Huffman Codierung

Die Huffman Codierung ist ein verlustfreies Kompressionsverfahren, das in einer Utilityklasse zu implementieren ist.

Bei der Datenkompression geht es darum, dass ein bestimmter Text, der aus einer beschränkten Anzahl von verschiedenen Zeichen besteht (=Alphabet) möglichst stark komprimiert wird, so dass er nach dem Versenden wieder vollständig - ohne Verluste - zusammengesetzt werden kann.

Die kleinste Einheit bei Kompressionsverfahren ist ein Bit.

Beispiel mit 2 Zeichen

Angenommen ein Text besteht nur aus zwei unterschiedlichen Zeichen, d. h. das Alphabet ist nur zwei Zeichen groß, z. B. "`L`" und "`R`". Folgender Text mit 16 Zeichen ist zu komprimieren:
"**LRRRLLLRLLLRRRRL**"

Dann lässt sich dieser Text optimal komprimieren, in dem man statt "`L`" ein 0-Bit schreibt und statt "`R`" ein 1-Bit.

Codetabelle	
Zeichen	Code (in Bit)
L	0
R	1

Die komprimierte Nachricht kann dann in Bits so dargestellt werden: **0111000100011110**

Da ein char in C# und Java ein Unicode-Zeichen 2 Bytes hat hat die ursprüngliche Nachricht 32 Bytes. Die kodierte Nachricht hat nur 2 Bytes.

Bei der Übermittlung der Nachricht, muss allerdings mit übermittelt werden, welches Zeichen durch 0 und welches Zeichen durch 1 kodiert wird. Dann kann die Nachricht wieder vollständig zusammengesetzt werden.

Beispiel mit 3 Zeichen

Wenn der Text aus drei Zeichen besteht, muss man die Codierung anpassen. Es gibt dann zumindest zwei Zeichen die mit zwei Bits kodiert werden müssen.

Codetabelle	
Zeichen	Code (in Bits)
L	0
R	10
U	11

Die Nachricht "**LRRULLLRLLLURRRL**" wird dann in folgende Bitfolge konvertiert:
010101100010000111010100 (=24 Bits, also 3 Bytes)

Happy coding!

Die Wahl der Codetabelle ist bei mehr als zwei Zeichen für eine optimale Codierung, allerdings nicht mehr beliebig, was leicht zu beweisen ist.

Angenommen wir hätten folgende Codetabelle gewählt.

Codetabelle	
Zeichen	Code
U	0
R	10
L	11

dann wäre "**LRRULLLRLLLURRRL**" folgendermaßen kodiert:
11101001111111011111010101011 (=30 Bits ¿ 24 Bits)

Der Grund ist naheliegend, häufige Zeichen sollten einen kurzen Code haben, Zeichen mit geringerer Häufigkeit, sollten auf die längeren ausweichen.

Huffman Codierung

Der Huffman-Algorithmus ist genau dazu da, eine optimale Codetabelle zu garantieren und ist in auf diversen Seiten im Internet visuell sehr gut erklärt.

`http://www.iti.fh-flensburg.de/lang/algorithmen/code/huffman/huffman.htm`

Aufgabenstellung

Implementiere eine Utility-Klasse **`HuffmanCoding`** die folgende Funktionalitäten zur Verfügung stellt:

- Codieren eines beliebigen Strings in eine Codetabelle. Recherchiere dazu den genauen Algorithmus.
- Berechnung der Bitlänge der kodierten Nachricht
- Erstellen eines Nachrichtenfiles (z. B. Extension `huf`), in dem die Codetabelle und die komprimierte Nachricht geschrieben wird. Die Codetabelle darf natürlich nicht kodiert sein.
- Auslesen eines solchen `huf`-Nachrichtenfiles und Wiederherstellung des ursprünglichen Nachricht.

Hinweise:

- Es ist erlaubt, die Codetabelle und die komprimierten Daten in separate Files zu schreiben. Diese könnten zu einem File gezippet und mit einer eigenen Fileextension versehen werden.
  - In C# gibt es System.IO.Compression.ZipFile zum Handlen von ZipFiles.
  - in Java kann das z. B. java.util.zip.ZipFile.
- In Java und C# gibt es Klassen, die das Arbeiten mit Bits wesentlich erleichtern können.
  - In C# gibt es `BitArray`. `https://www.tutorialspoint.com/csharp/csharp_bitarray.htm`
  - In Java gibt es `BitSet` `https://docs.oracle.com/javase/7/docs/api/java/util/BitSet.html`

Happy coding!