Kantonsschule Im Lee

Informatik

Kompression

Skript

Cyril Wendl

Inhaltsverzeichnis

Kapitel 1

Kompression

1.1 Einführung

1.1.1 Was ist Kompression?

Der Begriff „Kompression“ stammt — wie viele andere Begriffe — ursprünglich aus dem Latenischen (*compressio* = das Zusammendrücken, das Zusammenpressung, die Umarmung). Mit Kompression ist also häufig die „Verkleinerung“, bzw. das „Zusammendrücken“ von Daten gemeint, meist aus einem der folgenden Gründen:

1. **Speicherplatz optimieren**: Angesichts des stets wachsenden Speicherbedarfs von Datenzentren sowie privaten Geräten wird es immer wichtiger, Dokumente zu „komprimieren“, d.h., deren Grösse zu verringern, falls möglich (siehe Abbildung 1.1).

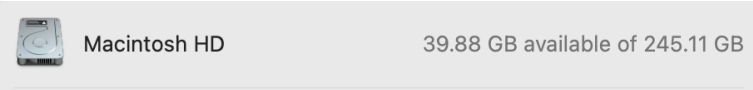


Abbildung 1.1: Beispiel einer Datei sowie deren Dateigrösse, unkomprimiert (original) und komprimiert

2. **Speicherplatz optimieren**: Häufig kann mit wenig Effort Speicherplatz optimiert werden, womit der Platzbedarf auf Speichermedien wie Festplatten verringert werden kann. Da Festplatten — ähnlich einem Reisekoffer — Kosten verursachen, will man den Speicherbedarf möglichst minimal halten (siehe Abbildung 1.2).

Abbildung 1.2: Speicherplatz optimieren

3. **Wartezeit verkürzen**: Computer und Geräte kommunizieren in Netzwerken, innerhalb welchen oftmals grosse Datenmengen übertragen werden müssen. Die Verbindungen zwischen den Geräten haben jedoch nur eine begrenzte Kapazität und werden daher häufig als ***Flaschenhälse*** bezeichnet.

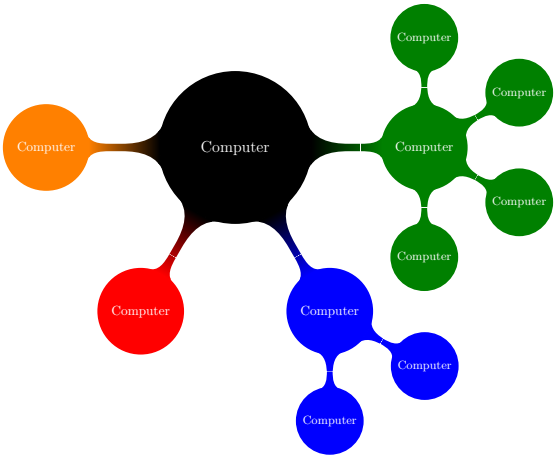


Abbildung 1.3: Visualisierung von Computernetzwerken und deren *Flaschenhälsen*

Historischer Kontext

- Leder und Stein: Teure Materialien
- Römische Zahlendarstellung: z.B. Zahl **999**

`CCCCCCCCCXXXXXXXXXIIIIIIIII`

↓

Erfindung der Ziffern D (500), L (50), V(5)

↓

`DCCCCLXXXXVIIII`

↓

Effizientere Schreibweise

↓

IM

1.2 Verlustfreie vs. verlustbehaftete Kompression

Einige Beispiele:

(a)

(b)

Abbildung 1.4: `LeeLogo.jpg (48 KB, 300 dots per inch)`

Anwendung	Verlustfrei	Verlustbehaftet
Audio	`.flac, .aac`	`.mp3`
Dateien	`.zip, .rar`	-
Video	`.raw`	`.mp4, .avi, .mov`
Fotografie	`.raw, .psd, .gif`	`.jpg, .png`
Grafik	`.svg, .pdf`	

1.3 Kompressionsalgorithmen

Aufgabe 1.1

Was könnte folgende Sequenz bedeuten?

WENN 😊 HINTER 😊 △, △ EINIGE 😊 ANDEREN 😊 NACH.

Lösungsvorschlag zu Aufgabe 1.1

Viele Bedeutungen sind möglich, zum Beispiel:

WENN GRIECHEN HINTER GRIECHEN RENNEN, RENNEN EINIGE GRIECHEN ANDEREN GRIECHEN NACH.

Wie können wir folgenden Text effizient kodieren?

AACABAAACBACAACADAAD

Da wir hier nur vier Buchstaben (A, B, C und D) haben, reichen 2 Bits für die Kodierung aus, da 2 Bits $2^2 = 4$ Möglichkeiten zur Kodierung bieten.

