Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Y. Maus, O. Saukh 27 Mai, 2016

Algorithmen und Datenstrukturen
Sommersemester 2016
Übungsblatt 4

Abgabe bis 12:00, Freitag, 03 Juni, 2016

Achtung: Bitte fügen Sie Ihrer Lösung eine (kurze) Datei *erfahrungen.txt* mit Ihren Erfahrungen mit dem jeweiligen Aufgabenblatt hinzu. Das Forum können (und sollen) Sie nicht nur für Fragen zur Übung, sondern auch für Fragen zur Vorlesung benutzen.

Aufgabe 1: Universelles Hashing (4 Punkte)

Sei $K := [0, \ldots, p-1]$ eine Menge möglicher Schlüssel und betrachten Sie die folgende Menge von Hashfunktionen

$$H = \{h_{a,b} : a, b \in \{0, \ldots, p-1\}\} \text{ mit} \tag{1}$$
$$h_{a,b}(x) = (ax + b) \mod m, \tag{2}$$

wobei $p \gg m$ gelte. Zeigen Sie, dass es eine Teilmenge $S \subseteq K$ der Größe $\lceil \frac{p}{m} \rceil$ gibt, so dass für jede Kombination von zwei Schlüsseln x and y in S, für jede Funktion $h \in H$, $h(x) = h(y)$.

Aufgabe 2: Range Queries in Binärbäumen (5 Punkte)

In der Vorlesung haben wir eine rekursiven Algorithmus gesehen, um alle Schlüssel eines binären Suchbaums in einem Bereich $[a, b]$ auszugeben. Zeigen Sie, dass so ein Range-Query mit Bereich $[a, b]$ angewandt (sowie in der Vorlesung gezeigt) Laufzeit $\mathcal{O}(T + R + 1)$ benötigt, wobei T die Tiefe des Baumes und R die Anzahl der Elemente im Bereich $[a, b]$ ist.
Hinweis: Zeigen Sie zunächst, dass die Aussage insbesondere für die Bereiche $[a, \infty]$ *und* $[-\infty, b]$ *gilt.*

Aufgabe 3: Anwendung von Hashing (6 Punkte)

Manchmal ist es von Interesse, zu wissen, welche Wörter in einem Text am häufigsten vorkommen. Im Buch "Harry Potter and the Chamber of Secrets" zum Beispiel kommt das Wort *Harry* etwa 1500-mal vor.
Schreiben Sie ein Programm, welches einen vorgegebenen Text einliest und die k häufigsten Wörter im Text wiedergibt, k ist hierbei ein Eingabeparameter.[1] Die Laufzeit soll in Abhängigkeit von k der Anzahl Wörter des zu analysierenden Textes sein. Bei einem Text mit n Wörtern soll die Laufzeit Ihres Programmes $\mathcal{O}(kn)$ sein. Ihr Programm soll davon ausgehen, dass der Eingabetext vorverarbeitet wurde: in jeder Zeile der Eingabe steht genau ein Wort; das sollte die Aufgabe etwas erleichtern.
Im *public* SVN Verzeichnis liegen zwei Dateien, *test.txt* und *input.txt*, beide im oben beschriebenen Format. Erstere können Sie zum Testen verwenden, die zweite Datei müssen Sie unter Verwendung von $k = 15$ analysieren und Ihr Ergebnis sollen Sie in einer Datei *output.txt* abspeichern. Zu dieser Aufgabe stellen wir auch einige vorgefertigte Strukturdateien zur Verfügung, die Ihnen eventuell helfen können.
Trivia: Mit Datenstrukturen, die Sie erst noch kennenlernen müssen, lässt sich die Laufzeit auf $\mathcal{O}(n \log k)$ reduzieren. Es gibt allerdings auch eine etwas komplexere Methode, welche *Binary Search* und *Hashing* kombiniert, um die selbe Laufzeit zu erhalten. Wenn Sie sich herausgefordert fühlen, versuchen Sie doch, diese Lösung zu finden. Oder vielleicht sehen Sie sogar, wie man das Problem in Zeit $\mathcal{O}(n)$ lösen kann? ;)
Hinweis: Sie können annehmen, dass alle Operationen einer Hashtabelle konstanten Aufwand haben.

Aufgabe 4 (5 Punkte)

Für einen Online Shop, welcher n Produkte führt, sollen Sie eine effiziente Datenstruktur zur Verwaltung des Lagerzustands entwerfen. Alle Aufträge sollen in exakt der Reihenfolge abgearbeitet werden, in welcher sie eintreffen. Ein Auftrag hierbei ist ein Tupel *(Produkt, Anzahl, Auftragsdetails)*, d.h., fuer jedes Produkt in einer Bestellung wird ein eigener Auftrag erstellt. Zudem soll Ihre Datenstruktur folgendes unterstützen:

- *Insert(Auftrag)*: Man kann einen neuen Auftrag in $O(1)$ in die Datenstruktur einfügen.
- *ExtractNext()*: Ein Auftrag wird in $O(1)$ aus der Datenstruktur entnommen.
- *Query(Product)*: Gibt in $O(1)$ die Gesamtbestellmenge des betreffenden Produkts in allen Aufträgen zurück. D.h., gibt es zum Produkt P die t Aufträge

  $$\{A_1, \ldots, A_t\} = \{(P, k_1, D_1), \ldots, (P, k_t, D_t)\},$$

  dann soll *Query(P)* den Wert $k^P_{total} = \sum_{i=1}^{t} k_i$ zurückgeben.

Des weiteren soll die Datenstruktur speichereffizient sein, im folgenden Sinne. Man garantiert Ihnen, dass nie mehr als $m \ll n$ Aufträge gleichzeitig in der Warteschlange sind, und daher will man auch, dass obige Datenstruktur nur $O(m)$ Speicherplatz benötigt.
Beschreiben Sie, wie Sie diese Datenstruktur implementieren. Verwenden Sie *zusätzlich* Pseudocode, um Ihre Lösung zu verdeutlichen.

[1] Einen Gleichstand können Sie beliebig auflösen, d.h., wenn z.B. $k = 5$ und die Häufigkeiten der Wörter sind (absteigend sortiert) $24, 18, 18, 9, 9, 9, 9, 8, 8, \ldots$, dann sollen die Wörter mit den Häufigkeiten $24, 18, 18$ sowie zwei beliebige Wörter mit Häufigkeit 9 zurückgegeben werden.