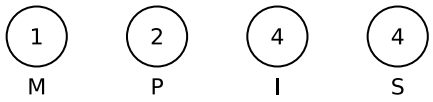
Erstellung eines Hufmann-Baumes

Am Beispiel der Codierung des Textes «**MISSISSIPPI**» soll der Huffman-Algorithmus erläutert werden. Zuerst zählt man, wie oft jedes Zeichen im Text vorkommt und erstellt eine Häufigkeitstabelle.

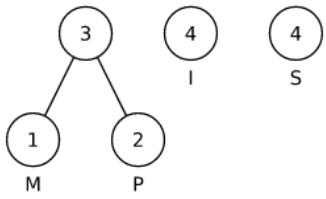
Zeichen	M	P	I	S
Häuffigkeit	1			

Nun geht es darum, einen Codierungsbaum zu erstellen. Die Häufigkeiten der Buchstaben bilden je einen Knoten. Die Häufigkeit steht im Knoten, der Buchstaben darunter. Die Knoten werden nach aufsteigender Häufigkeit sortiert:



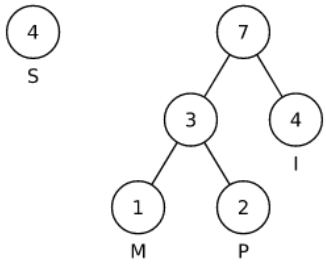
„Knoten“ mit Zeichenhäufigkeit

Nun werden die zwei Knoten mit den kleinsten Häufigkeiten an einen neuen Knoten angehängt. Der neue Knoten enthält die summierte Häufigkeit der ursprünglichen Knoten:



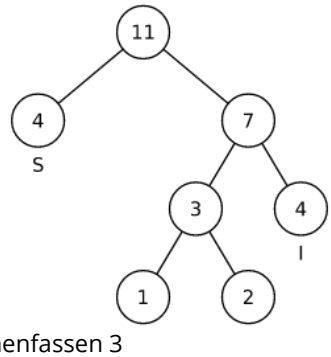
Knoten zusammenfassen 1

Das wird wiederholt, bis alle Knoten miteinander verbunden sind. Wenn zwei Knoten die gleiche Häufigkeit haben, spielt es keine Rolle, welcher gewählt wird:



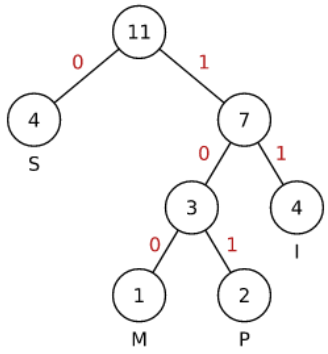
Knoten zusammenfassen 2

Wichtig ist, dass nach jedem Schritt die Knoten wieder neu sortiert werden.



Knoten zusammenfassen 3

Wenn der Baum fertig ist, werden alle Äste, welche nach links zeigen, mit einer «0» markiert, alle die nach rechts zeigen mit einer «1».



Kanten benennen

Nun kann eine Codierungstabelle erstellt werden, indem der Code für jedes Zeichen vom Baum abgelesen wird:

Zeichen	I	M	P	S
Code	11	100	101	0

Codierungstabelle «MISSISSIPPI»

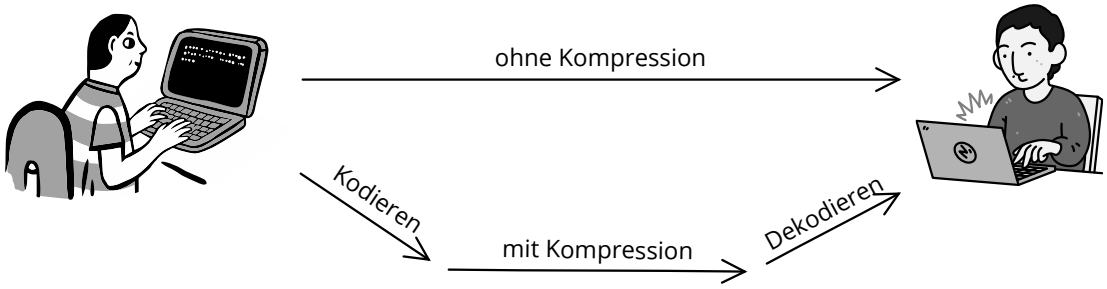
① Aufgabe

- Erstelle zu folgendem Text einen Huffman-Baum und codiere den Text entsprechend:

  EXTERNER EFFEKT

- Wie viel Bit beansprucht das Wort in Huffman-Codierung? Wie viel in ASCII?