Buchstabe	**A**	**B**	**C**	**D**
Häufigkeit	12	2	4	2
Relative Häufigkeit	60%	10%	20%	10%
„Naive“ Kodierung:	00	01	10	11
„Effiziente“ Kodierung:	0	110	10	111

- „Naive“ Codierung (40 Zeichen):
  `0000100001000000100100100000100011000011`
- „Effiziente“ Codierung (32 Zeichen, -20% Ersparnis):
  `00101100001011001000100111001111`

Die „effiziente“ Kodierung verwendet in diesem Fall 20% weniger Speicherplatz als die „naive“ Kodierung.

Beide Codierungen sind **präfixfrei**: Kein Codewort ist Anfangsstück (Präfix) eines anderen Codeworts. Dies ist notwendig, damit es keine Verwechslungsgefahr beim Interpretieren des Codes gibt.

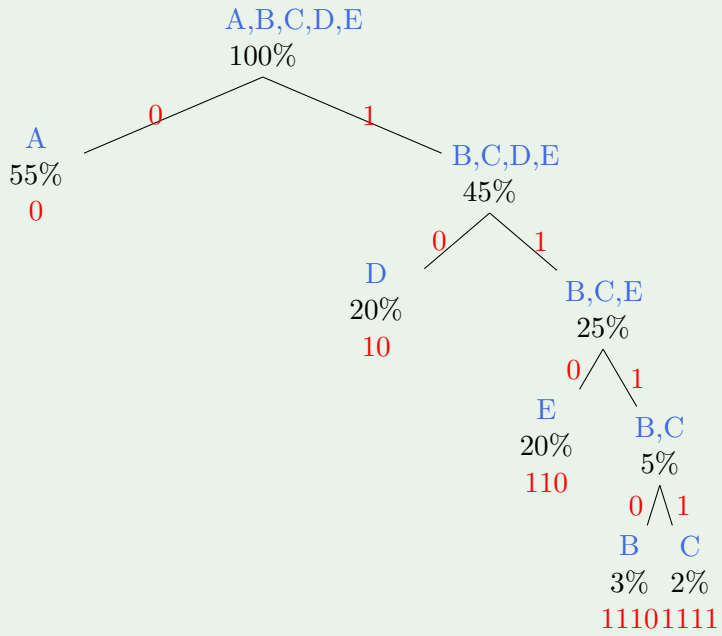
Aufgabe 1.2

Wir haben einen Text mit 100 Buchstaben und den Buchstaben A,B,C,D, sowie E. Im Text kommt A 55-mal, D und E jeweils 20-mal, B 3-mal und C 2-mal vor. Wählen Sie eine präfixfreie Kodierung der 5 Buchstaben des Alphabets, sodass die resultierende binäre Darstellung des Textes so kurz wie möglich wird.

1. Visualisieren Sie den Prozess bei der Wahl der Kodierungen der einzelnen Buchstaben mit einem Baumdiagramm.
2. Wie lange ist die binäre Darstellung des Texts mit Ihrer Kodierung?

✔ Lösungsvorschlag zu Aufgabe 1.2

Eine mögliche Lösung:

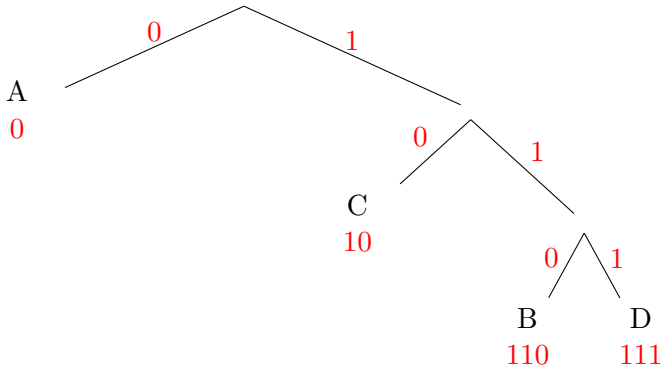


Die Länge des gesamten Texts berechnet sich aus der Summe der Häufigkeiten der Buchstaben mal deren Kodierungslänge:

Buchstabe	Kodierung	Länge der Kodierung
A	0	1
B	1110	4
C	1111	4
D	10	2
E	110	3

$$55 \times 1 + 3 \times 4 + 2 \times 4 + 20 \times 2 + 20 \times 3 = 175$$

Buchstabe	**A**	**B**	**C**	**D**
Relative Häufigkeit	60%	10%	20%	10%
„Effiziente“ Kodierung:	0	110	10	111

