Allgemeines Programmierpraktikum **SS 2008**
Dr. Henrik Brosenne Georg-August Universität Göttingen
Institut für Informatik

Übung 3

Testat nach Mittwoch, 14.05., 14.00 Uhr in der Gruppenübung.

Vorbereitung

Java

Informieren Sie sich in der *Java 2 Platform Standard Edition 5.0 API Specification* `http://java.sun.com/j2se/1.5.0/docs/api/` über das Einlesen von Dateien mit den Klassen `java.io.FileReader`, `java.io.BufferedReader` und `java.io.IOException`. Informieren Sie sich auch über die Klassen `java.lang.String` und `java.lang.Character`.

Aufgabe 1 – 100 Punkte

Huffman-Code

Schreiben Sie ein Programm, das zu einem vorgegebenen Text (in einer Textdatei) einen **optimalen Huffman-Code** berechnet. D.h. einen Huffman-Code, bei dem die Anzahl der Bits minimal ist, die für den Text zu übertragen sind.

Benötigte Klassen und deren Mindestanforderungen.

Codebaum

- Jeder Knoten hat genau einen 0-und einen 1-Nachfolger (Verzweigungsknoten) oder gar keinen Nachfolger (Blatt).
- Die Blätter tragen die Informationen, d.h. ein Zeichen aus dem Textalphabet.
- Der Weg von dem obersten Knoten (Wurzel) zu einem Blatt ist eindeutig. Die Markierungen (0 oder 1) der jeweils durchlaufenen Nachfolger werden aufgelistet. Diese Liste bildet das Codewort des, von dem ereichten Blatt getragenen, Textzeichens.
- Zwei Codebäume können miteinander verschmolzen werden. Der Wurzelknoten des resultierenden Baumes hat jeweils den Wurzelkoten eines der vorgegebenen Bäume als 0- bzw. 1-Nachfolger.
- Die `toString` Methode ist so überschrieben, das für jedes Blatt = Textzeichen eine Zeile zurückgeliefert wird.

Die Zeile besteht aus dem Textzeichen

- im Klartext, wenn es ein *druckbares Zeichen* ist
- als Zeichencode in Dezimaldarstellung gefolgt vom einem Doppelpunkt (`:`), wenn es ein *nicht druckbares Zeichen* ist

und dem zugehörigen Codewort.

Das Codewort ist eine Folge von 0 und 1, die sich dadurch ergibt über welche Folge von Nachfolgern das Blatt von der Wurzel aus erreicht wird.

Beispiel

Trägt ein Blatt das Zeichen `I` und wird über den 1-Nachfolger der Wurzel, dessen 0-Nachfolger und dessen 1-Nachfolger erreicht, ist das zugehörige Codewort `101`.

Hinweise

- Beim Erzeugen eines neuen Codebaumes kann ein Zeichen mit übergeben werden, dann besteht ein so erzeugter Baum aus einem Wurzelknoten = Blatt mit zwei `null`-Nachfolgern, der als Daten das übergebene Zeichen trägt.
- Sehen Sie eine `static` Methode zum Verschmelzen von zwei Codebäumen vor, die einen Codebaum zurückliefert, dessen Wurzelknoten als Nachfolger jeweils die Wurzelknoten der verschmolzenen Codebäume hat. Dabei wird lediglich ein neuer Wurzelknoten erzeugt, an den die Wurzelknoten der beiden anderen Codebäume angehängt werden.
- Eine Methode zum Einfügen von Verzweigungsknoten ist nicht nötig, Verzeigungsknoten können durch Verschmelzen erzeugt werden.

Prioritätswarteschlange für Codebäume

- Für jedes Element wird ein Sortierschlüssel gespeichert. Für zwei aufeinanderfolgende Elemente gilt, dass der Sortierschlüssel des vorderen Elements kleiner oder gleich dem Sortierschlüssel des hinteren ist (Prioriätseigenschaft).
- Wird ein neues Element eingefügt, wird es entsprechend seines Sortierschlüssels eingeordnet. Sind Elemente vorhanden mit dem gleichen Sortierschlüssel, wird das neue Element hinter diesen eingeordnet (Warteschlangeneigenschaft).
- Das vorderste Element (d.h. von allen Elementen mit dem kleinsten Sortierschlüssel das älteste) kann ausgelesen werden, dabei wird es aus der Warteschlange entfernt.