Kompression



Bei der Kompression wird versucht, Daten effizienter abzuspeichern, so dass weniger Speicherplatz belegt wird. Gerade beim Übertragen von Daten – z.B. Streamen eines Films, oder grosser Dateien– ist dies enorm wichtig. Wir unterscheiden zwei grundsätzlich unterschiedliche Arten von Kompression:

verlustfreie Kompression

Bei der verlustfreien Kompression, können die Daten vollständig rekonstruiert werden. D.h. es handelt sich – wie bei der Codierung – um eine **eindeutige und umkehrbare Abbildung**.

Anwendung
Überall dort, wo kein Verlust passieren darf – ein Programm läuft nicht mehr, wenn Befehle fehlen!Bilder, Audio und Video während der Produktion und Bearbeitung – sonst würde bei jedem Speichern (also Codieren) die Qualität abnehmen.

Beispiele
flac – Sound
zip, rar – Dateien
gif, png, raw, psd, xcf – Grafik

verlustbehaftete Kompression

Hier werden beim Speichern Daten entfernt. Dadurch kann entsprechend viel Speicher gewonnen werden, allerdings lassen sich die Original-Daten nur noch teilweise rekonstruieren. Es wird versucht vor allem nicht wichtige Daten wegzulassen.

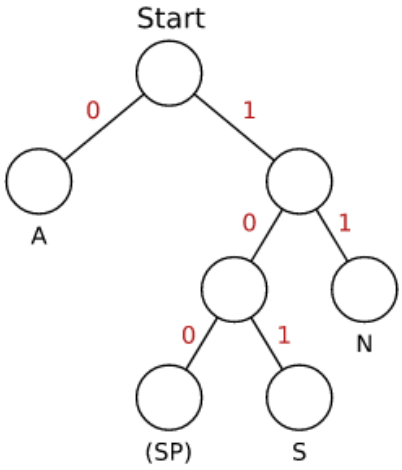
Anwendung
beim Fertigstellen von Medien, also wenn diese nicht mehr weiter bearbeitet werden sollen

Beispiele
mp3 – Sound
jpg – Grafik
mp4, avi, mov – Video

Huffman-Codierung

David Huffman hat 1952 ein Verfahren entwickelt, mit welchem Zeichen platzsparender codiert werden können. Seine Idee ist, dass Zeichen, welche häufig im Text vorkommen, einen kürzeren Code erhalten, als Zeichen, welche selten im Text vorkommen.

Codebaum
Ein Codebaum ist eine Struktur mit einem Startknoten. Von diesem aus geht es entweder nach links oder rechts unten weiter. Eine 0 im Code bedeutet nach links gehen, eine 1 nach rechts gehen. Wenn ein Knoten mit einem Buchstaben erreicht wird, hat man ein Zeichen decodiert, man beginnt wieder von vorne.



Decodiere die folgende Bitfolge mit dem obenstehenden Codebaum. (SP) steht für ein Leerzeichen (engl. space).

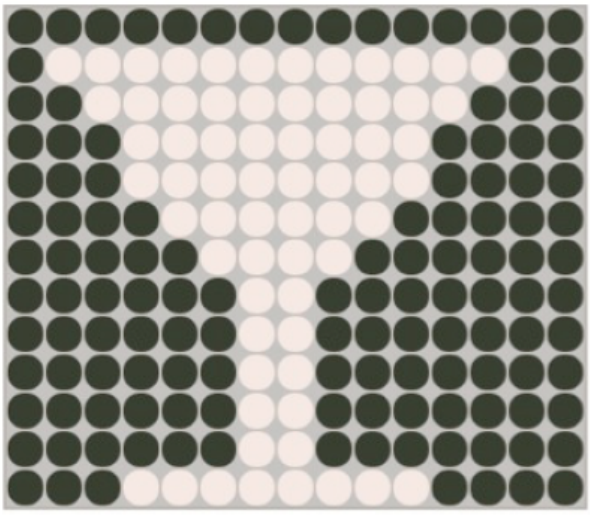
011110101100011011010 1

Codiere den erhaltenen Text nun wahlweise in ASCII und anschliessend in Binärcode.
Wie lange ist die erhaltene Bitfolge?