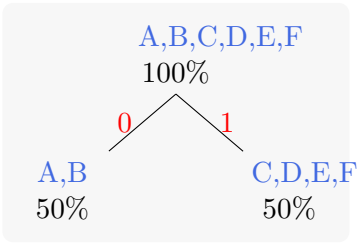


1.3.1 Top-Down-Ansatz: Maximale Balance

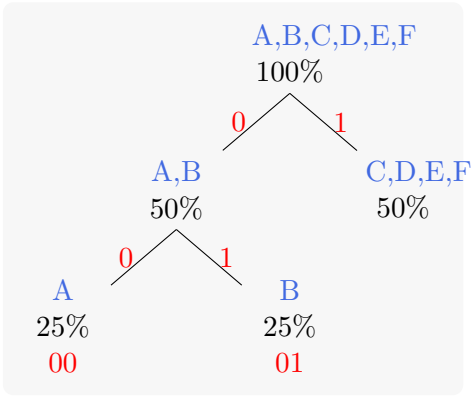
Idee: Summe der Buchstabenhäufigkeiten im linken und rechten Teilbaum ausbalancieren

Buchstabe	**A**	**B**	**C**	**D**	**E**	**F**
Relative Häufigkeit	25%	25%	13%	13%	12%	12%

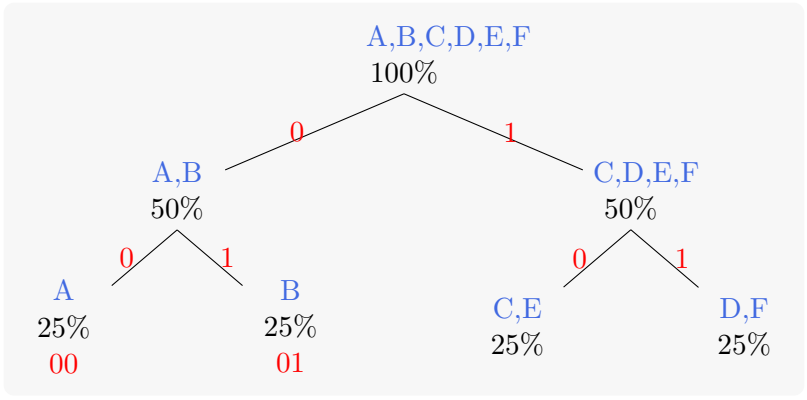
Tabelle 1.1: Beispiel-Häufigkeiten von Buchstaben



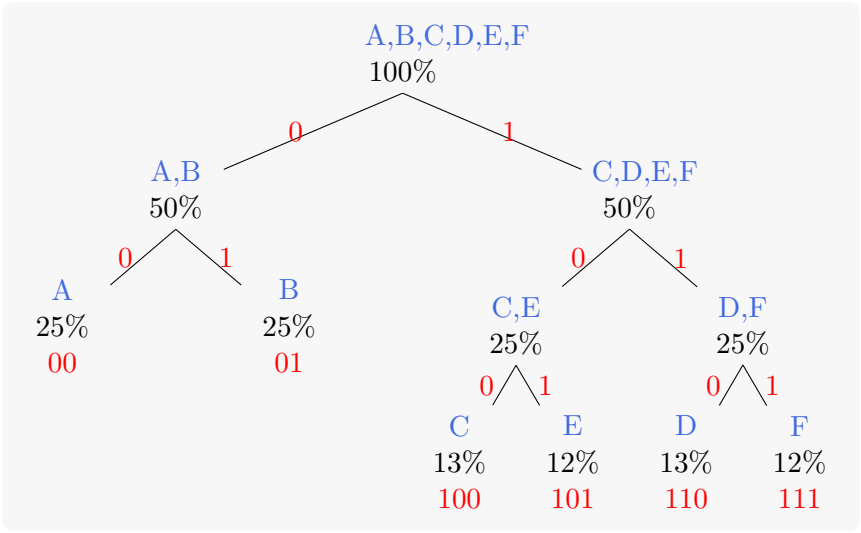
(a) Schritt 1



(b) Schritt 2



(c) Schritt 3



(d) Schritt 4 (Kodierung)

Abbildung 1.5: Schritt-für-Schritt-Erstellung des Baumdiagramms für Tabelle 1.1 mit dem ***Top-Down***-Ansatz.

Aufgabe 1.3

Nutzen Sie die Strategie der maximalen Balance, um die präfixfreien Kodierungen für Texte mit folgenden Buchstabenhäufigkeiten zu entwerfen.

1. A - 57%, B - 23%, C - 12%, D - 8%
2. A - 25%, B - 20 %, D - 14 %, E - 11 %, F - 10 %, G - 10 %, H - 5%, I - 5%
3. A - 33 %, B - 30 %, C - 17 %, D - 14 %, E - 6%

