Leitprogrammartige Unterrichtsunterlagen zur Huffman-Codierung

Julia Imhof
24.06.2019

Inhaltsverzeichnis

Einführung

Das Thema dieser Unterrichtseinheit ist die *Huffman-Codierung*. Die LPU sind für eine Doppellektion von 90 Minuten konzipiert. Die SuS besuchen das mathematisch-naturwissenschaftliche Gymnasium und befinden sich im zweiten Quartal des zweijährigen Ergänzungsfachs Informatik.

Vorwissen und neue Konzepte

Diese LPU setzen folgendes Vorwissen voraus

Allgemeines Vorwissen

- Die SuS wissen, was eine binäre Datendarstellung ist: Sie wissen, dass der Computer Zahlen, Text und Bilder als eine Folge von Bits darstellt.
- Sie können Dezimalzahlen in das Binärsystem umrechnen und umgekehrt.
- Die SuS haben sich im Rahmen einer Einführung in endliche Automaten mit den Konzepten Alphabete, Wörter und Sprachen beschäftigt und kennen deren Definitionen.
- Aus dem Mathematikunterricht wissen die SuS, was relative und absolute Häufigkeiten und eine Häufigkeitsverteilung sind.

Die SuS haben das Buch “einfach INFORMATIK 7-9 Programmieren Sekundarstufe I” bis einschliesslich Kapitel 7 durchgearbeitet.

Algorithmisches Denken und Programmierung (TigerJython)

- Die SuS kennen den Begriff des Algorithmus.
- Sie können einfache prozedurale Programme schreiben.
- Sie wissen, was Unterprogramme sind: Sie kennen die Technik des modularen Entwurfs.
- Die SuS beherrschen die Kontrollstrukturen, d.h. die Schleifen while, for, repeat und die If- Bedingung/If-then-else-Bedingung (if/elif/else).
- Die SuS kennen die Konzepte der Parameter und Variablen und verstehen den Unterschied zwischen Variablen und Parametern.
- Die SuS wissen, was ein Befehl (mit und ohne Parameter) im Kontext der Programmierung ist.
- Sie beherrschen die elementaren Datenstrukturen und Arrays.
- Die Klasse hat sich auch bereits mit der Rekursion auseinandergesetzt.

Die SuS haben das Buch “einfach INFORMATIK 7-9 Strategien entwickeln Sekundarstufe I“ durchgearbeitet und haben sich intensiv mit dem Konzept des Greedy-Algorithmus auseinandergesetzt.

Graphen

- Datenstruktur Graph: Die SuS kennen den Graphen als Struktur, die aus Knoten und Kanten besteht.
- Sie kennen die mathematische Definition eines Graphen G=(V, E), wobei V die Menge der Knoten und E die Menge der Kanten ist.
- Die SuS haben sich intensiv mit der graphischen und binären Darstellung von Graphen beschäftigt. (Kapitel 2 und 3 des Buches „einfach INFORMATIK 7-9 Strategien entwickeln Sekundarstufe I“)

Bäume

- Datenstruktur Bäume: Die SuS kennen den Baum als spezielle Form eines Graphen. Sie wissen, dass die Elemente eines Baums Knoten genannt werden, und dass sie durch Kanten verbunden sind. Sie kennen den Begriff des Vorfahren: Sie wissen, dass alle Knoten bis auf die Wurzel einen Vorfahren haben, und dass der Vorfahre eines Knotens auch Vater des Knotens genannt wird. Sie kennen die Begriffe Nachfolger und Kinder: Alle direkten Nachfolger eines Knotens sind die Kinder des Knotens. Sie wissen, dass Blätter Knoten ohne Kinder sind, und dass man die Knoten, die weder Wurzel noch Blätter sind, innere Knoten nennt. Sie kennen auch den Begriff Grad eines Knotens als maximale Anzahl der Nachfolgeknoten.

- Die SuS kennen die rekursive Definition eines Baums: Ein Baum T ist entweder leer oder er besteht aus einer Wurzel w mit 0..n Teilbäumen t_0.. t_n.

- Die SuS kennen die mathematische Definition und Notation eines Baums T=(V, E), bestehend aus einer Menge von Knoten V und einer Menge von gerichteten Kanten E. Die Wurzel $w \in V$ hat nur Ausgangskanten. Alle anderen Knoten haben genau eine Eingangskante. Für alle Kanten $e \in E$ gilt $e = (v_1,v_2)$, wobei $v_1,v_2 \in V$ und $v_1 \neq v_2$.

Binäre Bäume

- Die SuS wissen, dass ein binärer Baum ein Baum von Grad 2 ist, d.h. dass jeder Knoten eines binären Baums höchstens zwei Nachfolger hat.

Folgendes Vorwissen muss aktiviert werden:

- Das Konzept des binären Baums und seiner Bestandteile Wurzel, innere Knoten, Kanten und Blätter.
- Das Konzept des Greedy-Algorithmus, dies ist eine Methode, mit der eine gute, wenn auch nicht die garantiert beste Lösung für ein Optimierungsproblem gefunden werden kann..

Folgende Konzepte müssen neu eingeführt werden

- Der Begriff der *Textkompression*, welche es ermöglicht Text so darzustellen, dass er weniger Speicherplatz benötigt und schneller übertragen werden kann.

- Der Begriff der *verlustlosen Textkompression*: Der Ursprungstext kann nach der Kompression ohne Verlust von Information originalgetreu wiederhergestellt werden.

- Der Begriff der *präfixfreien Codierung*: Kein Codewort ist Anfang eines anderen Codeworts. Die *Huffman-Codierung* konstruiert ein eindeutiges *Codewort* für jedes Textzeichen.

- Der *Huffman-Algorithmus* mit den folgenden Konzepten:
    - Das Konzept der *Häufigkeitstabelle*, welche die Häufigkeit jedes Textzeichens im Text enthält.
    - Das Konzept des *Huffman-Baums*, welcher aus der *Häufigkeitstabelle bottom-up* aufgebaut wird.
    - Das Ablesen der binären *Codewörter* aus dem *Huffman-Baum.*

- Das Konzept der *mittleren Codewortlänge*, welche angibt, wie viele Binärzeichen im Durchschnitt pro Textzeichen benötigt werden.

- Es muss besprochen werden, dass die *Huffman-Codierung optimal* ist: Die *Huffman-Codierung* erstellt immer das kürzest mögliche *Codewort*, d.h. man kann einen Text mit Hilfe einer *Häufigkeitstabelle* nicht besser codieren.

Lernziele

Leitidee

Kompressionsverfahren sind in der heutigen mobilen Zeit wichtiger denn je. Text, Musik, Bilder und Videos sollen so schnell wie möglich im Internet übertragen oder platzsparend abgespeichert werden.
David Huffman entwickelte im Jahre 1952 ein heute noch sehr beliebtes Verfahren zur verlustlosen Kompression von Daten. Dieses Verfahren, die Huffman-Codierung, zeichnet sich durch ihre Präfixfreiheit und Optimalität aus. Präfixfreiheit bedeutet, dass ein Codewort, welches durch eine Kompression entsteht, nie der Anfang eines anderen Codeworts ist. So kann jedes Codewort auch eindeutig dekodiert werden. Weiter ist die Huffman-Codierung optimal, d.h. man kann einen Text (mit Hilfe der Häufigkeitsverteilung) nicht besser codieren. Aus diesen Gründen ist die Kenntnis der Kompression von Daten und insbesondere der Kompression durch die Huffman-Codierung Teil der Allgemeinbildung und muss im Unterricht behandelt werden.
Die intensive Auseinandersetzung mit dem Huffman-Algorithmus und seinen Teilschritten, also dem Aufbau der Häufigkeitstabelle, dem Aufbau des Huffman-Baums aus der Häufigkeitstabelle und das Bestimmen der Codewörter aus dem Huffman-Baum, fördert die Kompetenz des algorithmischen Denkens. Die SuS sollen das Konzept der Huffman-Codierung auch kennenlernen als weiteres Beispiel für einen Greedy-Algorithmus.

Dispositionsziel 1

Die *Huffman-Codierung* ist für die SuS eine Einführung in das Thema Kompressionsverfahren. Die Kenntnis der *verlustlosen Huffman-Codierung* wird deren Interesse für die *verlustbehaftete Kompression* wecken und die Motivation schaffen, weitere Verfahren zu entdecken und miteinander zu kontrastieren.

Dispositionsziel 2

Die SuS erkennen in einer geeigneten Aufgabenstellung, wann sie die *Huffman-Codierung* einsetzen können.

Operationalisiertes Ziel 1

Die SuS verstehen, dass *Kompressionsverfahren* ermöglichen, Daten so darzustellen, dass weniger Speicherplatz benötigt wird und die Daten schneller übertragen werden können. Sie sind in der Lage, dies einem/einer MitschülerIn zu erklären.

Operationalisiertes Ziel 2

Die SuS kennen die Begriffe *verlustlose Kompression, optimale Codierung, mittlere Codewortlänge* und *Präfixfreiheit* und sie sind in der Lage, sie einem/einer MitschülerIn zu erklären.

Operationalisiertes Ziel 3

Die SuS verstehen die *Huffman-Codierung* konzeptuell: Sie sind in der Lage, einem/r MitschülerIn ad hoc in eigenen Worten und anhand der *Häufigkeitstabellen* und des *Huffman-Baums* zu erklären, wie das *Verfahren der Huffman-Codierung* funktioniert.

Operationalisiertes Ziel 4

Die SuS sind in der Lage, aus einer *Häufigkeitstabelle* von Hand einen *Huffman-Baum* zu erstellen.

Operationalisiertes Ziel 5

Die SuS sind in der Lage, das *Codewort* für ein Textzeichen aus dem *Huffman-Baum* zu bestimmen.

Operationalisiertes Ziel 6

Die SuS sind in der Lage, gegeben ein *Huffman-Baum*, einen Text zu komprimieren und auch wieder herzustellen.

Concept Maps

Die nachfolgende Concept Map visualisiert die den LPU zugrundeliegenden Reflexionen.

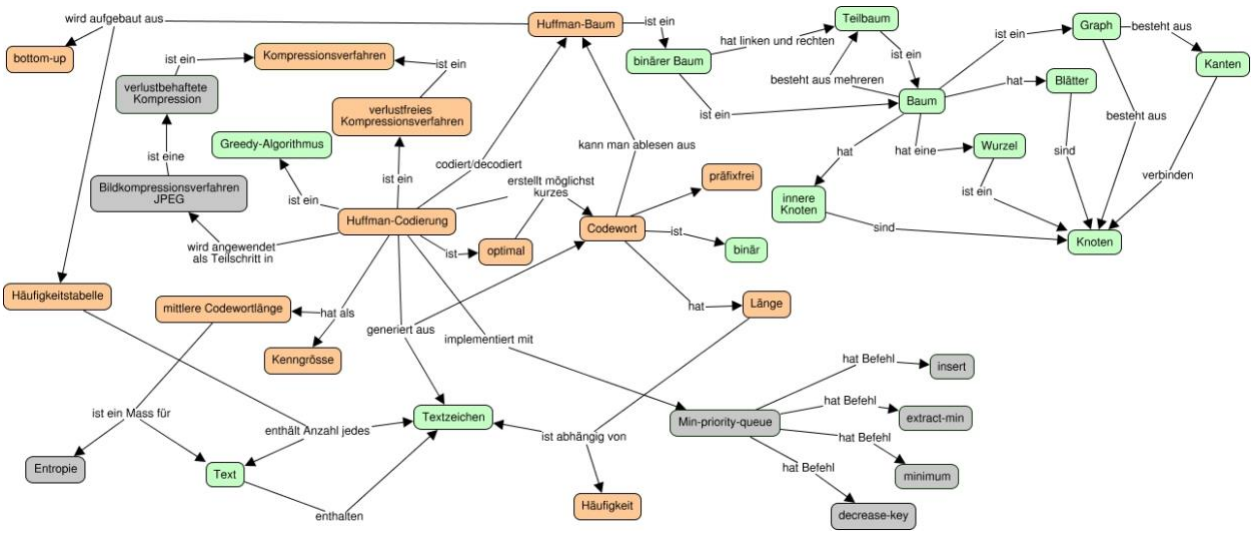


Abbildung 1 Concept Map Vorwissen und neue Konzepte

Die Concept Map in Abbildung 1 hebt das für die Doppellektion vorausgesetzte Vorwissen (in Grün) und die zu vermittelnden Fachbegriffe (in Orange) hervor. Das Vorwissen, auf dem die LPU aufgebaut werden, beschränkt sich auf die wichtigsten Begriffe, die für eine erfolgreiche Auseinandersetzung mit dem neuen Stoff von Bedeutung sind.

