

## 6 Information und Komplexität

### 6.1 Parameter von Binärbäumen

#### 6.1.1 Einfache Parameter

Zur Erinnerung: die Menge  $\mathcal{B}_n$  der Binärbäume mit  $n$  inneren Knoten ist definiert durch

$$\begin{aligned}\mathcal{B}_0 &= \{\square\} \\ \mathcal{B}_{n+1} &\leftrightarrow \mathcal{B}_0 \times \mathcal{B}_n \uplus \mathcal{B}_1 \times \mathcal{B}_{n-1} \uplus \mathcal{B}_2 \times \mathcal{B}_{n-2} \uplus \cdots \uplus \mathcal{B}_n \times \mathcal{B}_0 \\ &= \sum_{0 \leq k \leq n} \mathcal{B}_k \times \mathcal{B}_{n-k}\end{aligned}$$

Die Menge der Binärbäume ist  $\mathcal{B} = \uplus_{n \geq 0} \mathcal{B}_n$ .

**Definition 21.** Für Binärbäume  $t \in \mathcal{B}$  wird definiert:

- die Menge  $I(t)$  der *inneren Knoten* (internal) durch<sup>9</sup>:

$$I(t) = \begin{cases} \emptyset & \text{für } t = \square \\ \{\circ_t\} \cup I'(t_\ell) \cup I'(t_r) & \text{für } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

$i(t) = \sharp I(t)$  bezeichnet die Anzahl der inneren (“internal”) Knoten von  $t$ .

- die Menge  $E(t)$  der *äusseren Knoten* (Blätter) (external) durch:

$$E(t) = \begin{cases} \{\square\} & \text{für } t = \square \\ E'(t_\ell) \cup E'(t_r) & \text{für } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

$e(t) = \sharp E(t)$  bezeichnet die Anzahl der äusseren (“external”) Knoten von  $t$ .

- $s(t) = i(t) + e(t)$  bezeichnet die *Grösse* (size) (Anzahl der Knoten) von  $t$ .
- Für einen Knoten  $a \in I(t) \cup E(t)$  ist seine Höhe  $h(a, t)$  der Abstand von  $a$  zur Wurzel von  $t$ .
- Die *Höhe*  $h(t)$  ist die maximale Höhe eines Blattes:  $h(t) = \max\{h(b, t); b \in E(t)\}$ .

<sup>9</sup>Die Bezeichnungen  $I'(t)$  und  $E'(t)$  soll darauf hindeuten, dass die inneren bzw. äusseren Knoten eines Binärbaumes  $t$  mit  $t$  indiziert sind, also unterscheidbare Objekte sind.

**Lemma 53.** 1. Für die Anzahl  $i(t) = \#I(t)$  der inneren Knoten von  $t$  gilt :

$$i(t) = \begin{cases} 0 & \text{falls } t = \square \\ 1 + i(t_\ell) + i(t_r) & \text{falls } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

2. Für die Anzahl  $e(t) = \#E(t)$  der äusseren Knoten von  $t$  gilt :

$$e(t) = \begin{cases} 1 & \text{falls } t = \square \\ e(t_\ell) + e(t_r) & \text{falls } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

3. Für die Grösse  $s(t) = i(t) + e(t)$  von  $t \in \mathcal{B}$  gilt:

$$s(t) = \begin{cases} 1 & \text{falls } t = \square \\ 1 + s(t_\ell) + s(t_r) & \text{falls } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

4. Für die Höhe  $h(a, t)$  eines Knotens  $a \in I(t) \cup E(t)$  gilt:

$$h(\square, \square) = 0$$

$$h(a, \langle \circ, t_\ell, t_r \rangle) = \begin{cases} 0 & a = \circ \\ h(a, t_\ell) + 1 & a \in E(t_\ell) \cup I(t_\ell) \\ h(a, t_r) + 1 & a \in E(t_r) \cup I(t_r) \end{cases}$$

5. Für die Höhe  $h(t)$  von  $t \in \mathcal{B}$  gilt:

$$h(t) = \begin{cases} 0 & \text{falls } t = \square \\ 1 + \max\{h(t_\ell), h(t_r)\} & \text{falls } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

*Beispiel 10.* Beispiel eines Binärbaums mit  $i(t) = 8, e(t) = 9, s(t) = 17, h(t) = 4$  in Abbildung 22.

