Dr. Andreas Borchert
Michael Wiedemann
Abteilung Angewandte Informationsverarbeitung

1. Juli 2004
Blatt 8

Allgemeine Informatik II (SS 2004)

Abgabetermin: 8. Juli 2004

9 Zuteilung – was, schon wieder? – 10 Punkte

Wir haben inzwischen verschiedene Methoden kennengelernt, mit denen man aus einer Liste eine vorgegebene Anzahl von Elementen herauszieht. Dieses Mal bedienen wir uns hierfür des Algorithmus S aus dem Buch *The Art of Computer Programming – Volume 2 – Abschnitt 3.4.2* von Donald E. Knuth. Dieser Algorithmus arbeitet mit Wahrscheinlichkeiten. Der grosse Vorteil des Algoritmus: die Komplexität beträgt O(N), mit N = Zahl der Elemente in der Liste. Zum Vergleich: Beim Blatt 6 hatten wir (bei Verwendung eines Zufallszahlengenerators) eine Komplexitaet von $O(N^2)$.

Zunächst zum Ablauf des Algorithmus.

Der Algorithmus bestimmt zufällig n Elemente aus einer Liste mit N Elementen, wobei $0 < n \leq N$. Dabei benutzt der Algorithmus folgenden Ansatz: Sind bereits t Elemente abgearbeitet worden, so ist die Wahrscheinlichkeit der Auswahl für das $t+1$-te Element $W_{t+1} = (n-m)/(N-t)$, wobei m die Anzahl der bisher ausgewählten Elemente ist.

Der Algorithmus an sich besteht aus 5 Schritten:

1. Initialisieren: Wir benötigen zwei Variablen: t gibt an, wieviele Elemente der Liste bereits abgearbeitet wurden, m repräsentiert die Anzahl der bisher ausgewählten Elemente. Wir setzen $t = 0$, $m = 0$.
2. Zufallszahl generieren: Wir suchen uns eine Zufallszahl U mit Wert $0 \leq U \leq 1$. Die Zufallszahl sollte gleichverteilt sein (siehe Tipps).
3. Test: Falls folgende Ungleichung gilt, fahren wir mit Schritt 5 fort, sonst mit Schritt 4: $(N-t)U \geq n-m$ (siehe oben).
4. Auswahl: Wir wählen das nächste Element für unsere Auswahl aus, erhöhen dabei m und t um eins. Falls nun $m < n$, gehen wir zurück zu Schritt 2, da wir noch nicht fertig sind, ansonsten bricht der Algorithmus (mit $m = n$) ab.
5. Fortfahren: Wir wählen das nächste Element *nicht* aus, erhöhen nur t um eins und fahren mit Schritt zwei fort.

Nun gilt für Euch, ein Oberon-Programm zu schreiben, das mit diesem Algorithmus arbeitet. Anders als bei den vorherigen Blättern kommt noch etwas hinzu: Ihr sollt dieses Mal zwingend modularisieren. Genau genommen benötigen wir insgesamt drei Module:

- Ein Modul LinearLists, welches die Funktionalitäten eine linearen Liste zur Verfügung stellen soll, passend zu einer gegebenen Schnittstelle.
- Ein Modul Sampler, welches den oben angegebenen Algorithmus für lineare Listen umsetzt, ebenfalls passend zur gegebenen Schnittstelle.
- Darüber hinaus braucht Ihr natürlich noch kleines Hauptmodul, welches Tests mit Euren Modulen durchführen kann.

Für diese zwei Module finden sich die DEFINITIONs in den Beispielen zu diesem Blatt; Ihr wisst also bereits, welche Prozeduren zu schreiben sind, nur den Inhalt müsst Ihr noch liefern.

Selbstverständlich ist wieder eine Argumentverarbeitung einzubauen. Es sollen Ein- und Ausgabedateien verarbeitet werden können, die gewünschte Anzahl an ausgewählten Elementen n soll über die Kommandozeile eingelesen werden. Außerdem ist zu überprüfen, ob $0 < n \leq N$. Daraus ergibt sich folgender Programmaufruf:
Selector [-i infile] [-o outfile] count

Generelle Tipps:

- Schaut Euch meine Beispielmodularisierung vom letzten Übungsblatt an. Ihr werdet sehen, dass einige Prozeduren des Moduls gar nicht schwierig sind.
- Geht schrittweise vor, das Modul LinearLists ist realtiv einfach. Ihr könnt mit ***oc -c -u Modul.od Modul.om*** schnell überprüfen, ob Euer Modul syntaktisch korrekt ist.
- Das Modul Sampler wiederum besteht aus einer einzigen Prozedur, die nur die Funktionalität des Algorithmus umsetzt.
- Eine Zufallszahl, die gleichverteilt ist, liefert zum Beispiel : *RandomGenerators.RealValS*.

Viel Erfolg!