Diese LPU legen den Fokus auf die *verlustlose Kompression* von Text. Das Konzept der *verlustbehafteten Kompression* (eingesetzt in der Kompression von Bild-, Audio- und Videodateien) ist bewusst in Grau gehalten. Es gehört zwar zum Thema *Huffman-Codierung* (*Huffman-Codierung* wird oft als Teilschritt in *verlustbehafteten Kompressionsverfahren* eingesetzt), in der Doppellektion wird aber nicht näher auf die Konzepte eingegangen. Ebenfalls in Grau ist die Datenstruktur *Min-priority-queue aufgeführt.* Diese Datenstruktur wird in der Programmierung der *Huffman-Codierung* eingesetzt, um aus einer *Häufigkeitstabelle* einen *Huffman-Baum* aufzubauen. Für eine 90-minütige Einführung in die *Huffman-Codierung* ist dies zu viel neuer Stoff, dieser könnte aber in einer weiteren Doppelstunde eingeführt werden, wenn die Programmierung der *Huffman-Codierung* Thema ist. Die *Entropie* ist ebenfalls in Grau gehalten. Sie ist zwar eine wichtige Grösse in der Informationstheorie, in der Einführung in die *Huffman-Codierung* muss sie aber nicht zwingend behandelt werden.

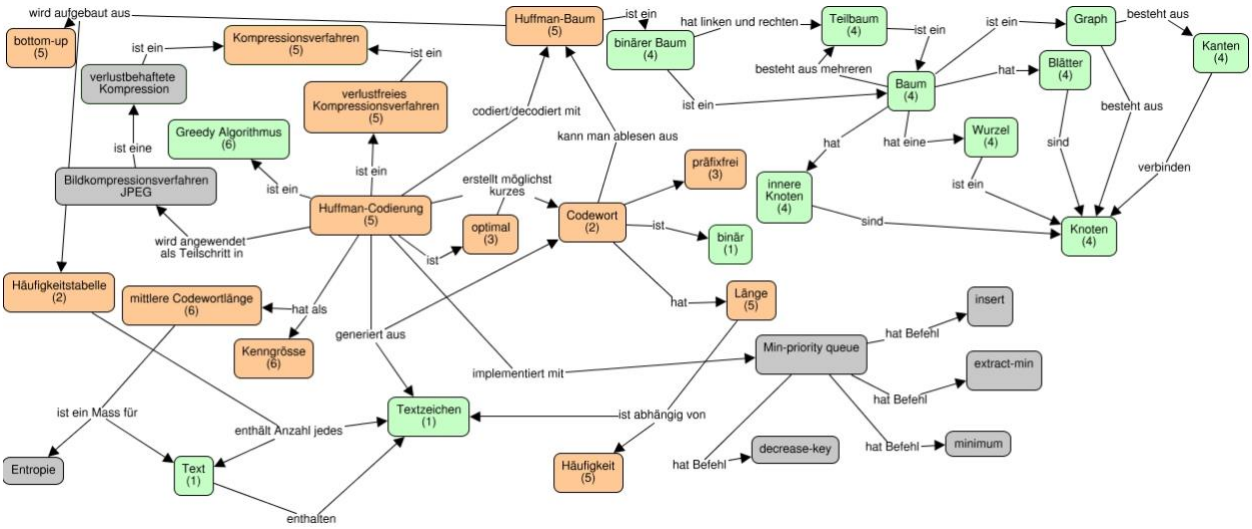


Abbildung 2 Lernzielgraph

Die zweite Concept Map (in Abbildung 2) stellt den Lernzielgraphen der LPU dar und erklärt die Sequenzierung des Unterrichtsstoffes.

Mit einer kognitiv aktivierenden Aufgabenstellung werden die SuS in die *verlustlose Textkompression* eingeführt. Die SuS entdecken, dass geeignete Codewörter *präfixfrei* sein müssen. Darüber hinaus erkennen sie, dass man für eine *optimale Codierung* die Häufigkeiten der Buchstaben in einem Text ausnutzen muss. Die *Häufigkeitstabelle* wird eingeführt und die SuS sehen, dass man mit Codewörtern variabler Länge (häufig vorkommende Textzeichen werden mit kurzen Codewörtern, selten vorkommende Textzeichen werden mit längeren Codewörtern codiert) Bits einsparen kann.

Nachdem das Vorwissen über binäre Bäume anhand geeigneter Aufgaben aktiviert worden ist, wird das neue Konzept *Huffman-Baum* eingeführt: Die SuS lernen, wie sie aus der Häufigkeitstabelle einen *Huffman-Baum bottom-up* aufbauen und die *Codewörter* für jedes Textzeichen aus dem *Huffman-Baum* ablesen können. Ebenfalls lernen sie, wie sie den codierten Text anhand des *Huffman-Baums* decodieren können.

Nachdem die SuS den *Huffman*-Algorithmus geübt haben, werden seine Eigenschaften näher untersucht. Die SuS setzen sich mit der *mittleren Codewortlänge* auseinander und erkennen, dass sie den gleichen Wert hat, auch wenn es für einen Text mehrere Möglichkeiten gibt, einen Huffman-Baum aufzubauen. Die SuS erkennen, dass es sich beim *Huffman*-Algorithmus um einen Greedy-Algorithmus handelt. Sie verstehen, wo sich der Algorithmus greedy (gierig) verhält, und mit dem Wissen über die *mittlere Codewortlänge* erkennen sie, wieso er eine *optimale Codierung* konstruiert.
Dabei geht es nicht darum, mathematische Sätze und Beweise zu verstehen, sondern es geht darum, eine Intuition für die Konzepte zu bekommen.

Die LPU schliessen mit einer Zusammenfassung der erlernten Konzepte. Danach folgt eine Wissenssicherungsphase “Teste dich selbst”. Am Ende werden den SuS weitere Aufgaben zum Thema angeboten sowie die Musterlösungen zu allen Aufgaben.

Huffman-Codierung

Einführung

Alice und Bob sind gute Freunde und gehen zusammen ins Gymnasium. Sie reden täglich miteinander. Nach der Matura wird Alice nach Lausanne ziehen, um an der EPFL zu studieren. Bob möchte in Zürich bleiben, um sein Studium an der ETH zu beginnen. Da Alice und Bob weiterhin in Kontakt bleiben wollen, möchten sie sich elektronische Nachrichten schreiben. Wie du weisst, müssen Nachrichten, damit sie übertragen werden können, in eine Folge von Bits (d.h. 0 oder 1) umgewandelt werden. Da Alice und Bob im Ergänzungsfach Informatik noch keine Verfahren behandelt haben, die es ihnen ermöglichen, ihre Nachrichten in Bitfolgen umzuwandeln, wollen wir ihnen helfen, ein geeignetes Verfahren zu finden.

Um die Lesbarkeit zu erhöhen, findest du im Text folgende Symbole:


Denkaufgaben, die dich bei der Bearbeitung der LPU unterstützen und dir helfen sollen, den Stoff besser zu verstehen;


Aufgaben zur Vertiefung des gelernten Stoffs.

Beispiel 1

Nehmen wir an, Alice' und Bobs Nachrichten bestehen nur aus dem Alphabet mit den Buchstaben a, b, d, k und r - also aus dem Alphabet ∑={a, b, d, c, r}. Alice möchte Bob eine erste Nachricht „abracadabra" schicken.


Damit die Nachricht übertragen werden kann, müssen Alice und Bob für jeden Buchstaben im gegebenen Alphabet eine Folge von Bits bestimmen. Hilfst du ihnen dabei? Wie würdest du die Buchstaben mit einer Folge von 0 und 1 darstellen?

Versuch 1

Alice und Bob haben beschlossen, folgende Bitfolgen – wir nennen sie ***Codewörter*** oder ***Codes*** – für die Buchstaben zu verwenden: a = 0, b = 1, d = 01**,** c = 10**,** r = 11; als ***Codewort-Tabelle*** wie folgt dargestellt:

Buchstabe	Codewort
A	0
B	1
D	01

C	10
R	11

Alice hat eine Nachricht ***codiert***, d.h. eine Nachricht mit Hilfe der Codewörter in eine Bitfolge umgewandelt: 011101000101110.

Welche Nachricht kann Bob aus der Bitfolge lesen? Hilf ihm, die Nachricht zu lesen, d.h. die Bitfolge in Buchstaben umzuwandeln. Ist die Nachricht eindeutig? Ist diese Menge von Codewörtern eine gute Wahl zur verständlichen Kommunikation?

Bob kann die Nachricht nicht eindeutig ***decodieren,*** d.h. er kann die Bitfolge nicht eindeutig den Buchstaben zuordnen. Der Anfang der Bitfolge „01“ kann z.B. „ab“ sein oder auch „d“, weiter kann die Bitfolge „11“ entweder „bb“ oder „r“ sein. Alice und Bob erkennen, dass die gewählten Codewörter nicht geeignet sind, da sie keine eindeutige Decodierung zulassen.

Wie müssen Alice und Bob die Codewörter wählen, damit sie beim Lesen der Nachricht die Buchstaben eindeutig zuordnen können? Was fällt dir auf, wenn du z.B. die Codewörter für „a“ und „d“ genauer anschaust?

Für eine eindeutige Decodierung ist es wichtig, dass ein Codewort nie ein Anfang eines anderen Codewortes ist, d.h. in unserem Beispiel ist a = 0 und d = 01 nicht zulässig, da „a“ den Anfang des Codewortes für „d“ bildet.

Welche anderen Codewörter in Alice‘ und Bobs erstem Versuch bilden den Anfang voneinander? Welche anderen Buchstaben können verwechselt werden?

Das Codewort für „b“ bildet den Anfang der Codewörter für „k“ und „r“.

Versuch 2

Alice und Bob wählen nun andere Codewörter mit fixer Länge für ihre Buchstaben.

Buchstabe	Codewort
a	000
b	001
d	010
c	011
r	100

Alice‘ erste Nachricht „abracadabra“ ist nun 000001100000011000010000001100000.

Kann Bob nun die Nachricht eindeutig lesen? Kann er die Codewörter eindeutig den Buchstaben zuordnen?

Da die Codewörter in Versuch 2 eine fixe Länge haben, ist klar, wann ein neues Codewort beginnt bzw. aufhört, und so kann die Nachricht eindeutig decodiert werden!

Alice und Bob schicken sich nun Nachrichten hin und her und sie bemerken, dass die Speicherung der Nachrichten viel Platz benötigt und dass das Senden viel zu lange dauert. Sie möchten die Nachrichten mit weniger Bits speichern und versenden, d.h. sie wollen die Nachrichten ***komprimieren***. Dabei wollen sie aber immer noch die Nachricht selbigen Inhalts verschicken und keine Information verlieren (d.h. nicht einfach die Nachricht kürzen!) .
Sie besprechen die Wahl der Codewörter und fragen sich, ob sie nicht andere Codewörter für ihre Nachrichten verwenden können.

Hast du eine Idee, was die beiden verbessern könnten? Schaue dir dazu das Wort „abracadabra“ genauer an. Was fällt dir auf?

Im Wort „abracadabra“ kommt der Buchstabe „a“ fünfmal vor, der Buchstabe „b“ zweimal, der Buchstabe „r“ zweimal und die Buchstaben „c“ und „d“ je ein einziges Mal.

Buchstabe	Häufigkeit
a	5
b	2
d	1
c	1
r	2

Wie können wir die Codewörter wählen, sodass die ***Codierung*** der Nachricht so wenig Bits wie möglich benötigt, aber trotzdem jeder Buchstabe eindeutig codiert wird?

