

# Codierung

---

## Codierung von Text

- Auch Buchstaben und andere Zeichen werden als Bitfolge gespeichert
- Es muss eine Festlegung getroffen werden, welche Bitfolge für welches Zeichen steht

- ASCII: American Standard Code for Information Interchange
  - Festgelegt im Jahr 1963, mit späteren Aktualisierungen
- Codierung mit 7 Bits pro Zeichen
  - Später häufig auf 8 Bit erweitert
- Definiert sowohl anzeigbare Zeichen als auch Steuerzeichen

# Aufgabe 1

- Schreibe mit einem Texteditor einen kurzen Text und speichere ihn in einer Datei mit der Endung `.txt`
- Vergleiche die Anzahl der eingegebenen Zeichen mit der Größe der Datei in Byte
- Öffne die Datei mit einem Hex-Editor
  - Suche im Internet nach einer ASCII-Tabelle und vergleiche mit der Anzeige im Hex-Editor
  - Ändere im Hex-Editor einige Bytes ab und betrachte die Datei anschließend wieder mit einem Texteditor

- Wie viele Zeichen lassen sich mit 8 Bits codieren?
  - 256
- Mit Klein- und Großbuchstaben, Ziffern, Sonderzeichen und Steuerzeichen bleibt nicht mehr viel Platz für weitere Zeichen
  - Was ist mit griechischen, kyrillischen, arabischen,... Zeichen?

- Um mehr als 256 unterschiedliche Zeichen verwenden zu können, wurden und werden im Wesentlichen zwei Möglichkeiten verwendet
  - Codepages (Zeichensatztabellen)
  - Unicode

- Tabelle, die die Zuordnung von Zahlen (also Bitfolgen) zu den dadurch repräsentierten Zeichen speichert
  - ASCII ist eine mögliche Codepage
- Eine andere Codepage erlaubt die Verwendung anderer Zeichen
  - Die Gesamtzahl der zur Verfügung stehenden Bitfolgen ändert sich dadurch nicht, sondern nur deren Bedeutung
- Zur Darstellung muss bekannt sein, gemäß welcher Codepage die Daten codiert wurden

- Versuch, alle weltweit benutzten Zeichen einheitlich zu codieren
  - Gewissermaßen eine einzige, gigantische Codepage
- Jedem codierten Zeichen ist ein sogenannter **Codepunkt** (Code Point) zugeordnet
  - Insgesamt umfasst der Standard **1 114 112** Codepunkte, einige sind jedoch reserviert
  - In der aktuellen Version des Standards sind **143 859** Zeichen definiert
  - Die Codepunkte von 0 bis 127 entsprechen exakt der ASCII-Codierung
- Neben aktuellen und historischen Schriftzeichen sind bspw. auch Emojis enthalten



- Unicode trennt die Codierung als Codepunkte von der tatsächlichen Darstellung als Bitfolge
- Verschiedene Formate (sog. “Unicode Transformation Format”) stehen zu Verfügung, zum Beispiel:

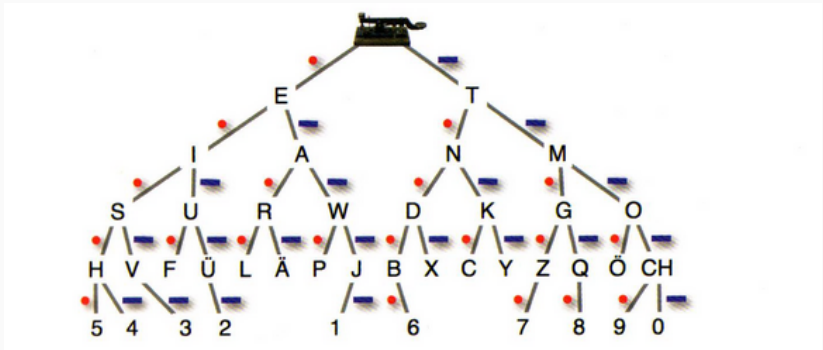
**UTF-32** Codiert jedes Zeichen immer mit 32 Bit