**Lemma 54.** Für alle binären Bäume  $t$  gelten folgende Aussagen:

1.  $e(t) = i(t) + 1$ , und damit  $s(t) = 2 \cdot i(t) + 1 = 2 \cdot e(t) - 1$
2.  $h(t) \leq i(t)$
3.  $e(t) \leq 2^{h(t)}$ , also  $\log e(t) \leq h(t)$

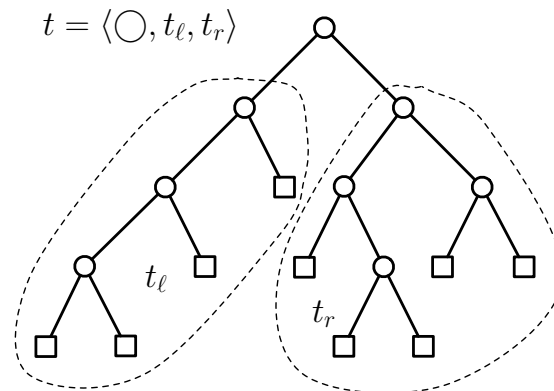


Abbildung 22: Einfache Parameter

*Beweis.* 1.

$$\begin{aligned}
 e(\square) - i(\square) &= 1 - 0 = 1 \\
 e(\langle \bigcirc, t_\ell, t_r \rangle) - i(\langle \bigcirc, t_\ell, t_r \rangle) &= e(t_\ell) + e(t_r) - (i(t_\ell) + i(t_r) + 1) \\
 &= e(t_\ell) - i(t_\ell) + e(t_r) - i(t_r) - 1 \\
 &= 1 + 1 - 1 = 1
 \end{aligned}$$

2.

$$\begin{aligned}
 h(\square) &= 0 = i(\square) \\
 h(\langle \bigcirc, t_\ell, t_r \rangle) &= 1 + \max\{h(t_\ell), h(t_r)\} \\
 &\leq 1 + \max\{i(t_\ell), i(t_r)\} \\
 &\leq 1 + i(t_\ell) + i(t_r) \\
 &= i(\langle \bigcirc, t_\ell, t_r \rangle)
 \end{aligned}$$

3.

$$\begin{aligned}
 e(\square) &= 1 = 2^0 = 2^{h(\square)} \\
 e(\langle \bigcirc, t_\ell, t_r \rangle) &= e(t_\ell) + e(t_r) \\
 &\leq 2^{h(t_\ell)} + 2^{h(t_r)} \\
 &\leq 2 \cdot 2^{\max\{h(t_\ell), h(t_r)\}} \\
 &= 2^{1+\max\{h(t_\ell), h(t_r)\}} \\
 &= 2^{h(\langle \bigcirc, t_\ell, t_r \rangle)}
 \end{aligned}$$

### 6.1.2 Weglänge

**Definition 22.** Für einen Binärbaum  $i \in \mathcal{B}$  wird definiert:

- die *innere Weglänge* als

$$w_i(t) = \sum_{a \in I(t)} h(a, t);$$

- die *äussere Weglänge* als

$$w_e(t) = \sum_{a \in E(t)} h(a, t);$$

- die *mittlere innere Weglänge* als:

$$\bar{w}(t) = \frac{w_i(t)}{i(t)};$$

- die *mittlere äussere Weglänge (mittlere Höhe)* als:

$$\bar{h}(t) = \frac{w_e(t)}{e(t)}.$$

**Lemma 55.** Zwischen innerer und äussere Weglänge eines Binärbaumes  $t$  besteht die Beziehung

$$w_e(t) = w_i(t) + 2i(t).$$

*Beispiel 11.* Illustration zur Weglänge in Abbildung 23.

## 6.2 Binäre Suchbäume

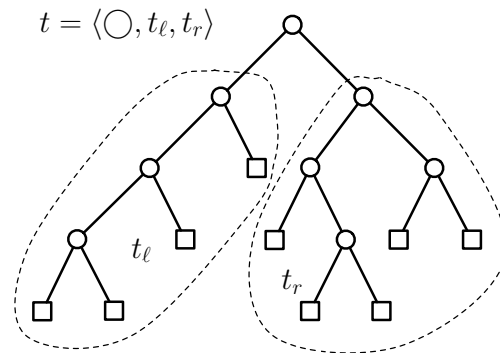
### 6.2.1 Definition

Es bezeichne im folgenden  $A$  eine totalgeordnete Menge (z.B.  $\mathbb{Z}, \mathbb{R}, \dots$ ). Wir betrachten nun Binärbäume, für die noch eine Abbildung  $v : I(t) \rightarrow A$  gegeben ist. In dieser Situation bezeichnen Für  $t = \langle \circ, t_\ell, t_r \rangle \in \mathcal{B}$

$$v_\ell : I(t_\ell) \rightarrow A : a \mapsto v(a)$$

$$v_r : I(t_r) \rightarrow A : a \mapsto v(a)$$

die Restriktionen von  $v$  auf den linken bzw. rechten Teilbaum.



