Universität Ulm - Abteilung Angewandte Informationsverarbeitung

Prof. Dr. Franz Schweiggert SS 2002
Hans Braxmeier 13. Juni 2002

6. Übungsblatt zur Vorlesung Allgemeine Informatik II

Abgabetermin: 20. Juni 2002

Aufgabe 1: Hashtabellen *(10 Punkte)*

Wer kennt nicht die D-Info CD auf der alle Telefonnummern von ganz Deutschland abgespeichert sind! Es wäre (zu) einfach die 10 Millionen Namen und Nummern der Reihe nach in einem Array abzulegen, denn bei der Suche nach einem Namen kommt das böse Erwachen: Die Suche beginnt beim 1. Eintrag und endet im schlechtesten Fall beim 10. Mio. Eintrag (im Mittel sind es „nur“ 5 Mio. Vergleiche)! Macht trotzdem keinen Spaß...

Schlaue Leute hatten nun die Idee, einen Telefonbucheintrag als Zahl eindeutig zu codieren (Wie die das machen sei den Göttern überlassen) und diese Zahl dann mit Hilfe einer Funktion in eine Adresse umzuwandeln bei der schließlich der entsprechende Eintrag gespeichert wird. Wird anschließend nach einem Namen gesucht, kann (fast) direkt an die entsprechende Adresse gesprungen werden und der Eintrag steht „sofort“ zur Verfügung!

Das Codieren überlassen wir den Göttern und nehmen an, die Einträge seien bereits codiert. Die Funktion, die die Adressen berechnet, lautet:

$Adresse = Zahl\ MOD\ Arraygr\ddot{o}\beta e$, wobei die Arraygröße eine Primzahl sein sollte!

Unser Beispielarray habe die Größe 7 (Primzahl)! Dann ergeben sich z.B. folgende Adressen:

- Tefefonbucheintrag 1 sei codiert als Zahl 23657 mod 7 ergibt Adresse 4
- Tefefonbucheintrag 2 sei codiert als Zahl 118712 mod 7 ergibt Adresse 6
- Tefefonbucheintrag 3 sei codiert als Zahl 14 mod 7 ergibt Adresse 0
- Tefefonbucheintrag 4 sei codiert als Zahl 412 mod 7 ergibt Adresse 6
- usw.

Falls 2 Einträge die gleiche Adresse erhalten (wie bei der Zahl 118712 und 412), so spricht man von einer Kollision. Kollisionen können sehr häufig sein und müssen entsprechend behandelt werden.

Jeder Eintrag des Arrays sollte deshalb aus einer zunächst leeren Pointerliste bestehen, die dann mit den entsprechenden Zahlencodes gefüllt wird. Kommt es zu einer Kollision, wird die Liste um einen Eintrag erweitert, wie in Abbildung 1 gezeigt.

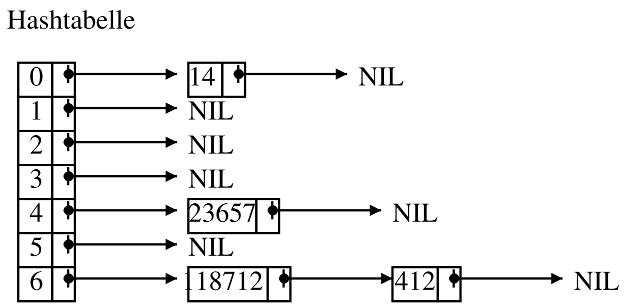

Abbildung 1: Belegung des Array

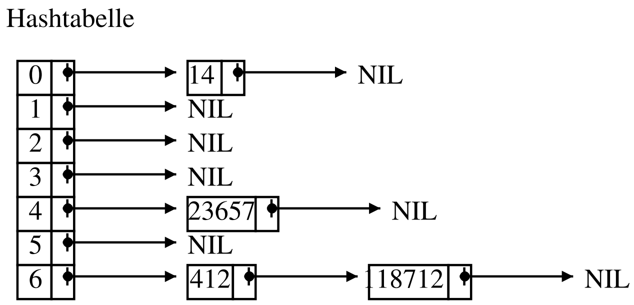

Abbildung 2: Aufsteigend sortierte Pointerlisten

Eure Aufgabe ist es nun ein Array mit den entsprechenden Pointerlisten zu definieren. Die codierten Zahlen sollen per Zufallsgenrator erzeugt und in Adressen umgewandelt werden. Abschließend müßt Ihr die Zahlen in den richtigen Pointerlisten speichern. Damit die Suche noch schneller wird, sollten die Zahlen in den Pointerlisten aufsteigend sortiert sein, wie in Abbildung 2 gezeigt.

Zusatzaufgabe: Elemente löschen *(10 Punkte)*

Erweitert Euer Programm so, daß Elemente auch gelöscht werden können!

Viel Erfolg!