Versuch 3

Um Bits einzusparen, haben Alice und Bob die Idee, häufig vorkommende Buchstaben mit kurzen Codewörtern darzustellen. Sie überlegen sich aus diesem Grund folgende Codewörter:

Buchstabe	Codewort
A	0
B	101
D	1001
C	1000
R	11

Überprüfe die Codewörter! Bildet ein Codewort den Anfang eines anderen Codewortes? Schreibe dir Alice’ erste Nachricht „abracadabra“ mit den neuen Codewörtern auf. Zähle die Anzahl Bits in der codierten Nachricht und vergleiche sie mit der Anzahl Bits, die benötigt wurde, um „abracadabra“ mit den Codewörtern aus Versuch 2 zu codieren. Wie viele Bits sparen Alice und Bob für diese Nachricht ein?

Kein Codewort bildet den Anfang eines anderen Codewortes. Alice‘ Nachricht sieht wie folgt aus:
0101110100001001010111 0.
In Versuch 2 erfordert die Codierung jedes Codewortes 3 Bits. Die Nachricht «abracadabra» besteht aus 11 Buchstaben, das heisst es werden 33 Bits benötigt.
Die neue Codierung mit den verschieden langen Codewörtern in Versuch 3 benötigt 23 Bits – Alice und Bob können 10 Bits einsparen.

Aufgabe 1

Alice und Bob beschliessen nun, ein anderes Alphabet zu benutzen: ∑={i, m, s, p}. Die erste Nachricht, die sie versenden wollen, ist „mississippi“:

a) Alice und Bob wählen folgende Codewörter:

Buchstabe	Codewort
I	0
M	1
S	10
P	11

Die erste Nachricht ist also 10101001010011110. Ist diese Codierung eindeutig? Wenn dies nicht so ist, finde andere Wörter als „mississippi“, die durch diese binäre Folge kodiert sind.

b) Alice und Bob wählen folgende Codewörter:

Buchstabe	Codewort
i	000
m	001
S	010
P	011

Die erste Nachricht ist also 001000010010000010010000011011000. Ist diese Codierung eindeutig? Wenn dies so ist, wieso kann die Nachricht eindeutig decodiert werden? Was können Alice und Bob an den Codewörtern verbessern?

c) Im dritten Versuch wählen Alice und Bob folgende Codewörter:

Buchstabe	Codewort
i	11
m	100
s	0
p	101

Ersetze die Buchstaben in „mississippi“ durch ihre Codewörter. Ist diese Codierung eindeutig? Wie viele Bits sparen Alice und Bob im Vergleich zur Codierung in Aufgabe b) ein?

Huffman-Verfahren

Wir kennen nun eine Eigenschaft, die uns eine zulässige Codierung von Nachrichten garantiert: Kein Codewort darf den Anfang eines anderen Codewortes bilden – dies nennt man ***präfixfrei***, d.h. kein Codewort darf Präfix eines anderen Codewortes sein.

Die Codierung muss ***verlustlos*** sein, d.h. die Nachricht muss eindeutig und ohne Verlust von Information decodiert werden können.

Weiter wollen wir eine Codierung finden, welche die Redundanz in den Nachrichten ausnutzt: Häufig vorkommende Buchstaben wollen wir mit kürzeren Codewörtern codieren, selten vorkommende Buchstaben können längere Codewörter haben.

Bis jetzt waren die Alphabete in unseren Beispielen auf wenige Buchstaben begrenzt und die Codierung konnte durch Ausprobieren einfach bestimmt werden. Dieses Verfahren ist für Alphabete mit einer grösseren Anzahl von Buchstaben nicht geeignet.

Wir wollen also für Alice und Bob ein Verfahren finden, welches, gegeben sind ein Alphabet und die Häufigkeit der Buchstaben in diesem Alphabet, eine präfixfreie Codierung mit kürzeren Codewörtern für häufig vorkommende und mit längeren Codewörtern für selten vorkommende Buchstaben erzeugt.
Ein solches Verfahren hat David Huffman (1925-1999) im Jahre 1952 im Rahmen seiner Doktorarbeit am MIT[1] entwickelt. Seine Idee ist es, aus den gegebenen Buchstaben und ihren Häufigkeiten einen binären Baum aufzubauen. Aus diesem Baum können die Codewörter für die Codierung und bei der Decodierung die Buchstaben abgelesen werden.

Abbildung 3
D.Huffman 1999

Repetition binäre Bäume

Weisst du noch, was ein binärer Baum ist? Bevor wir uns anschauen, wie das Verfahren von Huffman funktioniert, wollen wir einige wichtige Begriffe zum Thema binäre Bäume repetieren. Dazu kannst du die folgenden Aufgaben lösen:

Aufgabe 2

a) Beschrifte im binären Baum in Abbildung 4 die Bestandteile: Wurzel, Kante, innerer Knoten und Blatt.
b) Beschreibe die Eigenschaften (Anzahl der Vorgänger bzw. Nachfolger) der Bestandteile: Wurzel, innerer Knoten, Blatt.

[1] Massachussetts Institute of Technology, Technische Hochschule in den USA: http://www.mit.edu/

c) Welchen Grad hat ein binärer Baum? Wie viele Nachfolger hat ein Knoten im binären Baum höchstens?
d) Wie nennt man den direkten Vorfahren eines Knotens ausserdem ? Gibt es eine andere Bezeichnung für die direkten Nachfolger eines Knotens?
e) Wie lautet die rekursive Definition eines binären Baums?

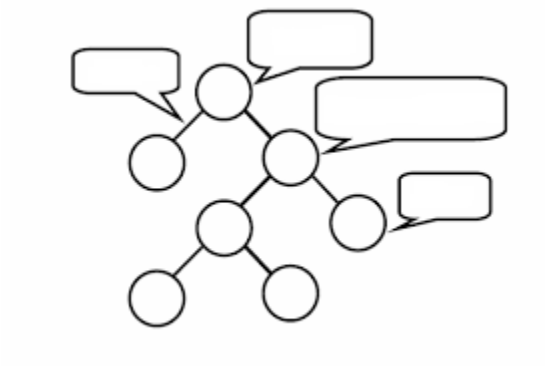


Abbildung 4 Binärer Baum

Aufbau des Huffman-Baums

Nun wollen wir uns aber den Aufbau eines ***Huffman-Baums*** anschauen. Der Baum wird ausgehend von den Blättern ***bottom-up*** aufgebaut:

1. Trage die Buchstaben in die Blätter ein. Deren Häufigkeiten kannst du unter das jeweilige Blatt schreiben.
2. Fasse die beiden vaterlosen Blätter mit den geringsten Häufigkeiten in einem Vaterknoten zusammen und addiere die Häufigkeiten der Kinder. Trage die Summe im Vaterknoten ein.
3. Führe das Verfahren fort, stets die beiden Blätter (Buchstaben) bzw. Vaterknoten mit den geringsten Häufigkeiten zu einem weiteren Knoten zusammenzuführen und die Summe einzutragen, bis es nur noch einen vaterlosen Knoten gibt (die Wurzel).
4. Den linken Kanten wird je eine 0, den rechten je eine 1 zugewiesen.

Schauen wir es uns an einem Beispiel an und kommen nochmals auf die erste Nachricht von Alice und Bob zurück: „abracadabra“. Wir wissen, dass die Buchstaben mit gewissen Häufigkeiten vorkommen und stellen daraus eine ***Häufigkeitstabelle*** auf:

Buchstabe	Häufigkeit
A	5
B	2
D	1
C	1
R	2

Beispiel 2

Schritt 1

Es werden die Buchstaben und ihre Häufigkeiten in die bzw. unter die Blätter eingetragen:

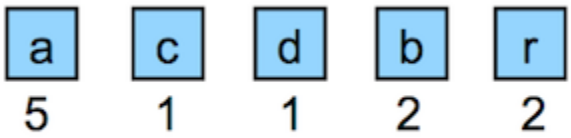


Abbildung 5 Huffman-Baum - Schritt 1

Schritt 2

Es gibt noch vaterlose Knoten: a, b, c, d und r. Fasse die beiden vaterlosen Blätter mit den geringsten Häufigkeiten (c und d haben beide den Wert 1) in einem Vaterknoten zusammen, indem die Häufigkeiten der Kinder addiert werden. Trage die Summe 2 im Vaterknoten ein.

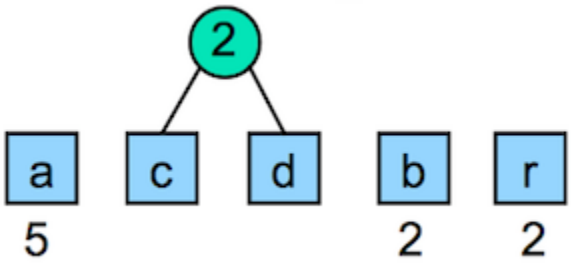


Abbildung 6 Huffman-Baum - Schritt 2

Schritt 3

Es gibt noch vaterlose Blätter: a, b, r. Fasse die zwei Knoten mit den geringsten Häufigkeiten in einem Vaterknoten zusammen (z.B. den Vaterknoten mit dem Wert 2 und Blatt b mit dem Wert 2) und addiere die Häufigkeiten der Kinder. Trage die Summe 4 im Vaterknoten ein.

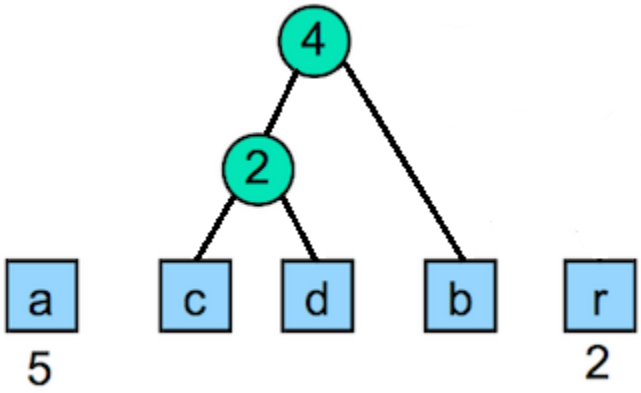


Abbildung 7 Huffman-Baum - Schritt 3

Schritt 4

Es gibt noch vaterlose Blätter: a, r. Fasse die zwei Knoten mit den geringsten Häufigkeiten (den Vaterknoten mit dem Wert 4 und Blatt r mit dem Wert 2) in einem Vaterknoten zusammen und addiere die Häufigkeiten der Kinder. Trage die Summe 6 im Vaterknoten ein.

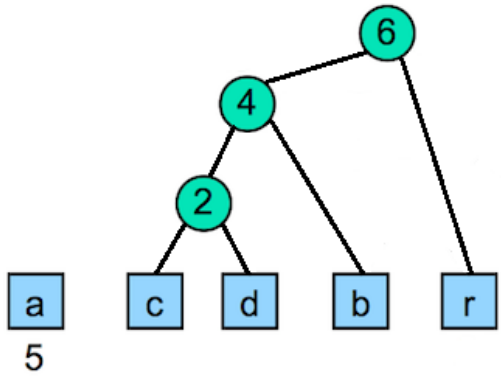


Abbildung 8 Huffman-Baum - Schritt 4

Schritt 5

Es gibt noch ein vaterloses Blatt: Blatt a. Fasse die zwei Knoten mit den geringsten Häufigkeiten (den Vaterknoten mit dem Wert 6 und das Blatt mit Wert 5) in einem Vaterknoten zusammen und addiere die Häufigkeiten der Kinder. Trage die Summe 11 im Vaterknoten ein.