Lösungsvorschlag zu Aufgabe 1.3

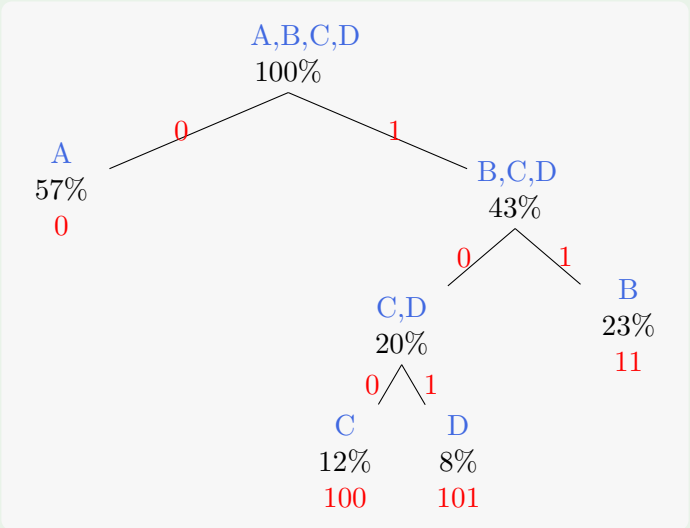


Abbildung 1.6: Lösungsvorschlag für Teilaufgabe 1

✔ Lösungsvorschlag zu Aufgabe 1.3

Für Teilaufgabe 2 gibt es mehrere Lösungsmöglichkeiten, z.B.:

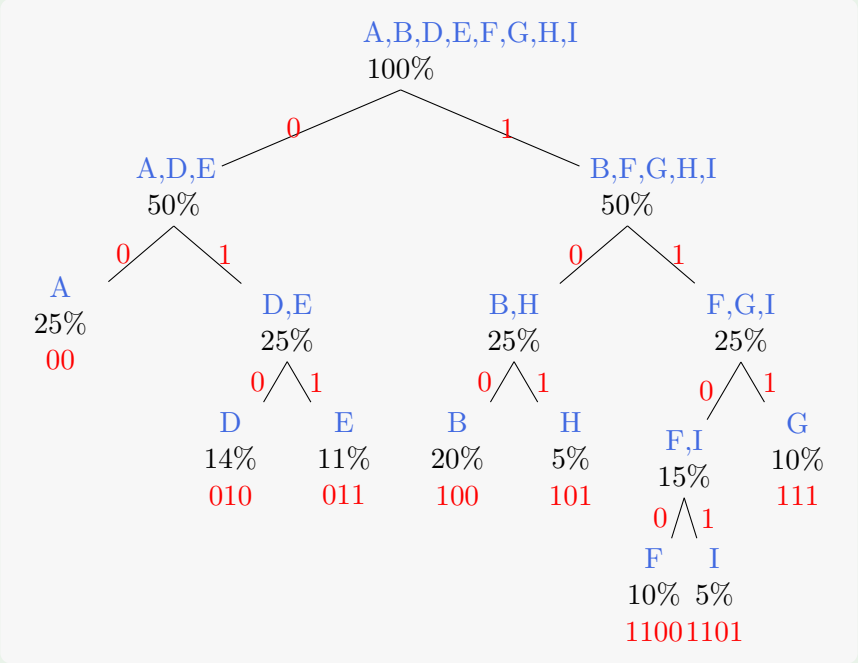


Abbildung 1.7: Lösungsvorschlag für Teilaufgabe 2

✔ Lösungsvorschlag zu Aufgabe 1.3

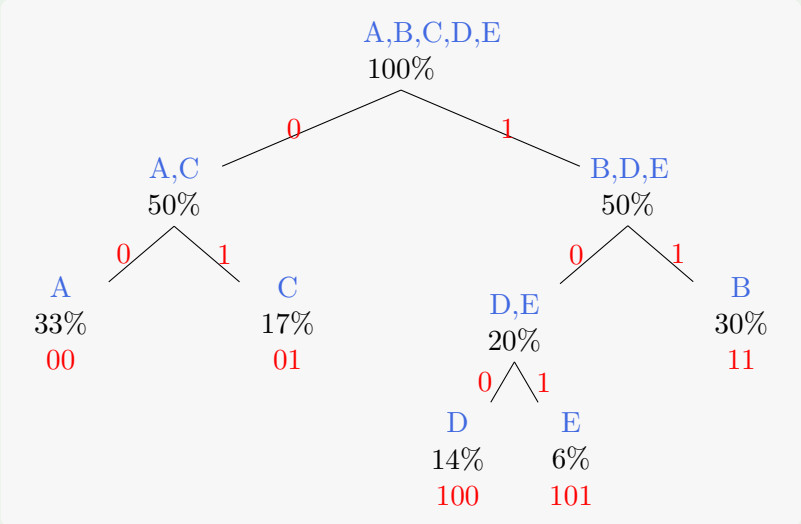


Abbildung 1.8: Lösungsvorschlag für Teilaufgabe 3

Aufgabe (Challenge) 1.4

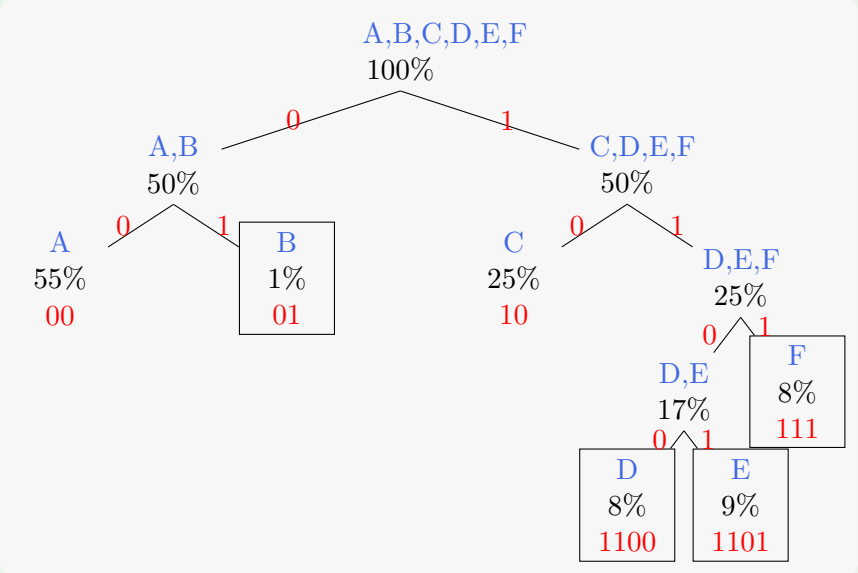
Schlagen Sie eine Häufigkeitsverteilung für Buchstaben so vor, dass die Strategie der maximalen Balance eindeutig ein Baumdiagramm bestimmt, in dem ein 10-mal häufigerer Buchstabe eine längere Kodierung hat als ein 10-mal seltener vorkommender Buchstabe.

Aufgabe 1.5

Problem: Was passiert bei folgender Verteilung?

Buchstabe	**A**	**B**	**C**	**D**	**E**	**F**
Relative Häufigkeit	55%	1%	25%	8%	9%	8%

Lösungsvorschlag zu Aufgabe 1.5



Wie wir sehen, wird der resultierende Baum nicht ausgeglichen: während der Buchstabe „F“ (8% Häufigkeit im Originaltext) beispielsweise mit „111“ kodiert wird, wird der Buchstabe „B“ (1 % Häufigkeit im Originaltext) mit „01“ kodiert, also mit einer kürzeren Bitfolge. Wie wir allerdings gesehen haben, sollten Buchstaben mit höherer Häufigkeit kürzer kodiert werden, da wir sie ja häufiger kodieren müssen und mit einer kürzeren Kodierungslänge somit insgesamt mehr Speicherplatz sparen.

1.3.2 Bottom-Up-Ansatz: Huffmann-Kodierung

Idee:

- Jeder Buchstabe wird als Graph mit einem Knoten (=Baum mit einem Blatt) betrachtet. Die Häufigkeit jedes Buchstabens wird neben dem Blatt notiert. Wir wollen nun die Buchstaben zu Teilbäumen verbinden, bis wir einen einzigen Baum haben.

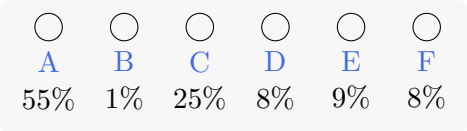


Abbildung 1.9: Beispiel-Häufigkeiten von Buchstaben

- In einer Schleife: Wir schauen alle isolierten Blätter und Wurzeln von Teilbäumen an. Diejenigen Knoten, die die kleinste (gemeinsame) Prozentzahl enthalten werden miteinander verbunden. Die neue Wurzel wird markiert mit dem Prozentsatz beider Knoten.

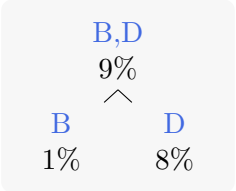
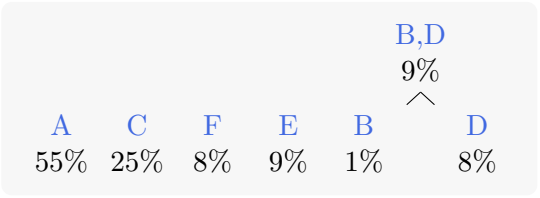


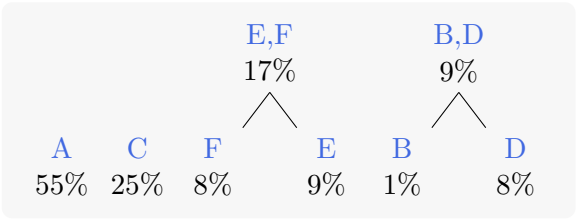
Abbildung 1.10: Erster Schritt im Huffmann-Kodierungs-Algorithmus

- Falls mehrere Möglichkeiten mit gleicher relativer Häufigkeit existieren, nehmen wir den Teilbaum mit der geringsten Tiefe.