$$w_i(t) = 14, w_e(t) = 30, i(t) = 8, \bar{w}(t) = \frac{14}{8} = \frac{7}{4}, \bar{h}(t) = \frac{30}{9} = \frac{10}{3}$$

Abbildung 23: Illustration zur Weglänge

**Definition 23.** Ein Paar  $(t, v)$  mit  $t \in \mathcal{B}$  und einer Abbildung  $v : I(t) \rightarrow A$  heißt *binärer Suchbaum* über  $A$ , wenn

- entweder  $t = \square$  (und somit  $v = \lambda =$  leere Funktion),
- oder  $t = \langle \bigcirc, t_\ell, t_r \rangle$  und mit
  - $v(\bigcirc) = a$ ,
  - $(t_\ell, v_\ell)$  ist ein binärer Suchbaum über  $A_{<a} = \{b \in A; b < a\}$ ,
  - $(t_r, v_r)$  ist ein binärer Suchbaum über  $A_{>a} = \{b \in A; b > a\}$ .

Insbesondere gilt also  $\max_{i \in I(t_\ell)} v(i) < v(\bigcirc) = a < \min_{i \in I(t_r)} v(i)$ .

$\mathcal{BS}(A)$  bezeichnet die Menge der binären Suchbäume über  $A$  und  $\mathcal{BS}_n(A)$  die Menge der  $(t, v) \in \mathcal{BS}(A)$  mit  $t \in \mathcal{B}_n$ .

Beachte: die Abbildung  $v : I(t) \rightarrow A$  ist immer eine *injektive* Abbildung, d.h. jedes Element  $a \in A$  kann in einem binären Suchbaum höchstens einmal auftreten.

*Beispiel 12.* Abbildung 24 zeigt einen binären Suchbaum über der Menge  $A = \{4, 5, 7, 9, 10, 11, 17, 22, 24, 31\}$ , oder auch über jeder Menge, die  $A$  als Teilmenge enthält.

**Lemma 56.** 1. Für binäre Suchbäume gilt die Strukturaussage

$$\mathcal{BS}(A) = \bigsqcup_{a \in A} (\{a\} \times \mathcal{BS}(A_{<a}) \times \mathcal{BS}(A_{>a})).$$

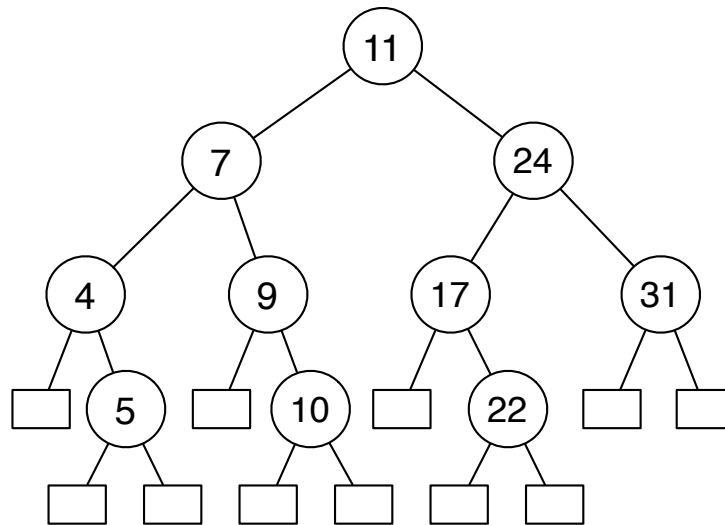


Abbildung 24: Binärer Suchbaum

2. Für die Anzahl der binären Suchbäume mit  $n$  inneren Knoten über einem endlichen Alphabet  $A$  gilt

$$\#\mathcal{BS}_n(A) = \binom{\#A}{n} \cdot c_n.$$

Beachte hierfür: ist  $t \in \mathcal{B}_n$  und ist eine  $n$ -elementige Teilmenge  $C$  von  $A$  gewählt (als diejenigen Elemente, die in dem Suchbaum auftreten sollen, so ist die Abbildung  $v : I(t) \rightarrow C$  durch die Struktur von  $t$  eindeutig festgelegt.