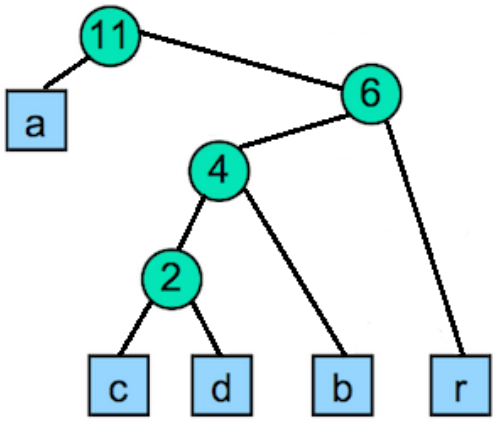


Abbildung 9 Huffman-Baum - Schritt 5

Schritt 6

Es gibt kein vaterloses Blatt mehr und noch genau einen vaterlosen Knoten (die Wurzel). Den linken Kanten wird je eine 0, den rechten je eine 1 zugewiesen.

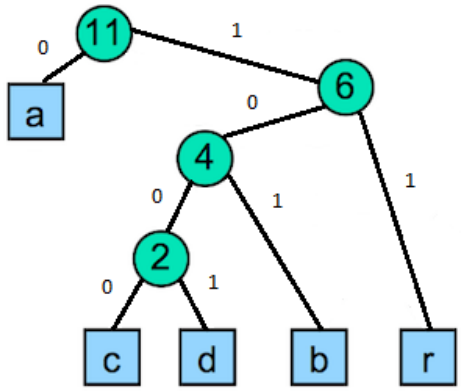


Abbildung 10 Huffman-Baum - Schritt 6

Codierung der Buchstaben aus dem Huffman-Baum ablesen

Nun kannst du die Codierung für die verschiedenen Buchstaben ablesen. Starte bei der Wurzel und folge den Kanten bis zu einem Blatt. Schreibe dir 0 auf, wenn du nach links gehst und 1, wenn du nach rechts gehst.

Codewort für „a“ bestimmen

Von der Wurzel gehst du erst nach links und erreichst Blatt „a“. Du bist einmal nach links gegangen, also ist das Codewort für „a“ gleich „0“.

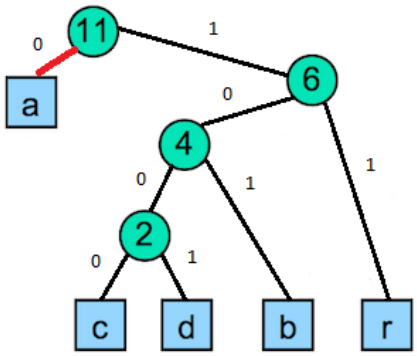


Abbildung 11 Codewort für „a“ bestimmen

Codewort für „r“ bestimmen

Für „r“ gehst du einmal nach rechts und schreibst dir „1“ auf (siehe Abbildung 12). Danach gehst du nochmals nach rechts und schreibst dir wieder „1“ auf (siehe Abbildung 13), d.h. für „r“ schreibst du dir das Codewort „11“ auf.

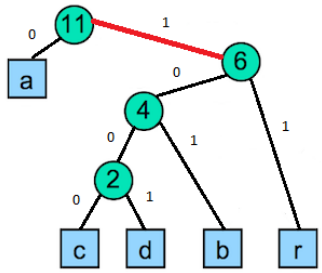


Abbildung 12 Codewort für „r“ bestimmen - 1. Schritt

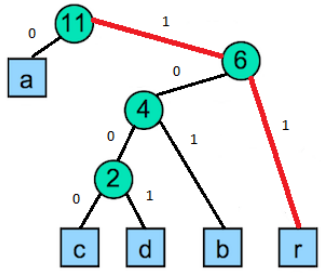


Abbildung 13 Codewort für „r“ bestimmen - 2. Schritt

Für die restlichen Blätter wählst du das gleiche Vorgehen und erhältst am Schluss folgende Codewörter:

Buchstabe	Codewort
A	0
B	101
D	1001
C	1000
R	11

Die Codierung von „abracadabra“: 01011101000010010101110. (Siehe auch Versuch 3 in Beispiel 1.)

Überprüfe, ob die Codewörter präfixfrei sind. Haben die häufig vorkommenden Buchstaben ein kürzeres Codewort? Was ist mit den selten vorkommenden Buchstaben?

Ja, die Codewörter sind präfixfrei, kein Codewort bildet den Anfang eines anderen Codewortes. Buchstaben, die häufiger vorkommen, sind a,b und r, sie haben kürzere Codewörter als c und d, die selten vorkommen.

Aufgabe 3

In Beispiel 2 wählen wir in Schritt 3 den Vaterknoten und das Blatt b, um den nächsten Vaterknoten im Baum zu bilden. Wie würde der Baum aussehen, wenn wir stattdessen die beiden Blätter b und r zusammenfassen (sie haben ja ebenfalls die minimale Häufigkeit 2)?

a) Zeichne den Huffman-Baum, wenn du wie oben beschrieben vorgehst.
b) Schreibe dir die Codewörter für die Buchstaben auf. Was fällt dir auf?
c) Codiere „abracadabra“ mit den neuen Codewörtern und vergleiche die Anzahl Bits mit der Anzahl Bits, die du für die Codierung von „abracadabra“ brauchst, wenn du mit den Codewörtern aus Beispiel 2 arbeitest.

Aufgabe 4

a) Zeichne den Huffman-Baum für „mississippi“ aus folgender Häufigkeitstabelle:

Buchstabe	Häufigkeit
I	4
M	1
S	4
P	2

b) Schreibe die Codewörter für die Buchstaben m, i, s und p auf.
c) Codiere „mississippi“ mit den Codewörtern aus Teilaufgabe b).

Decodierung des codierten Worts

Was passiert nun, wenn Bob eine codierte Nachricht von Alice erhält und diese decodieren will? Um sie decodieren zu können, muss Bob natürlich wissen, wie der Huffman-Baum aussieht, ansonsten weiss er nicht, wie Alice codiert hat. Er geht Bit für Bit durch das codierte Wort und geht den Baum wie folgt ab:

1. Starte bei der Wurzel.
2. Wenn eine 0 im codierten Wort ist, gehe nach links.
3. Wenn eine 1 im codierten Wort ist, gehe nach rechts.
4. Angekommen bei einem Blatt, schreibe den Buchstaben im Blatt auf und beginne wieder bei der Wurzel (bei 1.).

Nehmen wir unser Beispiel: Du hast den Huffman-Baum und die Bitfolge: 010111010000100101011110.

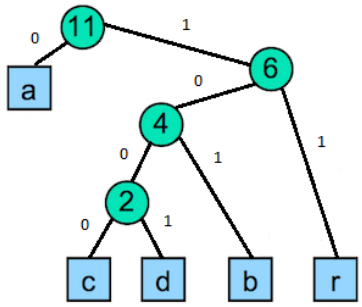


Abbildung 14 Huffman-Baum

Decodierung erster Buchstabe

Zuerst steht eine eine „0“ in der Bitfolge, also gehst du nach links im Baum. Du endest bereits in einem Blatt „a“ und schreibst dir „a“ auf (siehe Abbildung 15):

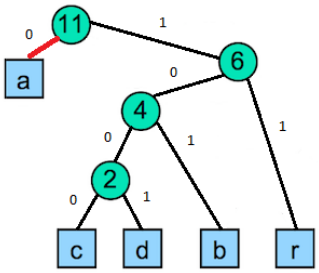


Abbildung 15 Decodierung erster Buchstabe

Decodierung zweiter Buchstabe

Dann beginnst du wieder bei der Wurzel. Das nächste Bit ist eine 1, du gehst nach rechts im Baum (siehe Abbildung 16), das nächste Bit ist eine 0, du gehst nach links im Baum (siehe Abbildung 17). Das nächste Bit ist wieder eine 1, du gehst nach rechts im Baum und kommst zu Blatt „b“ (siehe Abbildung 18). Du schreibst dir „b“ auf.

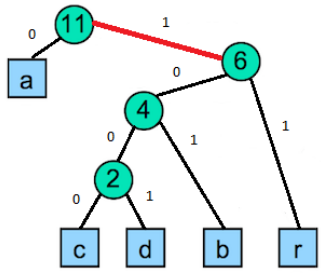


Abbildung 16 Decodierung zweiter Buchstabe - Schritt 1

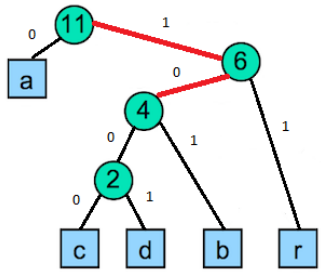


Abbildung 17 Decodierung zweiter Buchstabe - Schritt 2

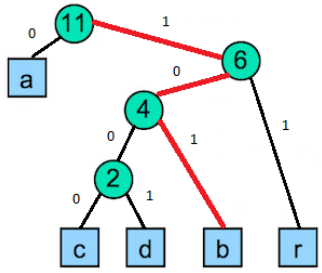


Abbildung 18 Decodierung zweiter Buchstabe - Schritt 3

Danach fängst du wieder bei der Wurzel an. Du wiederholst dies so lange, bis die Bitfolge fertig ist und du das codierte Wort in decodierter Form hast.

Aufgabe 5

Gegeben sind die folgenden Buchstaben und ihre Häufigkeiten:

Zeichen	R	I	C	H	T	G
Häufigkeit	25	21	19	18	13	4

a) Erstelle einen Huffman-Baum und bestimme das Codewort für jeden Buchstaben.
b) Decodiere die folgende Nachricht (von links nach rechts!) 0010110100110100111 mittels des in Teilaufgabe a) generierten Huffman-Baums.

Aufgabe 6

Gegeben sind die folgenden Buchstaben und ihre Häufigkeiten:

Zeichen	A	C	D	H	I	N	R	T	U	W
Häufigkeit	2	2	2	2	2	1	1	1	1	1

a) Generiere eine Huffman-Codierung für die Buchstaben und gib die entsprechende Code-Tabelle an.
b) Überführe den Satz „ICH WAR DA DU NICHT“ mittels der in Teilaufgabe a) generierten Codewörter in eine codierte Bitfolge.

In dem Abschnitt Aufgaben zur Vertiefung des gelernten Stoffs findest du noch weitere Aufgaben, um den Aufbau des Huffman-Baums und die Codierung/Decodierung zu üben.

Optimalität der Huffman-Codierung

Alice und Bob sind glücklich. Sie haben ein Verfahren gefunden, um ihre Nachrichten zu codieren und zu decodieren. Das Verfahren ist verlustlos und erzeugt präfixfreie Codewörter.

Beim Aufbau des Huffman-Baums für „abracadabra“ (siehe Beispiel 2) ist den beiden aber aufgefallen, dass es nicht nur einen eindeutigen Huffman-Baum gibt. In Aufgabe 3 hast du bereits einen weiteren Huffman-Baum für „abracadabra“ aufgebaut. In Abbildung 19 siehst du diese beiden Huffman-Bäume nebeneinander.