verlustfreie Komprimierung von Bildern

Wie wir bereits gelernt haben, können Bilder als Raster bestehend aus einzelnen Pixel gespeichert werden. Das Bild unten besteht aus 195 Pixel in 13 Zeilen und 15 Spalten. Dieses Bild besitzt zwei Farbwerte, hell oder dunkel. Ein Pixel kann in diesem Fall mit einem Bit, einer Eins oder einer Null gespeichert werden. Das Bild kann durch das folgende Bitmuster beschrieben werden:

15,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,
1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,
1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,
1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,
1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,
1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,
1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,
1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,
1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,
1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,
1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,
1,1,1,0,0,0,0,0,0,0,0,1,1,1,1



Eine kompaktere Art der Codierung ist die Lauflängen-Methode. Dabei wird gezählt wie viele Pixel der gleichen Farbe aufeinander folgen. Das obere Bild beginnt mit 16 dunklen Pixel, dann folgen 12 helle, danach wieder 4 dunkle, usw. Eine Lauflängen Codierung des obigen Bildes sieht wie folgt aus:

15,0,16,12,4,10,6,8,7,8,8,6,10,4,12,2,13,2,13,2,13,2,13,2,10,8,4

Zur Decodierung des Codes und für das Zeichnen des Bildes wird die Anzahl Spalten benötigt, welche das Bild hat. Dies ist der erste Wert der hier gewählten Lauflängen Codierung danach folgen abwechslungsweise die Anzahl heller und dunkler Pixel. In diesem Fall beginnt es mit 0 hellen Pixel, dann 16 dunklen Pixel, dann 12 hellen, u.s.w.

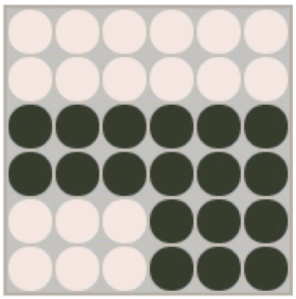
② **Dekodieren - Erzeugen von Pixelbildern**
Zu den folgenden Lauflängen Codes sollen die entsprechenden Bilder gezeichnet werden. Es muss darauf geachtet werden, dass die Anzahl der Spalten entsprechend der ersten Zahl im Lauflängen Code ist. Dekodiere (zeichne) die folgenden Bilder:

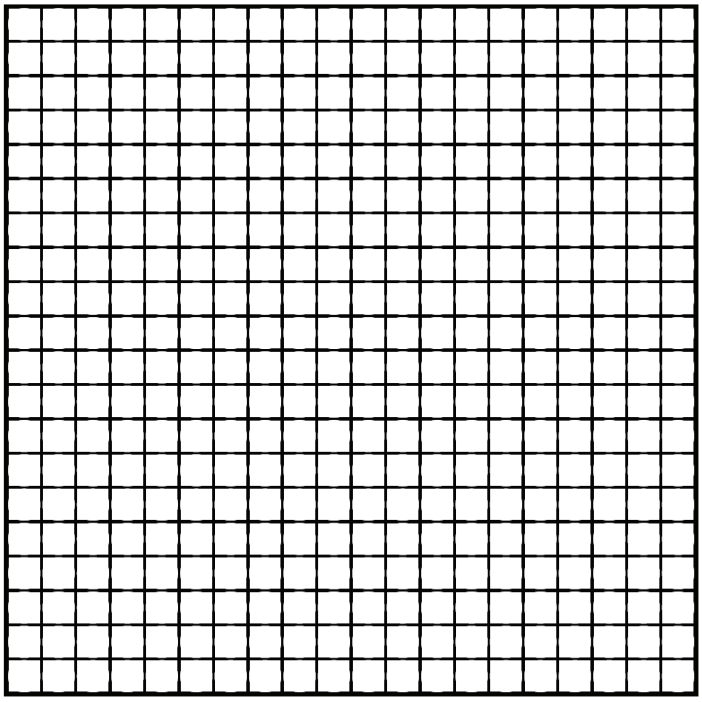
a) 19,0,42,12,8,10,8,1,1,8,1,1,7,2,1,6,1,2,7,3,1,4,1,3,7,3,2,2,2,3,7,2,1,2,2,2,1,2,7,1,1,8,1,1,8,10,8,12,22
b) 19,0,42,5,1,5,7,13,6,13,6,13,6,13,7,11,9,9,11,7,13,5,15,3,17,1,9
c) 15,0,36,1,13,3,11,5,9,7,7,9,5,11,3,13,4,9,6,9,6,3,3,3,6,3,3,3,6,3,3,3,6,3,3,3,4
d) 17,0,25,5,9,8,9,8,9,6,1,1,9,2,5,1,9,1,6,1,9,1,6,1,9,1,4,3,9,1,3,4,6,4,3,4,6,4,3,4,6,3,29