### 6.2.2 Suchen in binären Suchbäumen

Binäre Suchbäume dienen zum Speichern von Information und sollen einen schnellen Zugriff ermöglichen. Grundproblem ist die Frage, ob ein gegebenes Element  $a \in A$  in  $(t, v) \in \mathcal{BS}(A)$  vorkommt. Die Idee der Suche ist offensichtlich:

- wenn der Baum nur aus einem Blatt besteht, endet die Suche mit einem negativen Resultat;
- andernfalls: vergleiche  $e$  mit dem Element an der Wurzel von  $t$ :
  - wenn dies gleich  $a$  ist, hat man eine positive Antwort,
  - wenn  $a$  kleiner ist als das Wurzelement, sucht man  $a$  im linken Teilbaum;

- wenn  $a$  grösser ist als das Wurzelement, sucht man  $a$  im linken Teilbaum;

---

**Algorithm 13** Suche in binären Suchbäumen
 

---

```

procedure BSTSEARCH( $(t, v) :: bst, a :: elem$ )  ▷ Suche  $a$  im Suchbaum  $(v, t)$ 
  if  $t = \square$  then
    return(fail)
  end if
  if  $v(\bigcirc) = a$  then
    return(found)
  end if
  if  $v(\bigcirc) > a$  then
    BSTsearch( $(t_\ell, v_\ell), a$ )
  else
    BSTsearch( $(t_r, v_r), a$ )
  end if

end procedure

```

---

Die Bedeutung der Weglänge für diesen Suchprozess sollte klar sein:

- Die (mittlere) innere Weglänge von  $t$  ist ein Maß für den (mittleren) Aufwand erfolgreicher Suche in  $t$ .
- Die (mittlere) äussere Weglänge von  $t$  ist ein Maß für den (mittleren) Aufwand erfolgloser Suche in  $t$ .

### 6.2.3 Konstruktion von binären Suchbäumen

Aus einer Folge  $\mathbf{a} = a_1 a_2 \dots a_n \in A^*$  konstruiert man iterativ einen binären Suchbaum, der genau die (verschiedenen) Folgeelemente enthält. Dafür noch eine Bezeichnung: Für eine Folge  $\mathbf{a} = a_1 a_2 \dots a_n \in A^+$  sei

$$\mathbf{a}_< = a_{i_1} a_{i_2} \dots a_{i_k} \quad \text{mit } 1 < i_1 < i_2 < \dots < i_k \quad \text{die Teilfolge der } a_i < a_1,$$

$$\mathbf{a}_> = a_{j_1} a_{j_2} \dots a_{j_\ell} \quad \text{mit } 1 < j_1 < j_2 < \dots < j_\ell \quad \text{die Teilfolge der } a_j > a_1.$$

**Definition 24.** Der von einer Folge  $\mathbf{a} = a_1 a_2 \dots a_n \in A^*$  erzeugte binäre Suchbaum  $bs(\mathbf{a}) = (t, v)$  ist definiert durch

- Für die leere Folge  $\varepsilon$  ist  $bs(\varepsilon) = (\square, \varepsilon)$ ;

– Für  $a \in A^+$  sind  $t = \langle \bigcirc, t_\ell, t_r \rangle$  und  $v : t \rightarrow A$  gegeben durch

$$v(\bigcirc) = a_1, \quad bs(\mathbf{a}_<) = (t_\ell, v_\ell), \quad bs(\mathbf{a}_>) = (t_r, v_r)$$

Ein sequentieller Algorithmus zur Konstruktion von binären Suchbäumen verwendet sukzessives Einfügen bei erfolgloser Suche. Dies soll ohne weitere Formalisierung an einem Beispiel demonstriert werden, siehe Abbildung 25.