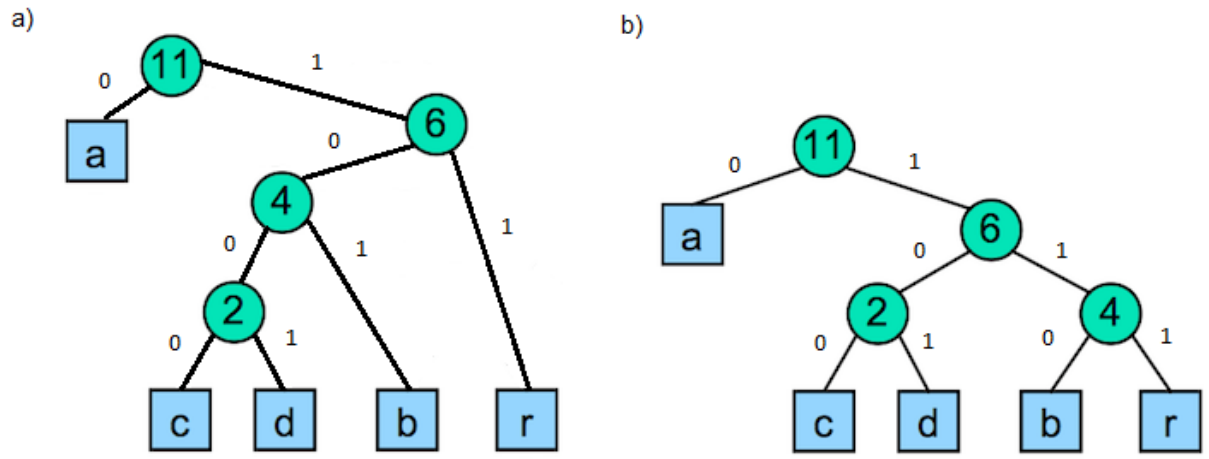


Abbildung 19 Zwei Huffman-Bäume für "abracadabra"

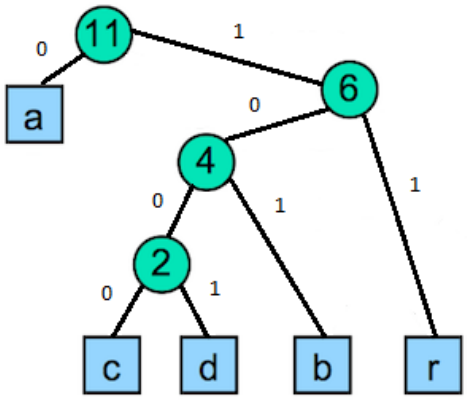
Alice und Bob fragen sich nun, ob vielleicht einer der beiden Huffman-Bäume besser ist, d.h. kürzere Codewörter erstellt als der andere, und sie „abracadabra“ vielleicht noch kürzer codieren können.

Hast du eine Idee, wie Alice und Bob bestimmen können, welchen Baum sie für ihre Codierungen wählen sollen?

Um zu bestimmen, welchen der beiden Bäume sie wählen sollen, berechnen sie die ***mittlere (durchschnittliche) Codewortlänge***. Sie gibt an, wie viele Binärzeichen im Durchschnitt für die Codierung eines Buchstabens benötigt werden.

Möglichkeit 1

Alice bestimmt die mittlere Codewortlänge für den Baum a) (siehe Abbildung 20):



Buchstabe	Codewort
a	0
b	101
d	1001
c	1000
r	11

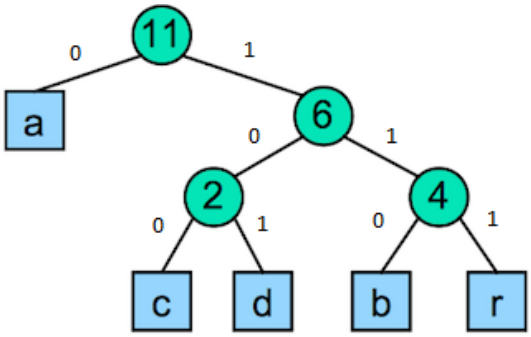
Abbildung 20 Huffman-Baum a) und Codewort-Tabelle für "abracadabra"

Buchstabe „a“ kommt mit einer Häufigkeit von 5 in der Nachricht vor. Die Nachricht ist 11 Buchstaben lang, also kommt Buchstabe „a“ mit einer relativen Häufigkeit von 5/11 in der Nachricht vor. Das Codewort für Buchstabe „a“ ist ein Bit lang, also trägt er 5/11*1 zur mittleren Codewortlänge bei. Buchstabe „b“ kommt zweimal in der Nachricht vor, hat also eine relative Häufigkeit von 2/11. Das Codewort für Buchstabe „b“ ist drei Bit lang, also trägt er 2/11*3 zur mittleren Codewortlänge bei. Wenn wir für die restlichen Buchstaben d, c und r in gleicher Weise vorgehen und die so entstandenen Produkte aufsummieren, erhalten wir die folgende mittlere Codewortlänge:

5/11 * 1 + 2/11 * 3 + 1/11 * 4 + 1/11 * 4 + 2/11 * 2 = 23/11 = 2.09 Bits pro Buchstabe.

Möglichkeit 2

Bob schaut sich den Baum b) an (siehe Abbildung 21) und geht ebenso wie Alice vor.



Buchstabe	Codewort
a	0
b	110
d	101
c	100
r	111

Abbildung 21 Huffman-Baum b) und Codewort-Tabelle für "abracadabra"

Buchstabe „a“ kommt mit einer Häufigkeit von 5 in der Nachricht vor. Die Nachricht ist 11 Buchstaben lang, also kommt Buchstabe „a“ mit einer relativen Häufigkeit von 5/11 in der Nachricht vor. Das Codewort für Buchstabe „a“ ist ein Bit lang, also trägt er 5/11*1 zur mittleren Codewortlänge bei. Buchstabe „b“ kommt zweimal in der Nachricht vor, hat also eine relative Häufigkeit von 2/11. Das Codewort für Buchstabe „b“ ist drei Bit lang, also trägt er 2/11*3 zur mittleren Codewortlänge bei. Wenn wir für die restlichen Buchstaben d, c und r identisch vorgehen und die Produkte aufsummieren, erhalten wir folgende mittlere Codewortlänge:

5/11 * 1 + 2/11 * 3 + 1/11 * 3 + 1/11 * 3 + 2/11 * 3 = 23/11 = 2.09 Bits pro Buchstabe.

Alice und Bob stellen fest, dass sich die berechneten mittleren Codewortlängen nicht unterscheiden! Keiner der Bäume erstellt also bessere oder kürzere Codewörter. Die Anzahl der Binärzeichen, welche im Durchschnitt für die Codierung eines Buchstabens benötigt werden, ist bei beiden Bäumen gleich.

Aufgabe 7

Es gibt noch eine dritte Möglichkeit, einen Huffman-Baum für „abracadabra“ aufzubauen. Findest du heraus, wie er aussieht?

a) Erstelle den Huffman-Baum.
b) Bestimme die Codewörter und schreibe sie in einer Codewort-Tabelle auf.
c) Bestimme die mittlere Codewortlänge. Was fällt dir auf?

Es gibt einen weiteren Weg, die mittlere Codewortlänge zu berechnen, und zwar indem man die Werte der inneren Knoten des Huffman-Baums aufsummiert.
Um dies besser zu veranschaulichen, verwenden wir im Huffman-Baum anstelle der Häufigkeiten die relative Häufigkeit der Buchstaben in der Nachricht (siehe Abbildung 22) .

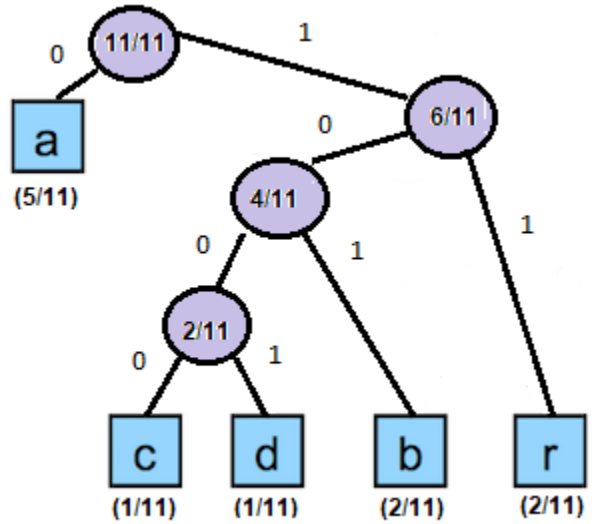


Abbildung 22 Huffman-Baum mit relativen Häufigkeiten

Die folgende Tabelle soll veranschaulichen, aus welchem Grund das funktioniert. Die linke Spalte besteht aus allen Werten der inneren Knoten (*kursiv*). Die rechte Spalte zeigt, wie der Wert der inneren Knoten durch die Summe der Blätter gebildet wird.

2/11			1/11+1/11
4/11	=	*2/11*+2/11 =	1/11+1/11+2/11
6/11	=	*4/11*+2/11 =	1/11+1/11+2/11+2/11
11/11	=	*6/11*+5/11 =	1/11+1/11+2/11+2/11+5/11

Wenn wir die Werte in der linken Spalte aufsummieren, ergibt dies 23/11=2.09 – die mittlere Codewortlänge! Wenn wir die rechte Spalte aufsummieren, sieht man, dass die mittlere Codewortlänge von 23/11 auch die Summe von viermal dem Wert 1/11, nochmals viermal dem Wert 1/11, dreimal dem Wert 2/11, zweimal dem Wert 2/11 und einmal dem Wert 5/11 ist.

Allgemein können wir sagen, dass in einem Huffman-Baum (d.h. in einem Baum, in dem der Wert jeden Knotens die Summe der Werte seiner Kinder ist) die gewichtete Summe der Blätter (wobei das Gewicht die Anzahl Kanten von den Blättern zur Wurzel ist) gleich der Summe der inneren Knoten ist.

Schauen wir uns dazu nun einen allgemeineren Huffman-Baum an (siehe Abbildung 23). Die Buchstaben in den zwei Blättern haben die relative Häufigkeit 1/11 und 1/11 und deren Codewörter haben eine Länge von d Bits. Im Baum sind diese Blätter Kinder des inneren Knotens mit dem Wert 2/11, welcher mit der Wurzel durch d-2 innere Knoten verbunden ist (a_1 bis a_{d-2}).

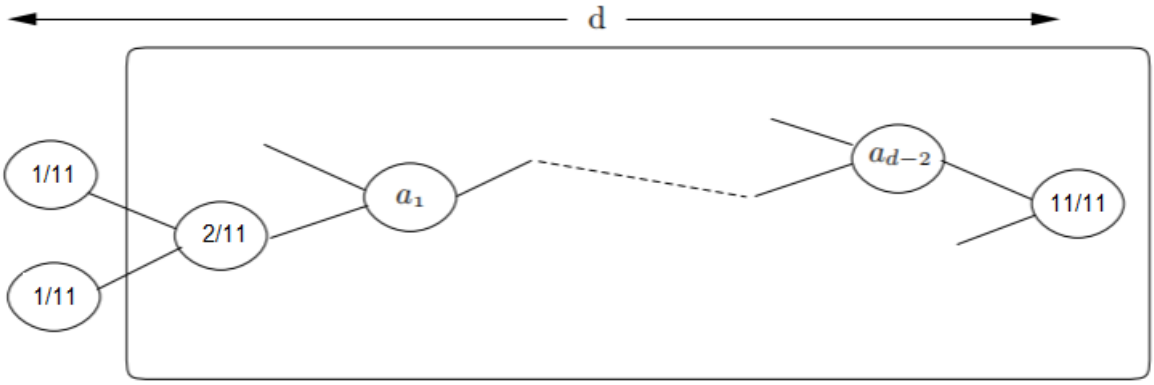


Abbildung 23 Huffman-Baum

Die folgende Tabelle hat d Zeilen und zeigt, dass die beiden Werte 1/11 und 1/11 in den inneren Knoten genau d-mal vorkommt. Summieren wir die Werte aller inneren Knoten auf, gibt es eine Summe, welche 1/11 * d und 1/11 * d der beiden Blätter enthält. Da diese zwei Blätter beliebig sind, ist es klar, dass die Summe gleiche Beiträge von den anderen Blättern enthält, also ist die Summe die mittlere Codewortlänge. Da die Summe gleich der Summe der linken Spalte ist, ist es klar, dass die Summe der Werte der inneren Knoten ebenfalls gleich der mittleren Codewortlänge ist.

2/11				1/11+1/11
a_1	= *2/11*	+…	=	1/11+1/11+ …
a_2	= a_1	+…	=	1/11+1/11+ …
…	=			1/11+1/11+ …
a_{d-2}	= a_{d-3}	+…	=	1/11+1/11+ …
11/11	= a_{d-2}	+…	=	1/11+1/11+ …

Alice und Bob verstehen nun, dass es zwar verschiedene Huffman-Bäume und somit andere Mengen von Codewörtern für die Buchstaben in einer Nachricht gibt, die mittlere Codewortlänge aber für alle die gleiche ist.

Nun fragen sie sich aber, ob die Huffman-Codierung optimal ist, d.h. ob sie auch die kürzest möglichen Codewörter erstellt. Dazu wollen wir uns das Verfahren etwas genauer anschauen:

Um codieren und decodieren zu können, muss aus den Buchstaben und deren Häufigkeiten im Text der Huffman-Baum aufgebaut werden. Dabei werden immer die beiden Knoten (Blätter oder innerer Knoten) mit der aktuell geringsten Häufigkeit ausgewählt.

