ den Behälter $k$ mit $k := \min\{j|B(j) + i \leq 100\}$). Ist z.B. der erste Behälter zu 50% und der zweite zu 60% belegt, so würde ein Objekt der Grösse 35 in den ersten Behälter gelegt. Bei einem Objekt der Größe 55 würde ein neuer Behälter begonnen. Schreiben Sie einen Algorithmus, der die Anzahl der benötigten Behälter berechnet, wenn die "First-Fit"-Heuristik verwendet wird.
- Bei der *"Best-Fit"-Heuristik* wählt man den Behälter so, dass der verbleibende Platz im gewählten Behälter möglichst gering wird (wähle zu einem Objekt der Größe $i$ den Behälter $j$ mit $B(j) + i$ maximal unter der Bedingung $B(j) + i \leq 100$). Im obigen Beispiel würde das Objekt der Grösse 35 in den zweiten Behälter gelegt. Schreiben Sie einen Algorithmus, der die Anzahl der benötigten Behälter berechnet, wenn die "Best-Fit"-Heuristik verwendet wird.

Führen Sie ihre Algorithmen für $n = 12$ und für 1000 zufällig erzeugte Beispiele aus und geben Sie aus, wieviel Behälter jedes Verfahren im Mittel benötigt hat.

b) Untersuchen Sie, welchen Einfluss eine aufsteigende bzw. absteigende Sortierung der Objekte nach ihrer Größe auf das Ergebnis der benötigten Behälteranzahl hat (zur Sortierung können Sie einen der verifizierten Algorithmen von Aufgabenblatt 1 oder 3 verwenden). Hat es einen Einfluss auf die Laufzeit beim Backtracking?

c) Konstruieren Sie ein Beispiel, bei dem die "First-Fit"-Heuristik im Vergleich zum Optimum sehr viele Behälter benötigt. Wie viele Behälter benötigt die "Best-Fit"-Heuristik für dieses Beispiel?

d) Finden Sie ein Beispiel, bei dem die "Best-Fit"-Heuristik mehr Behälter benötigt als die "First-Fit"-Heuristik.

**3. Kopiertechniken (vgl. 3.1.3.11)** (mittel) (votieren, 1.5+1.5+2+2 Punkte)
In der Vorlesung haben Sie bereits einige Standardtechniken zur Speicherbereinigung kennengelernt. In dieser Aufgabe sollen Sie die Kopiertechnik, die im Skript unter 3.1.3.11 skizziert ist, konkretisieren. Überlegen Sie sich Vorgehensweisen zu folgenden vier Fällen:

a) ohne Beseitigung der Fragmentierungen falls keine zyklischen Strukturen vorliegen

b) mit Kompaktifizierung falls keine zyklischen Strukturen vorliegen

c) ohne Beseitigung der Fragmentierungen falls allgemeine Geflechte erlaubt sind

d) mit Kompaktifizierung falls allgemeine Geflechte erlaubt sind

Es werden maximal 20 (der erreichbaren 22) Punkte des Aufgabenblattes gewertet. Alles weitere unter

http://www.informatik.uni-stuttgart.de/fmi/fk/lehre/ss03/info_II/default.htm