**UTF-16** Codiert Zeichen mit 2 Byte oder mit 4 Byte

**UTF-8** Codiert Zeichen mit variabler Bytezahl, zwischen 1 und 4 Byte

- Java verwendet UTF-16 für Zeichen und Zeichenfolgen

## Exkurs: Morsecode (1)



**Abbildung 1:** Codebaum Morsecode (Bildquelle: Eisenmann, Informatik Klasse 7, CC-BY-NC-SA)

## Exkurs: Morsecode (2)

- Warum ist das “E” mit einer kürzeren Zeichenfolge codiert als bspw. “X”?
  - Der Buchstabe “E” kommt (im Englischen, aber auch im Deutschen) häufiger vor als “X”, daher spart man Platz (und Zeit), wenn man ihn kürzer codiert
- Warum reichen – und . zur Übermittlung einer Nachricht im Morsecode nicht aus?
  - Der Morsecode ist nicht **präfixfrei**: Die Codierung eines Zeichens kann auch das Präfix (“der Anfang”) der Codierung eines anderen Zeichens sein
    - Beispiel: . – . codiert ein “R”, aber . – codiert bereits ein “A”, . codiert ein “E” und – codiert ein “T”
  - Es braucht ein Pausezeichen, um zu signalisieren, wann ein Zeichen zuende ist und das nächste beginnt

- Eine Codierung, die häufige Zeichen “sparsamer”, d. h. mit weniger Bits codiert, wäre praktisch
- Eine präfixfreie Codierung ist wünschenswert, denn wie codieren wir ein Pausezeichen, wenn nur 0 und 1 zur Verfügung steht?
- Wie könnte eine solche Codierung aussehen?

## Aufgabe 2

- Bastel-Vorarbeit
  - Schneide zunächst die winzigen Buchstabenkärtchen vom ersten der zwei ausgeteilten Blätter aus
  - Lege/klebe dann die beiden Blätter so zusammen, dass ein großer, zusammenhängender Baum entsteht
- Codierung
  - Lege die Buchstabenkärtchen von oben nach unten der Reihe nach ( \_ (Leerzeichen), A, B... ) auf die grau hinterlegten Felder (vorletzte Baumebene)
  - Ermittle die Codierung des Buchstabens D
- Optimierte Codierung
  - Schiebe E zwei “Ebenen” nach links (Codierung wird 001)
  - Welches Problem ergibt sich für D und wie lässt es sich lösen?

# Buchstabenhäufigkeiten

- Unser Codebaum soll nicht willkürlich Zeichen “kürzer” oder “länger” codieren, sondern nach Häufigkeit
- Mittlere Buchstabenhäufigkeiten in der deutschen Sprache:

Z	H
–	18,5%
E	14,7%
N	7,96%
I	6,15%
S	5,92%
R	5,7%
A	5,3%

Z	H
T	5,01%
D	4,14%
H	3,88%
U	3,54%
L	2,8%
C	2,49%
G	2,45%

Z	H
M	2,06%
O	2,04%
B	1,54%
W	1,54%
F	1,35%
K	0,99%
Z	0,92%

Z	H
P	0,64%
V	0,55%
J	0,22%
Y	0,03%
X	0,02%
Q	0,02%

## Aufgabe 3

- Ordne die Buchstabenplättchen auf dem Codebaum so an, dass folgendes erfüllt ist
  - Die Zeichen **\_** und **E** sollen mit 3 Bit codiert sein
  - Die Zeichen **N**, **I**, **S** und **R** sollen mit 4 Bit codiert sein
  - Die nächsten 8 Zeichen in der Tabelle (**A** bis **G**) sollen mit 5 Bit codiert sein
  - Alle verbleibenden Zeichen sollen mit 6 Bit codiert sein
- Ermittle, wie viele Bit zur Codierung des Worts **SPAGHETTIEIS** mit Deinem Codebaum benötigt werden
  - Vergleiche mit einer Codierung mit einer festen Länge von 5 Bit pro Zeichen

- Der Huffman-Algorithmus ermittelt für eine vorgegebene Menge von Zeichen und deren Häufigkeiten einen optimalen Codebaum
- Ablauf:
  1. Suche die beiden Zeichen mit der geringsten Häufigkeit und fasse sie zu einem neuen “Superzeichen” zusammen
  2. Ordne dem so erzeugten “Superzeichen” als Häufigkeit die Summe der Häufigkeiten der beiden Einzelzeichen zu
  3. Wiederhole diese Schritte, bis alle (Super)Zeichen zusammengefasst sind
- Beispiel: A (16) B (7) C (5) D(5) E (3)
  - siehe Tafelbild :-)



- Erstelle einen Codebaum mit dem Huffman-Algorithmus für
  - A (5) B (7) C (10) D (12) E (3) F (8) G (9)

- Durch den Huffman-Algorithmus ist eine verlustfreie Kompression der Daten möglich
  - Verlustfrei heißt, dass sich die Originaldaten wieder exakt rekonstruieren lassen