Du kennst bereits eine Klasse von Algorithmen, welche eine Lösung aufbaut, indem sie die aktuell beste Alternative aussuchen. Wie heissen diese Algorithmen?

Diese Algorithmen heissen Greedy-Algorithmen. Greedy-Algorithmen bestimmen die Lösung eines Problems in mehreren Schritten. Bei jedem Schritt wird versucht, die vielversprechendste Möglichkeit auszuwählen.

Wie du weisst, lösen Greedy-Algorithmen Optimierungsprobleme. Welches Optimierungsproblem wird denn mit dem Huffman-Verfahren gelöst?

Mit dem Huffman-Verfahren wollen wir eine optimale Codierung finden. Optimal bedeutet, dass die Codierung die mittlere (durchschnittliche) Codewortlänge minimiert. Wir wollen, dass die Anzahl Binärzeichen, welche im Durchschnitt für die Codierung eines Buchstabens benötigt wird, minimal ist.

Damit wir sehen, wie die Huffman-Codierung eben genau diese mittlere Codewortlänge minimiert, fassen wir es sprachlich formaler:

- Wir möchten einen Text komprimieren. Die Länge des Textes ist M Buchstaben lang.
- Die Zeichen, aus denen der Text besteht, nennen wir Buchstaben und bezeichnen sie mit b_1, b_2, ….. b_n.
- Die Wahrscheinlichkeit des Buchstaben b_i (seine relative Häufigkeit im Text) nennen wir p_i. Berechnet wird die relative Häufigkeit wie folgt:

$$p_i = \frac{absolute\ Häufigkeit\ von\ p_i}{M}.$$

Bei der Codierung ersetzen wir jeden Buchstaben im Originaltext durch ein Codewort für diesen Buchstaben. Formaler ausgedrückt: Der Buchstabe b_i im Originaltext wird durch ein binäres Codewort der Länge l_i ersetzt. Gesucht ist ein optimaler Code. Optimal bedeutet, dass der Code die mittlere (durchschnittliche) Codewortlänge minimiert.

Berechnet wird die mittlere Codewortlänge L wie folgt: $L = \sum_{i=0}^{n} p_i * l_i$.

Wir können annehmen, die Buchstaben b_1, b_2, b_3,..., b_n seien bereits so nummeriert, dass die Wahrscheinlichkeiten der Buchstaben absteigend sortiert sind: $p_1 >= p_2 >= p_3 >= ... >= p_n$. Wir fordern, dass die wahrscheinlicheren Buchstaben kürzere Codewörter und die seltenen Buchstaben längere Codewörter haben sollen. Das heisst, es soll gelten: $l_1 <= l_2 <= l_3 <= ... <= l_n$. Genau das erreicht die Huffman-Codierung: Je höher die Wahrscheinlichkeit für einen Buchstaben im Text ist, desto kürzer ist das Codewort für diesen Buchstaben. Warum ist das so wichtig? Nun, es ist anhand der Formel zur Berechnung von L leicht einzusehen, dass die

mittlere Codewortlänge L nur dann minimal sein kann, wenn eben die Codewortlängen aufsteigend sortiert sind.

Anmerkung: Der Algorithmus ist natürlich kein Beweis für die Optimalität des Huffman-Codierung! Der Beweis baut aber auf den Eigenschaften des Algorithmus auf.

Teste dich selbst

1. Was ermöglichen Kompressionsverfahren wie die Huffman-Codierung?
2. Welches sind die Eigenschaften der Huffman-Codierung?
3. Was bedeutet Präfixfreiheit? Warum ist sie für eine Codierung wichtig?
4. Erkläre, wie – gegeben sind die Buchstaben und ihre Häufigkeiten - ein Huffman-Baum aufgebaut wird.
5. Wie werden die Codewörter aus dem Huffman-Baum abgelesen?
6. Wie kann man eine mit Huffman codierte Bitfolge decodieren? Was muss bekannt sein?
7. In welche Klasse von Algorithmen gehört die Huffman-Codierung und erkläre, warum das so ist.
8. Beschreibe in eigenen Worten, was die mittlere Codewortlänge ist.

Zusammenfassung

In diesem Kapitel hast du ein Verfahren zur ***verlustlosen Textkompression***, die ***Huffman-Codierung***, kennengelernt. Bei der ***Textkompression*** wird der Text so verdichtet, dass der benötigte Speicherplatz sinkt und die Übertragungszeit verkürzt wird. ***Verlustlos*** bedeutet, dass der Ursprungstext nach der Kompression originalgetreu wiederhergestellt werden kann.
Die Idee des Verfahrens von ***Huffman*** beruht auf einem einfachen Prinzip: Statt jedem Buchstaben im Text ein gleich langes ***Codewort*** zuzuordnen, bekommen Buchstaben, die häufig im Text vorkommen, ein kürzeres Codewort als selten vorkommende Buchstaben.

Als Hilfsmittel verwendet die ***Huffman-Codierung*** einen binären Baum - den ***Huffman-Baum***. Die Buchstaben und die Werte ihrer Häufigkeiten im Text bilden die Blätter des Baumes. Der Baum wird aus diesen Blättern ***bottom-up*** wie folgt aufgebaut: Es werden immer die zwei Knoten mit der geringsten Häufigkeit zu einem Vaterknoten zusammengefasst und die Häufigkeiten der Kinder addiert. Die Summe der Häufigkeiten der Kinder ist der Wert im Vaterknoten.
Sobald der Baum nur noch einen vaterlosen Knoten hat (die Wurzel), ist der ***Huffman-Baum*** fertig aufgebaut. Die Kanten, die links aus einem Knoten abgehen, werden mit „0“ beschriftet, die Kanten, die rechts aus dem Knoten abgehen, werden mit „1“ gekennzeichnet.

Das ***Codewort*** für einen Buchstaben liest man aus dem ***Huffman-Baum*** ab, indem man von der Wurzel ausgehend im Baum bis zum gesuchten Buchstaben am Ende des Baumes (im Blatt) wandert und die dabei gefundenen Beschriftungen an den Kanten („0“ bzw. „1“) notiert. Bei der Decodierung einer Huffman-codierten Bitfolge geht man ebenso vor: Von der Wurzel ausgehend geht man nach links, wenn in der Bitfolge eine „0“ steht und rechts, wenn eine „1“ vorkommt. Sobald man in einem Blatt angekommen ist, hat man einen Buchstaben decodiert und fängt wieder bei der Wurzel an. Dies wiederholt man so lange, bis man am Ende der Bitfolge angelangt ist.

Die ***Codewörter*** der ***Huffman-Codierung*** sind ***präfixfrei***, d.h. kein Codewort ist Anfang eines anderen Codewortes. Das Huffman-Verfahren erstellt immer die kürzest mögliche Codierung, d.h. man kann einen Text mit Hilfe der Häufigkeiten nicht besser codieren. Die Codierung ist ***optimal,*** d.h. sie minimiert die ***mittlere Codewortlänge***. Diese gibt an, wie viele Binärzeichen im Durchschnitt für die Codierung eines Buchstabens benötigt werden.

Aufgaben

Aufgabe 8

Gegeben sind folgende Buchstaben und ihre Häufigkeiten:

Zeichen	E	B	T	N	A	D	S
Häufigkeit	64	13	8	5	4	3	3

a) Generiere eine Huffman-Codierung und gib die entsprechende Code-Tabelle an.
b) Dekodiere folgende Nachricht 1000111111010111010111001010 mittels der in Teilaufgabe a) generierten Codewörter.

Aufgabe 9

Gegeben sind folgende Buchstaben und ihre Häufigkeiten:

Zeichen	E	L	F	G	I	O	R	V
Häufigkeit	2	2	2	2	2	1	1	1

a) Generiere eine Huffman-Codierung für die Buchstaben und gebe die entsprechende Code-Tabelle an.
b) Überführe die Buchstabenfolge „VIEL ERFOLG“ mittels der in Teilaufgabe a) generierten Codewörter in eine Huffman-codierte Bitfolge.
c) Wie viele Bits benötigst du, um „VIEL ERFOLG“ zu codieren?

Aufgabe 10

Gegeben sind folgende Buchstaben und ihre Häufigkeiten:

Zeichen	C	I	H	A	F	G	L	R	S	T
Häufigkeit	2	2	2	1	1	1	1	1	1	1

a) Generiere eine Huffman-Codierung für die Buchstaben und gib die entsprechende Code-Tabelle an.

b) Überführe die Buchstabenfolge „RICHTIG FALSCH“ mittels der in Teilaufgabe a) generierten Codewörter in eine Huffman-codierte Bitfolge.
c) Wie viele Bits benötigst du, um die Buchstabenfolge „RICHTIG FALSCH“ zu codieren?

Aufgabe 11

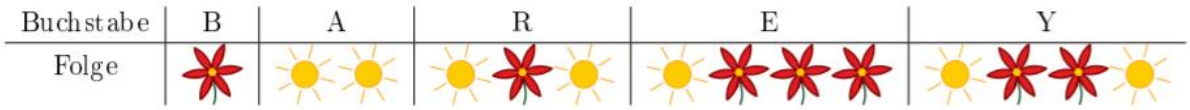
Gegeben sind folgende Buchstaben und ihre Häufigkeiten:

Zeichen	I	C	G	H	T	R	W
Häufigkeit	4	2	2	2	2	1	1

a) Generiere eine Huffman-Codierung für die Buchstaben und gib die entsprechende Code-Tabelle an.
b) Überführe die Buchstabenfolge „RICHTIG WICHTIG“ mittels der in Teilaufgabe a) generierten Codewörter in eine Huffman-codierte Bitfolge.
c) Wie viele Bits benötigst du, um die Buchstabenfolge „RICHTIG WICHTIG“ zu codieren?

Aufgabe 12 Blumen und Sonnen

Barbara hat 2 Stempel bekommen. Einer druckt eine Blume, der andere druckt eine Sonne. Sie überlegt, wie sie nur mit Blumen und Sonnen ihren Namen stempeln kann. Für verschiedene Buchstaben bestimmt sie verschiedene Folgen von Blumen und Sonnen:



Ihren eigenen Namen Barbara muss sie dann so stempeln:

Nun stempelt Barbara den Namen einer befreundeten Person:

Welchen Namen hat sie gestempelt?
A) Abby
B) Arya
C) Barry
D) Ray

Aufgabe 13 Zerteile den Code

In einem speziellen Code für Texte wird jeder Buchstabe durch ein Codewort aus den Ziffern 0 bis 9 kodiert. Die Besonderheit ist: Kein Codewort darf mit dem Codewort eines anderen Buchstaben beginnen. Ein Beispiel: Der Buchstabe X wird durch 12 kodiert. Nun kann Y durch 2 kodiert werden. Denn 12 beginnt nicht mit 2 (und 2 nicht mit 12). Jetzt kann Z durch 11 kodiert werden; denn weder 12 noch 2 beginnen mit 11 und 11 beginnt weder mit 12 noch mit 2. 21 wäre jedoch nicht als Codewort für Z erlaubt, weil es mit 2, also dem Codewort von Y beginnt.

Das Wort **BEBRAS** wird durch die nachfolgende Ziffernfolge kodiert. Teile die Ziffernfolge in die Codes der einzelnen Buchstaben!