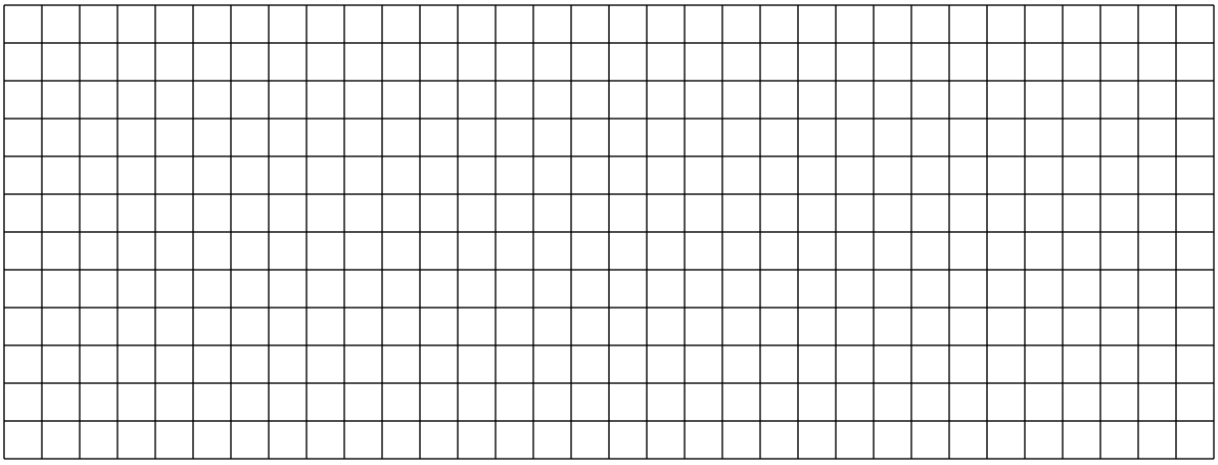
③ **Lauflängen kodieren**

Zeichne in einem leeren Raster ein Muster oder eine Pixelgrafik .
Kodiere nun das erstellte Werk mit Hilfe der Lauflängen Methode.
Die erste Zahl im Code gibt die Anzahl Spalten an, die folgende Zahl die Anzahl helle Pixel. Als Beispiel liefert das Pixel-Muster rechts den folgenden Code.

6,12,12,3,3,3,3







④ **Dekodieren**

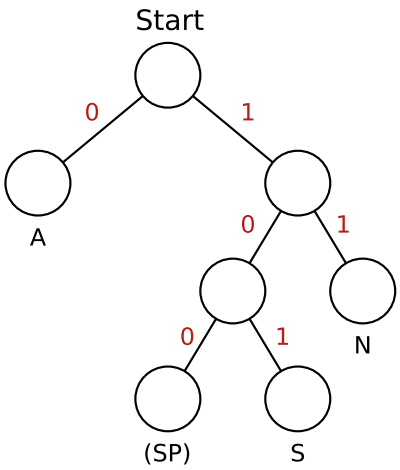
Die erstellten Lauflängen Codierungen können mit Hilfe der elektronischen Beilage überprüft werden. Mit dieser lassen sich auch Pixel-Muster beliebiger Grösse interaktiv erzeugen.
Scanne dazu den nebenstehenden QR-Code und gib deinen Code ein. Klicke anschliessend auf „Decodieren“ und überprüfe dein Bild.

verlustfreie Komprimierung von Texten durch die Huffman Codierung

David Huffman hat 1952 ein Verfahren entwickelt, mit welchem Zeichen platzsparender codiert werden können. Seine Idee ist, dass Zeichen, welche häufig im Text vorkommen, einen kürzeren Code erhalten, als Zeichen, welche selten im Text vorkommen.

Codebaum

Ein Codebaum ist eine Struktur mit einem Startknoten. Von diesem aus geht es entweder nach links oder rechts unten weiter. Eine 0 im Code bedeutet nach links gehen, eine 1 nach rechts gehen. Wenn ein Knoten mit einem Buchstaben erreicht wird, hat man ein Zeichen decodiert, man beginnt wieder von vorne.



Huffmann-Baum für „Anna“

① Decodiere die folgende Bitfolge mit dem nebenstehenden Codebaum.
(SP) steht für ein Leerzeichen (engl. space).

```
1 01111010110001101101 01
```

① Codiere den erhaltenen Text nun in ASCII und anschliessend in Binärcode.
Wie lange ist die erhaltene Bitfolge?