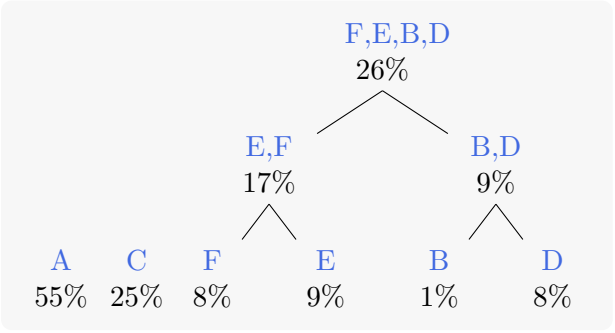
Welche Knoten werden als nächstes miteinander verbunden?



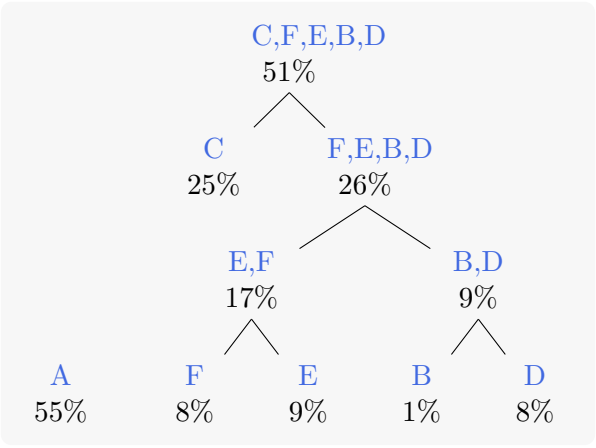
(a) Schritt 1



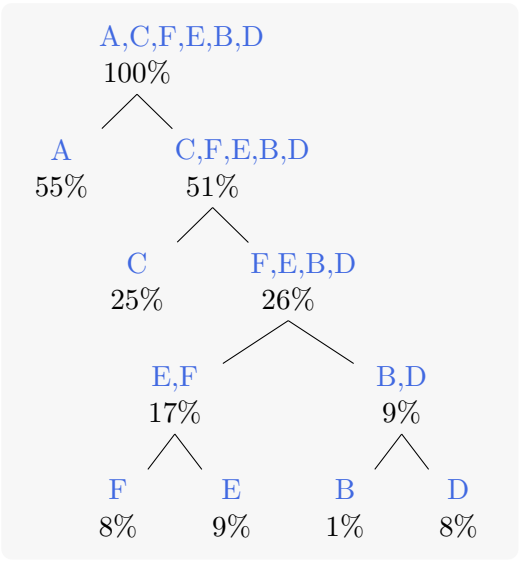
(b) Schritt 2



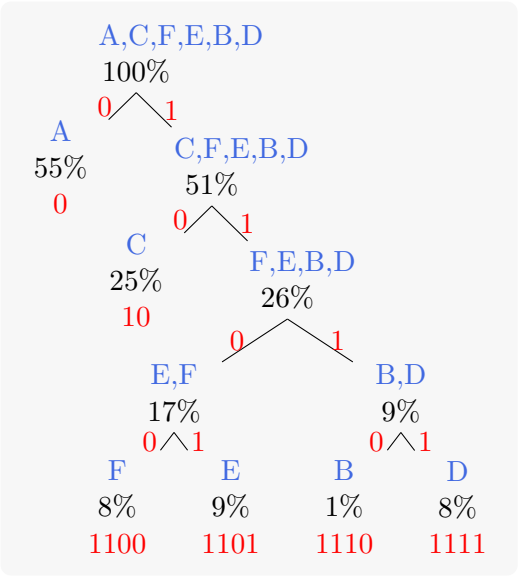
(c) Schritt 3



(d) Schritt 4



(e) Schritt 5



(f) Schritt 6 (Kodierung)

Abbildung 1.11: Schritt-für-Schritt-Erstellung des Baumdiagramms für Abbildung 1.9 mit der ***Huffmann***-Kodierung

Aufgabe 1.6

Weshalb haben wir in Schritt 2 einen Baum {E,F} statt {F,B,D} gemacht?

Als Fazit kann folgendes festgehalten werden:

- Strategie der **Maximale Balance**: *top-down*-Ansatz (von Wurzel zu Blättern), kann zu suboptimalen Kodierungen führen

- **Huffman-Kodierung**: *bottom-up*-Ansatz (von Blättern zur Wurzel), garantiert bestmögliche Kodierungslänge für alle Buchstaben

Aufgabe 1.7

Verwenden Sie den Algorithmus von Huffman, um präfixfreie Kodierungen für die folgenden Häufigkeitsverteilungen von Buchstaben zu konstruieren.

