Universität Konstanz
Fachbereich Informatik & Informationswissenschaft
Prof. Dr. Ulrik Brandes / Dr. Andreas Karrenbauer / Dr. Jürgen Lerner / Martin Mader

Algorithmen und Datenstrukturen
WS 2010/2011

10. Übungsblatt

Ausgabe: 22. Dezember 2010 **Abgabe:** 12. Januar 2011, 10 Uhr
Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Aufgabe 1: Hashing – quadratische Sondierung — 5 Punkte

Gegeben sei die folgende Hashfunktion

$$h\colon \mathbb{N} \to \{0,\ldots,9\};\ h(n) := \texttt{Quersumme}(n) \bmod 10\ ,$$

wobei die `Quersumme` einer Zahl n gegeben ist durch die Summe über alle Ziffern in der Dezimaldarstellung von n. (Zum Beispiel ist $\texttt{Quersumme}(431) = 4+3+1 = 8$.)

Fügen Sie die folgenden Zahlen, in der gegebenen Reihenfolge, unter Verwendung von h in eine Hashtabelle der Länge 10 ein. Bei Kollisionen wird mittels **quadratischer Sondierung** eine neue Arraystelle gesucht (open hashing; keine Verkettung).

$$937,\quad 543,\quad 8365,\quad 45,\quad 8768,\quad 26$$

Aufgabe 2: Hashing – `remove` — 3 Punkte

Eine Hashtabelle, die open hashing verwendet, soll neben `insert` und `find` Operationen auch `remove` (Löschen von Elementen) unterstützen. Beschreiben Sie wie Löschoperationen möglichst effizient realisiert werden können indem Elemente nur als gelöscht markiert werden; beschreiben Sie wie die Implementierungen der `insert` und `find` Operationen angepasst werden müssen.

Aufgabe 3: Hashfunktion für Strings — 2 Punkte

Eine Hashfunktion für beliebig lange Zeichenketten (Strings) sei dadurch definiert, dass eine gegebene Hashfunktion für Zeichenketten der Länge höchstens k auf die Teilwörter bestehend aus den ersten k Zeichen angewandt wird.

Beschreiben Sie Vorteile und/oder Nachteile bezüglich Berechnungsaufwand und Kollisionsverhalten, wenn mit dieser Hashfunktion ein Wörterbuch für eine natürliche Sprache realisiert werden soll.

Aufgabe 4: Textanalyse  10 Punkte

Implementieren Sie das Interface `material.u10.IWordCounter` durch eine Klasse `u10.name.WordCounter`, mit der man für große, im Dateisystem gespeicherte Texte die Häufigkeiten der vorkommenden Wörter bestimmen kann.

(a) Stellen Sie einen Konstruktor `WordCounter(String fileName)` zur Verfügung. Die Methode `read()` soll die im Konstruktor angegebende Datei analysieren. Für bessere Skalierbarkeit soll der Text nicht vollständig in den Hauptspeicher gelesen und erst dann analysiert werden, sondern Zeile für Zeile. Ignorieren Sie beim Lesen Sonderzeichen, ersetzen Sie Großbuchstaben durch Kleinbuchstaben, und entfernen Sie Leerzeichen vor oder nach Wörtern; Sie können dazu Methoden aus `java.lang.String` anwenden.

Mittels der Methode `getFrequency(word)` soll nach erfolgreichem Lesen die Anzahl der Vorkommen des gegebenen Wortes angefragt werden können. Die Methode `getNumberOfWords()` soll die Anzahl unterschiedlicher Wörter im Dokument zurückgeben. (6 Punkte)

(b) Die Methode `getKMostFrequent(k, words, frequencies)` soll die k häufigsten Wörter und die zugehörigen Häufigkeiten in die übergebenen Listen geschrieben werden (nicht-aufsteigend sortiert nach Häufigkeit; bei gleicher Häufigkeit lexikographisch sortiert). (4 Punkte)

Sie dürfen alle Klassen aus `java.io.*` und die in dieser Vorlesung implementierten Klassen verwenden. Dokumentieren Sie im Klassenkommentar ihre Vorgehensweise. Beschreiben Sie gegebenenfalls vorgenommene Änderungen an den bisher implementierten Datenstrukturen. Auf der WWW-Seite zur Übung finden Sie Beispieltexte.