Die ausführbare Klasse.

- Der Name einer Texttextdatei wird auf der Kommandozeile übergeben.
- Lesen Sie die Datei zeilenweise mit `java.io.BufferedReader.readLine()` ein und wandeln Sie die Zeilen jeweils in `byte[]` mit `java.lang.String.getBytes()` um. Dadurch wird die Menge der möglichen Zeichen relativ klein und es bietet sich an, ein Feld zu benutzen, um die Vorkommen eines Zeichens zu zählen.
- Der optimale Huffman-Code wird berechnet und ausgegeben.

Berechnung des optimalen Huffman-Code

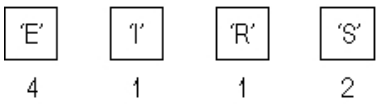
1. Der Text wird gelesen. Dabei wird zu jedem Textzeichen die Häufigkeit seines Vorkommens bestimmt. Damit auch ein leerer Text ein sinnvolles Ergebnis liefert, wird immer das Vorhandensein eines Zeichens mit Code 0 angenommen.
2. Für jedes vorkommende Textzeichen, wird ein Codebaum erzeugt. Jeder Codebaum besteht nur aus einem Blatt, welches das Textzeichen trägt.

   Diese Codebäume werden in eine Prioritätswarteschlange einsortiert, wobei die Anzahl der Vorkommen des jeweiligen Textzeichens als Sortierschlüssel benutzt wird.
3. Die ersten beiden Bäume (kleinste Sortierschlüssel) werden aus der Warteschlange entnommen und verscholzen. Der Sortierschlüssel des neuen Baumes ist die Summe der Schlüssel der verschmolzenen Bäume. Mit diesem Schlüssel wird der neue Baum in die Prioritätswarteschlange einsortiert.
4. Schritt 3. wird solange wiederholt bis die Warteschlange nur noch einen Codebaum enthält, dieser repräsentiert den Huffman-Code.

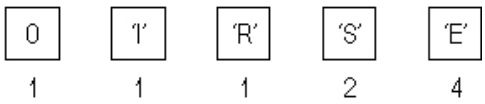
Huffman-Codebaum aus einem Text erstellen

'SEEREISE'

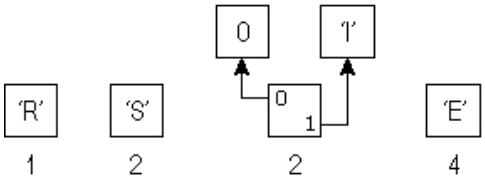
Codebäume



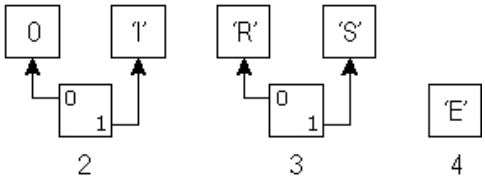
Prioritätswarteschlange (mit 0 beginnen)



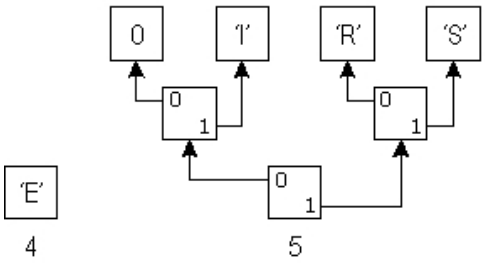
Entnehmen, Verschmelzen, Einfügen



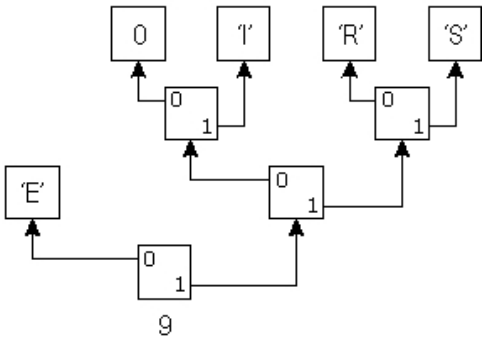
Entnehmen, Verschmelzen, Einfügen



Entnehmen, Verschmelzen, Einfügen



Entnehmen, Verschmelzen, Einfügen



Fertigen Codebaum entnehmen