- A - 25%, B - 25 %, C - 13 %, D - 12%, E - 6%, F - 7%, G - 6%, H - 6%
- A - 15%, B - 15 %, C - 15 %, D - 15 %, E - 15 %, F - 15 %, G - 10 %
- A - 45%, B - 15 %, C - 7%, D - 7%, E - 6 %, F - 10%, G - 10%
- A - 33%, B - 18 %, C - 18 %, D - 17%, E - 4 % , F - 4 % , G - 3 %, H - 3 %

Vergleichen Sie in der Klasse die entstandenen präfixfreien Kodierungen. Die Lösungen zur jeweiligen Häufigkeitsverteilung dürfen sich unterscheiden. Wo liegen die Unterschiede und was haben die unterschiedlichen Lösungen immer gemeinsam?

Bei welchen der vier Häufigkeitsverteilungen der Buchstaben werden Sie mit der Strategie der maximalen Balance gleich gute präfixfreie Kodierungen der Buchstaben erhalten wie mit der Huffman-Strategie und bei welchen nicht?

Aufgabe 1.8

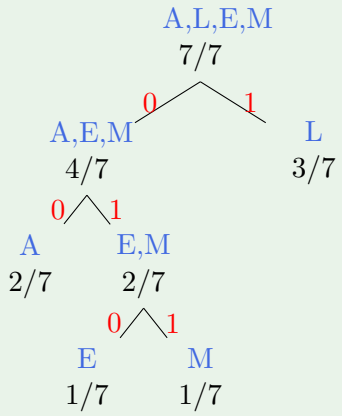
Komprimieren Sie das Wort „`ALLEMAL`“ mit der Huffman-Kodierung. Wie lange ist der Text komprimiert? Wie lange wäre er ohne Kompression?

✔ Lösungsvorschlag zu Aufgabe 1.8

Wir beginnen damit, die Wortlänge zu zählen und die relativen Buchstabenhäufigkeiten zu bestimmen:

- Wortlänge: 7
- Buchstaben-Häufigkeiten: `A` - 2/7, `L` - 3/7, `E` - 1/7, `M` - 1/7

Nun können wir den Baum erstellen. Das Endergebnis könnte wie folgt aussehen:



Daraus ergibt sich folgende Kodierung für den Text `ALLEMAL`:

- `A: 00`
- `L: 1`
- `E: 010`
- `M: 011`

Der gesamte Text würde also wie folgt kodiert: `0011010011001` (13 Zeichen).

Ohne Kompression würden wir 2 Bits pro Buchstaben benötigen, da wir insgesamt 4 verschieden Buchstaben (`A`, `L`, `E`, `M`) im Text haben und 2 Bits geben uns $2^2 = 4$ Möglichkeiten. Eine mögliche Kodierung wäre:

- `A: 00`
- `L: 01`
- `E: 11`
- `M: 10`

In diesem Fall würden wir 14 Bits für den Text benötigen. In diesem Beispiel haben wir durch die Kompression also nur ein Bit gespart. In anderen Fällen kann es jedoch viel mehr sein!

Aufgabe 1.9

Klicken Sie auf diesen Link, um Huffman-Bäume Schritt-für-Schritt zu visualisieren.

1.4 Arithmetische Kompression

Die Huffman-Kodierung führt zu einer optimalen Kompression einzelner Zeichen in vielen Fällen – insbesondere, wenn die Buchstabenhäufigkeiten bei 1/2, 1/4, 1/8 etc. liegen. Wenn jedoch eine Häufigkeit z.B. bei 30% liegt, sind 2 Bits (optimal für 25%) verschwenderisch! Jorma Rissanen hat daher eine Methode entwickelt, um ganze Texte anstatt einzelne Buchstaben zu kodieren. Die Grundidee dahinter ist folgende: Jedem Text wird eine Zahl zwischen 0 und 1 zugeordnet. Je länger der Text, desto mehr Nachkommastellen besitzt er. Dies führt unter gewissen Umständen zu kürzeren Texten als bei Huffman.

Beispiel 1.1:
Angenommen, wir haben einen längeren Text, der nur aus den vier Buchstaben `A, B, C` und `D` besteht und folgende Buchstabenhäufigkeiten hat:

Buchstabe	A	B	C	D
Häufigkeit	50%	20%	20%	10%

In diesem Fall könnte das Wort `ADBA`, welches Teil des längeren Texts ist, mit einer Kommazahl kodiert werden, wie in Abbildung 1.12 gezeigt.

Das Intervall $[0, 1)$ (von und mit 0 bis und ohne 1) wird in vier Gebiete unterteilt, deren Grösse der relativen Häufigkeit der Zeichen entspricht. Da wir das Wort `ADBA` kodieren wollen, unterteilen wir das Intervall in ein weiteres Teilintervall für Wörter der Länge 2, beginnend mit `A`. Hier verwenden wir wieder dieselben Buchstabenhäufigkeiten.