1 2 1 1 2 2 3 3 3 2 1

Musterlösungen

Aufgabe 1

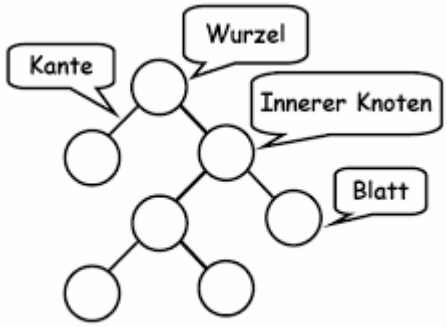
a) Nein, die Codierung ist nicht eindeutig, z.B. ist das Codewort für m der Anfang der Codewörter für p und s. Es gibt viele andere Wörter, die man noch aus der Bitfolge decodieren könnte. Zerteilt man die Bitfolge in einzelne Bits 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0, wäre das decodierte Wort „mimimiimimiimmmmi“, man könnte aber auch „sssissippi“ decodieren (10 10 10 0 10 10 0 11 11 0) oder „smisimisimmmmi“ (10 1 0 10 0 1 0 10 0 1 1 1 1 0) usw.

b) Ja, die Codierung ist eindeutig. Sie berücksichtigt aber nicht die Redundanz der Buchstaben s, i und p. Die Codierung kann verbessert werden, indem für häufig vorkommende Buchstaben kürzere Codewörter gewählt werden als für selten vorkommende Buchstaben.

c) Mississippi wird wie folgt codiert: 100 11 0 0 11 0 0 11 101 101 11. Die Codewörter sind eindeutig. Für die Codierung mit Codewörtern fester Länge wie in Aufgabe b) werden 33 Bits verwendet, für die Codierung mit verschieden langen Codewörtern werden 21 Bits verwendet.

Aufgabe 2

a)



b) Der Knoten ohne Vorgänger heisst Wurzel. Knoten mit Nachfolgern heissen innere Knoten. Knoten ohne Nachfolger heissen Blätter.

c) Ein binärer Baum ist ein Baum von Grad 2. Jeder Knoten im binären Baum hat höchstens zwei Nachfolger.

d) Den direkten Vorfahren nennt man auch Vater des Knotens. Die direkten Nachfolger eines Knotens werden auch Kinder des Knotens genannt.

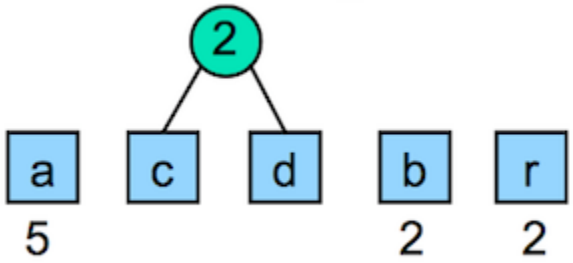
e) Ein binärer Baum ist entweder leer oder er besteht aus einer Wurzel w mit einem linken und einem rechten Teilbaum, bei welchen es sich wieder um binäre Teilbäume handelt.

Aufgabe 3

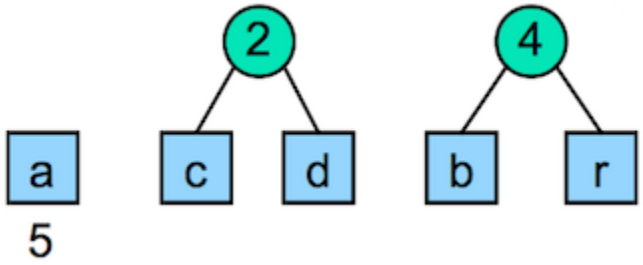
a) Aufbau des Huffman-Baums

I. Schritt 1

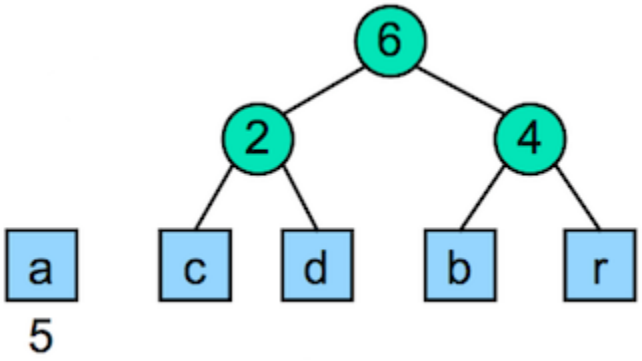
II. Schritt 2



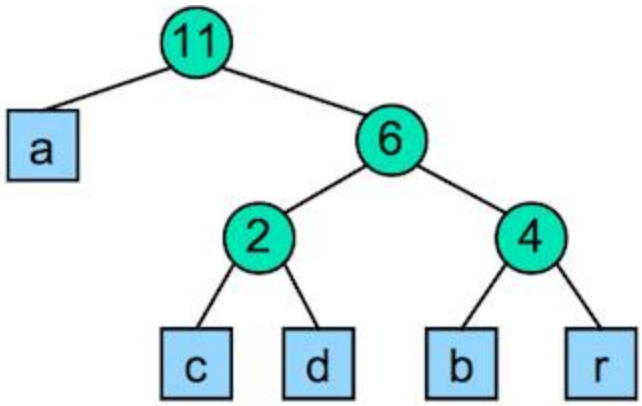
III. Schritt 3



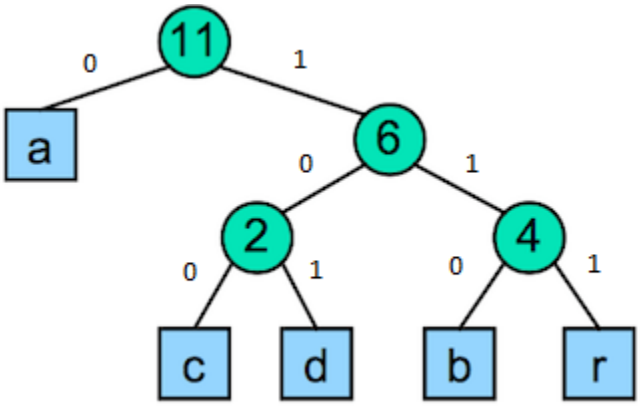
IV. Schritt 4



V. Schritt 5



VI. Schritt 6



b) Die Codewörter für d und c sind kürzer, aber das Codewort für r ist länger.

Buchstabe	Codewort
A	0
B	110
D	101
C	100
R	111

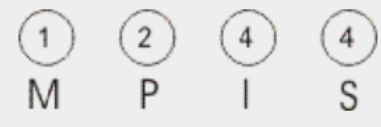
c) Die Codierung für „abracadabra“ ist 01101110100010101101110. Die Codierung benötigt 23 Bits wie auch die Codierung von „abracadabra“ mit den Codewörtern.

Buchstabe	Codewort
A	0
B	101
D	1001
C	1000
R	11

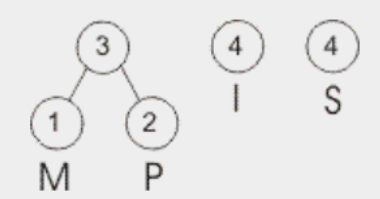
Aufgabe 4

a) Huffman-Baum

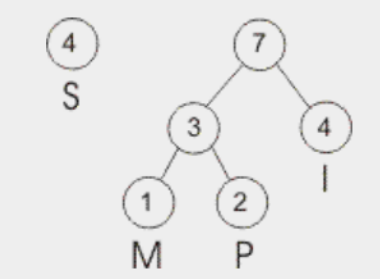
I. Schritt 1



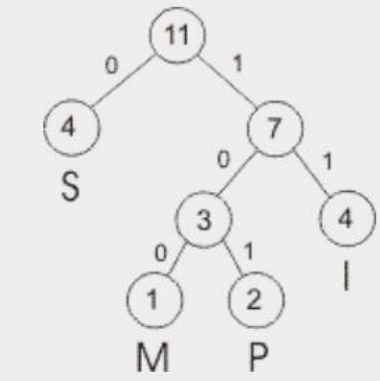
II. Schritt 2



III. Schritt 3



IV. Schritt 4



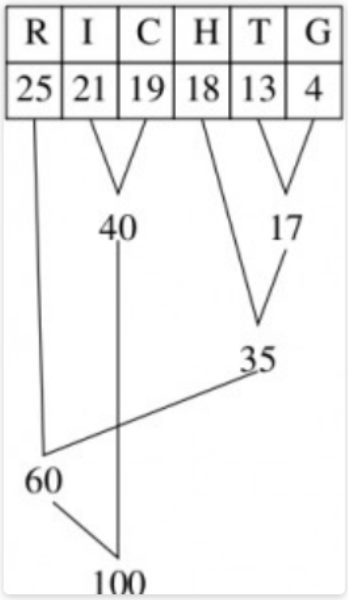
b) Codewort-Tabelle:

Buchstabe	Codewort
I	11
M	100
S	0
P	101

c) mississippi codiert: 100 11 0 0 11 0 0 11 101 101 11

Aufgabe 5

a)



R	00
I	10
C	11
H	010
T	0110
G	0111

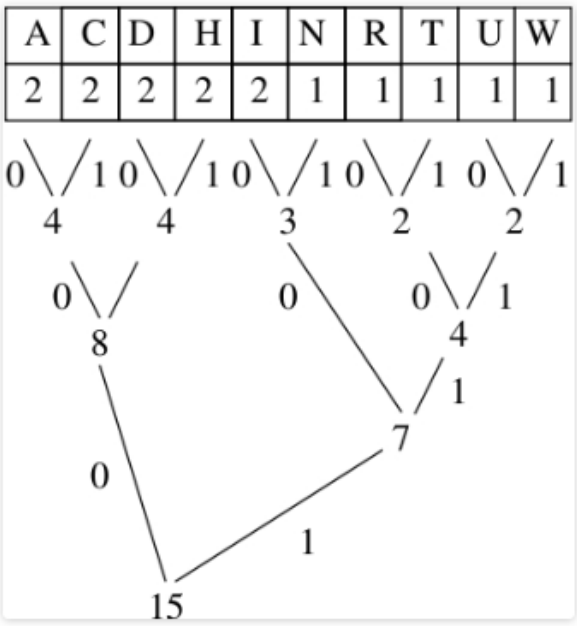
b)

00	10	11	010	0110	10	0111
R	I	C	H	T	I	G

Die gesuchte Nachricht lautet: „RICHTIG“

Aufgabe 6

a)



A	000
C	001
D	010
H	011
I	100
N	101
R	1100
T	1101
U	1110
W	1111

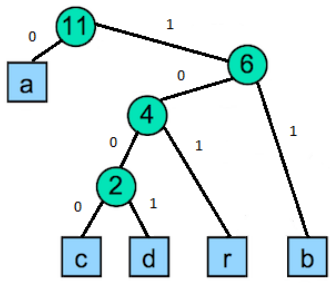
b)

I	C	H	W	A	R	D	A
100	001	011	1111	000	1100	010	000

D	U	N	I	C	H	T
010	1110	101	100	001	011	1101

Aufgabe 7

a)



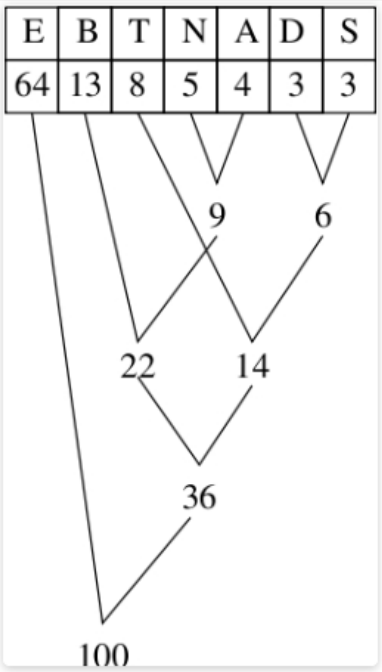
b)

Buchstabe	Codewort
A	0
B	11
D	1001
C	1000
R	101

c) 5/11 * 1 + 2/11 * 2 + 1/11 * 4 + 1/11 * 4 + 2/11 * 3 = 23/11 = 2.09 Bits pro Buchstabe. Die mittlere Codewortlänge ist gleich wie bei den anderen beiden Huffman-Bäumen für „abracadabra“.