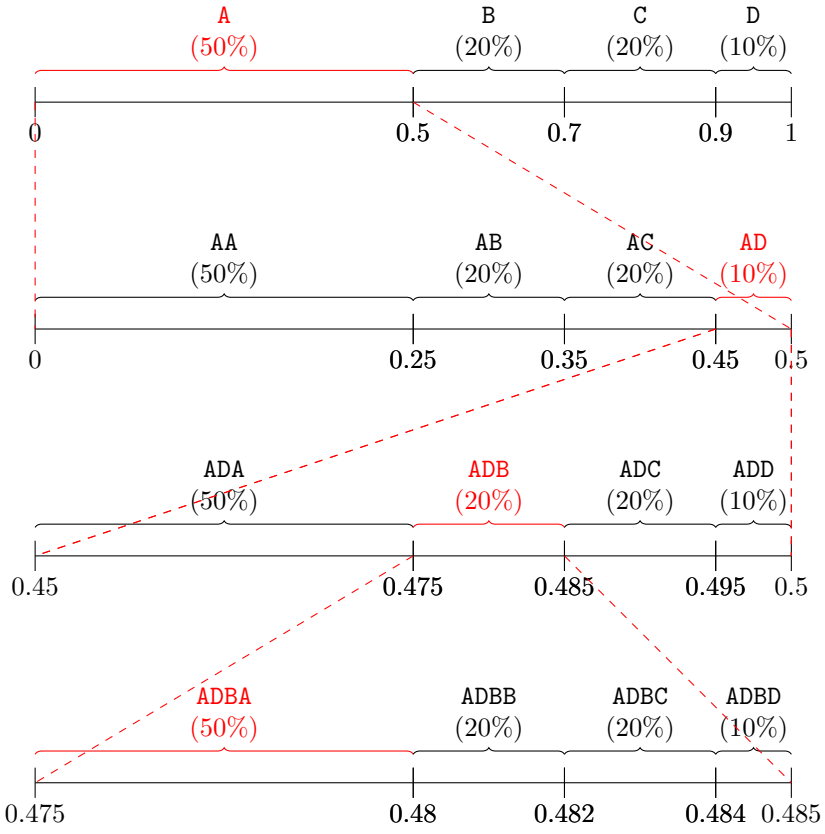


Abbildung 1.12: Arithmetische Kodierung des Worts `ADBA`

Der Text „`ADBA`“ wird also durch das Intervall $[0.475, 0.48[$ kodiert.

Aufgabe 1.10

Bestimmen Sie die arithmetische Kodierung der Wörter `BAD`, `BADAC` und `ACAB` basierend auf folgenden Häufigkeiten:

Buchstabe	A	B	C	D
Häufigkeit	50%	25%	12.5%	12.5%

Aufgabe 1.11

Dekomprimieren Sie die folgenden arithmetischen Kodierungen, wenn die Buchstabenhäufigkeitsverteilung wie folgt aussieht: A - 40 %, B - 25 %, C - 25 %, D - 10 %.

- 0.155, Textlänge 4
- $[0.104, 0.144)$
- $[0.2, 0.21)$
- 0.2595, Textlänge 5

Lösungsvorschlag zu Aufgabe 1.11

- AADC, Intervall: [0.1544, 0.1584)
- AAC
- ABBA
- ABDDB, Intervall: [0.2594, 0.25965)

Aufgabe (Challenge) 1.12

Lösen Sie die Challenge-Aufgaben auf Moodle zur Kompression!

Anhang A

Lernziele Kompression

- ☐ Ich kann Beispiele aus der Geschichte angeben, die aufzeigen, dass Kompression schon früh ein Bedürfnis war.
- ☐ Ich kann den Unterschied zwischen verlustbehafteten und verlustfreien Komprimierungen erklären und Beispiele dafür angeben.
- ☐ Ich weiss, was eine präfixfreie Kodierung ist, und wieso diese Eigenschaft wichtig ist.
- ☐ Ich kann mit einem Baumdiagramm eine präfixfreie Kodierung finden.
- ☐ Ich kann die absoluten und relativen Häufigkeiten von Zeichen in einem (kurzen) Text bestimmen.
- ☐ Ich kann eine präfixfreie Kodierung mit der Methode der maximalen Balance für einen Text finden, von dem die relativen Häufigkeiten der Zeichen gegeben ist.
- ☐ Ich kann ein Beispiel für eine relative Häufigkeit von Buchstaben angeben, bei welchem die Kodierung der maximalen Balance einen selteneren Buchstaben kürzer kodiert als einen häufigeren.
- ☐ Ich kann eine Huffman-Kodierung für einen Text finden, von dem die relativen Häufigkeiten der Zeichen gegeben ist.
- ☐ Ich kann ausgehend von der Kodierung und der absoluten Häufigkeiten der vorkommenden Zeichen berechnen, wie viele Bits für die Speicherung eines Textes nötig sind.
- ☐ Ich kann bei gegebener relativer Häufigkeit der Zeichen einen (kurzen) Text arithmetisch Kodieren.
- ☐ Ich kann bei gegebener relativer Häufigkeit der Zeichen eine arithmetische Kodierung dekomprimieren.