Aufgabe 8

a)



E	0
B	100
T	110
N	1010
A	1011
D	1110
S	1111

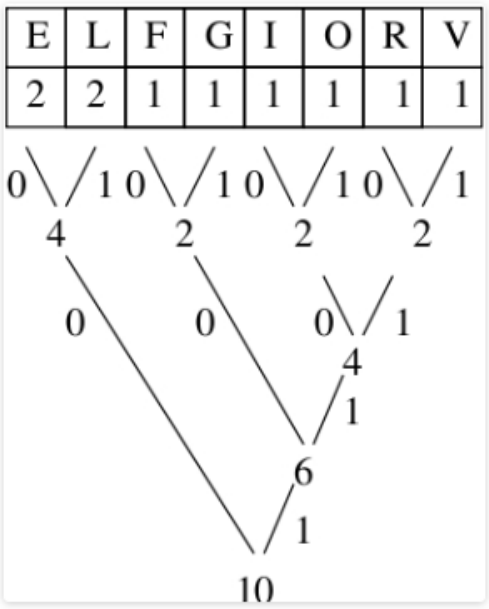
b)

100	0	1111	110	1011	1010	1110	0	1010
B	E	S	T	A	N	D	E	N

Die gesuchte Nachricht lautet: „BESTANDEN“

Aufgabe 9

a) Es gibt viele Möglichkeiten, den Huffman-Baum aufzubauen. Dies ist eine Möglichkeit:



E	00
L	01
F	100
G	101
I	1100
O	1101
R	1110
V	1111

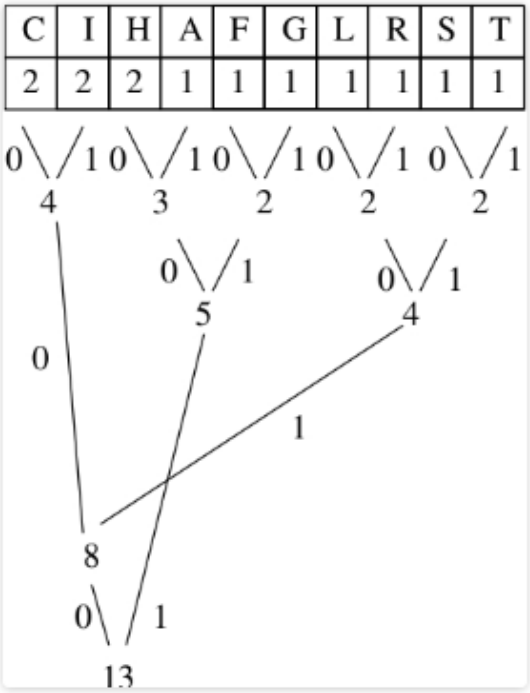
b)

V	I	E	L	E	R	F	O	L	G
1111	1100	00	01	00	1110	100	1101	01	101

c) Es werden 30 Bits benötigt.

Aufgabe 10

a) Es gibt viele Möglichkeiten, den Huffman-Baum aufzubauen. Dies ist eine Möglichkeit:



C	000
I	001
H	100
A	101
F	110
G	111
L	0100
R	0101
S	0110
T	0111

b)

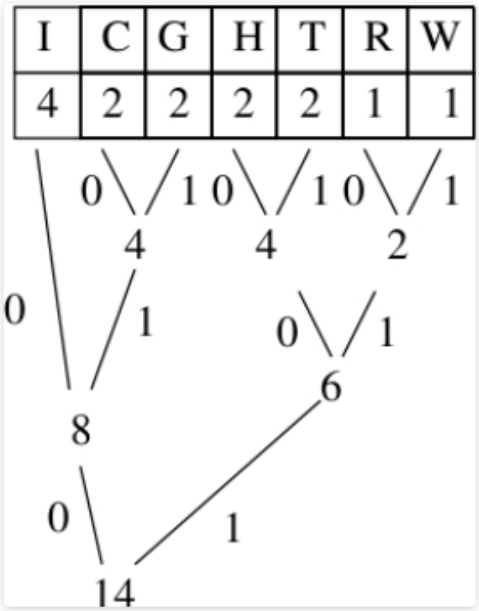
R	I	C	H	T	I	G
0101	001	000	100	0111	001	111

F	A	L	S	C	H
110	101	0100	0110	000	100

c) Es werden 43 Bits benötigt, um „RICHTIG FALSCH“ zu codieren.

Aufgabe 11

a)



I	00
C	010
G	011
H	100
T	101
R	110
W	111

b)

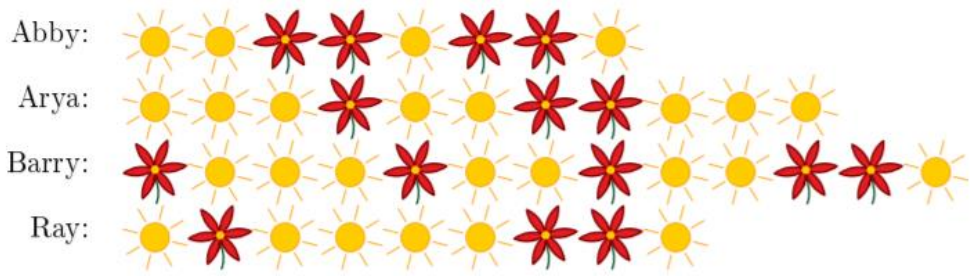
R	I	C	H	T	I	G
110	00	010	100	101	00	011

W	I	C	H	T	I	G
111	00	010	100	101	00	011

c) Es werden 46 Bits benötigt, um «RICHTIG WICHTIG» zu codieren.

Aufgabe 12 Blumen und Sonnen

Die richtige Antwort ist Abby. Die Namen von Barbaras Freunden haben folgende Codes:



Das Codieren von Daten kann auf verschiedene Weise geschehen. Zum Beispiel ist es oft üblich, dass die Zeichen, die auf der Tastatur eingegeben werden, in UTF-8, einer Variante von Unicode, abgespeichert werden. Dabei benötigen die häufigen Zeichen genau 1 Byte, das über 250 verschiedene Zeichen ermöglicht. Für seltenere Zeichen werden dann vier Byte Platz verwendet; damit ermöglicht man viele Millionen verschiedener Zeichen der verschiedensten Sprachen der Erde. Das System funktioniert schon sehr gut, aber auch bei den häufigen Zeichen werden einige viel häufiger verwendet als andere, beispielsweise das «E» oder das «N» im Vergleich zu «X» oder vom «Ö». Auch hierfür gibt es sinnvolle Codes, die dann mit gänzlich variabler Länge von Symbolen arbeiten. Bei solchen Codes mit variabler Länge ist es sinnvoll, dass ein Code eines Zeichens niemals der Anfang eines Codes eines anderen Zeichens ist. Dadurch kann die Bedeutung der einzelnen Codes schnell und einfach erkannt werden. Solche Codes nennt man Präfixcode. Ein bekannter Präfixcode ist der Morsecode. Wenn man nun einen möglichst platzsparenden Code haben will, dann

muss man wissen, wie häufig die einzelnen Zeichen vorkommen, und man kann mit der sogenannten Huffman-Codierung einen besonders platzsparenden Code berechnen. Jeder Huffman-Code ist auch ein Präfixcode.

Aufgabe 13 Zerteile den Code

Die Lösung ist 1 21 1 22 33 321.

Wir betrachten die Ziffernfolge von links. Falls B durch 12 kodiert würde, wäre die folgende 1 das Codewort für E. Direkt dahinter käme nämlich wieder 12 für das zweite B. Doch das ist gegen die Regel: Das Codewort für B würde mit dem für E beginnen. Längere Anfangsstücke der Ziffernfolge (z.B. 121, 1211 etc.) können auch nicht Codes für B sein, weil sie alle nur einmal in der Ziffernfolge enthalten sind. Folglich ist das Codewort für B die Ziffer 1. Nun muss das Codewort für E folgen. Die 2 alleine ist es nicht, denn dann müsste der Anfang der Ziffernfolge zu BEBB dekodiert werden. Das Codewort 21 hingegen ist richtig: Erstens wird der Anfang so zu BEB und zweitens ist ein längeres Codewort nicht möglich, da sonst nur noch die letzte 1 das zweite B sein kann – aber BEBRAS hört nicht mit B auf. Nun wissen wir, dass 1 21 1 die Kodierung für BEB ist. Wir machen mit dem Rest weiter: 2233321 ist die Kodierung für RAS. Die 2 alleine kann also nicht das Codewort für R sein, sonst hätten wir RR zu Beginn. Das Codewort für R muss also mindestens die 22 beinhalten. Am Ende wiederum sind 1 und 21 schon als Codes vergeben. Das Codewort für S muss also mindestens die Folge 321 sein. Zwischen 22 und 321 steht 33. Das muss das Codewort für A sein: Das einzig andere, noch mögliche Codewort wäre 3. Dann müsste 3321 das Codewort für S sein – und würde mit dem Codewort für A beginnen; das ist gegen die Regel. Die Aufteilung des hinteren Teils ist also 22 33 321.

Literaturverzeichnis

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein Introduction to Algorithms, Second Edition, The MIT Press, 2001, S.138-145 und S.385-388

Juraj Hromkovic, Tobias Kohn, einfach INFORMATIK 7-9 Programmieren Sekundarstufe I, Klett und Balmer Verlag, 2018

Juraj Hromkovic, einfach INFORMATIK 7-9 Strategien entwickeln Sekundarstufe I, Klett und Balmer Verlag, 2018

Hans-Joachim Böckenhauer, Juraj Hromkovic, Formale Sprachen, Springer Vieweg, 2013

Matthias Niklaus, Bäume in der Informatik, https://www.ethz.ch/content/dam/ethz/special-interest/dual/educeth-dam/documents/Unterrichtsmaterialien/informatik/B%C3%A4ume%20in%20der%20Informatik%20(Leitprogramm)/baeume.pdf, zuletzt besucht am 5.5.2019

Timur Erdag und Björn Steffen, Bäume in der Informatik, https://www.ethz.ch/content/dam/ethz/special-interest/dual/educeth-dam/documents/Unterrichtsmaterialien/informatik/Bin%C3%A4re%20Suchb%C3%A4ume/leitprogramm.pdf, zuletzt besucht am 5.5.2019

Ralph-Erich Hildebrandt, Konrad-Adenauer-Gymnasium Langenfeld,
Bäume – Allgemeines und Definition
http://www.informatikwelt.de/info12/kapitel23/baumallgemein.htm, zuletzt besucht am 5.5.2019

Dr. Jochen Ziegenbalg, PH Karlsruhe,
http://www.ziegenbalg.ph-karlsruhe.de/materialien-homepage-jzbg/cc-interaktiv/huffman/codierung.htm, zuletzt besucht am 5.5.2019

Christoph Oehler, Raimond Reichert,
https://www.swisseduc.ch/informatik/daten/huffmann_kompression/, zuletzt besucht am 5.5.2019

H.W. Lang, Hochschule Flensburg
http://www.iti.fh-flensburg.de/lang/algorithmen/code/huffman/huffman.htm,
zuletzt besucht am 5.5.2019

Ann-Kathrin Kaptain, ERG Saalfeld,
http://www.erasmus-reinhold-gymnasium.de/info/angew_inf/huffmann.html, zuletzt besucht am 5.5.2019

https://filestore.aqa.org.uk/resources/computing/AQA-8520-TG-HC.PDF, zuletzt besucht am 5.5.2019

https://beckassets.blob.core.windows.net/product/readingsample/565846/9781848000711_excerpt_001.pdf, zuletzt besucht am 23.6.2019

Aufgaben aus

http://parcog.com/huffman-aufgaben-und-losungen/, zuletzt besucht am 5.5.2019

https://informatik-biber.ch/documents/2016/Informatik-Biber-2016-Schuljahre7-8-mitLoesungen.pdf, zuletzt besucht am 5.5.2019

https://informatik-biber.ch/documents/2017/Informatik-Biber-2017-Alle-Stufen-mitLoesungen.pdf, zuletzt besucht am 5.5.2019

Bilder aus

Alice und Bob:
http://rmg.zum.de/wiki/Benutzer:Deininger_Matthias/Facharbeit/Alice_Bob_und_Mallory, zuletzt besucht am 5.5.2019

David Huffman:
https://www.maa.org/press/periodicals/convergence/discovery-of-huffman-codes, zuletzt besucht am 22.6.2019