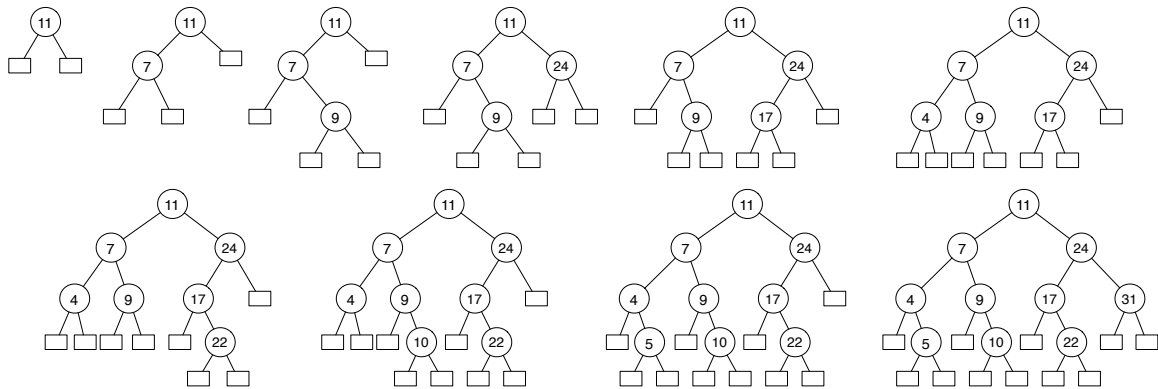


Abbildung 25: Iterative Konstruktion des binären Suchbaumes zu der Folge  $\mathbf{a} = (11, 7, 9, 24, 17, 4, 22, 10, 5, 31)$

*Beispiel 13.* Abbildung 25 zeigt, dass der binäre Suchbaum aus Abbildung 24 durch die Folge  $\mathbf{a} = (11, 7, 9, 24, 17, 4, 22, 10, 5, 31)$  erzeugt wird. Eine andere Folge, die denselben Suchbaum erzeugt, ist beispielsweise  $\mathbf{b} = (11, 24, 31, 17, 7, 9, 22, 4, 5, 10)$ .

**Definition 25.** Zwei Folgen  $\mathbf{a}, \mathbf{b} \in A^*$  sollen äquivalent heißen, notiert mit  $\mathbf{a} \equiv \mathbf{b}$ , wenn sie den gleichen Suchbaum erzeugen, d.h., wenn  $bs(\mathbf{a}) = bs(\mathbf{b})$  ist.

**Lemma 57.** Für Folgen  $\mathbf{a} = a_1 a_2 \dots a_m, \mathbf{b} = b_1 b_2 \dots b_n \in A^*$  gilt

$$\mathbf{a} \equiv \mathbf{b} \iff \begin{cases} a_1 = b_1 \\ \mathbf{a}_< \equiv \mathbf{b}_< \\ \mathbf{a}_> \equiv \mathbf{b}_> \end{cases}$$

Für die Analyse des Suchaufwandes in einem Binärbaum, den man zuvor konstruiert hat, sind nun folgende Überlegungen relevant:

- Wird  $a \in A$  in den  $A$ -Suchbaum  $(t, v)$  (erfolgreich) gesucht, so ist die Höhe  $h(b, t)$  des Knotens  $b \in I(t)$  mit  $v(b) = a$  relevant.

- Es kommt sowohl auf das gesuchte Datum an, als auch auf den Suchbaum (d.h. die Reihenfolge, in der die Daten zur Konstruktion des Suchbaumes verwendet wurden).
- Man kann sich auf die erfolgreiche Suche beschränken, da innere und äussere Weglänge in einem bekannten Zusammenhang mit einander stehen, siehe Lemma 55.

### 6.2.4 Suchkomplexität

Für das Konstruieren von binären Suchbäumen und die Suche in ihnen wird das *Permutationsmodell* verwendet:

- Für  $n$  (verschiedene) Daten  $\in A$  betrachte alle  $n!$  Permutationen als gleichwahrscheinliche inputs für die Konstruktion eines  $bs$ .
- Man kann ohne Einschränkung annehmen, dass  $A = \{1, 2, \dots, n\}$  ist und dass alle Permutationen  $\sigma \in \mathcal{S}_n$  inputs sind.
- Es wird mit gleicher Wahrscheinlichkeit nach jedem der Daten  $k \in \{1, 2, \dots, n\}$  gesucht

Für  $\sigma \in \mathcal{S}_n$  sei

$$bs(\sigma) = (t^\sigma, v^\sigma) \quad \text{mit } t \in \mathcal{B}_n$$

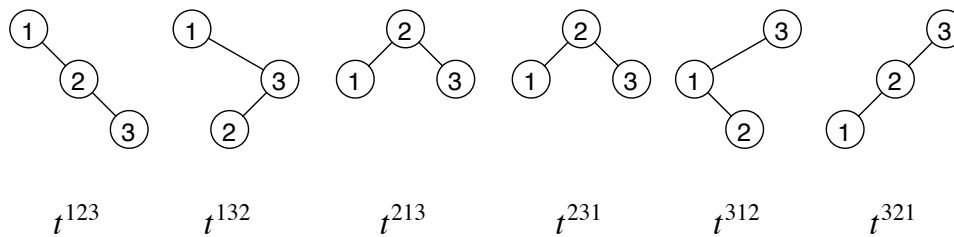
der binäre Suchbaum, der aus  $\sigma$  durch Lesen von links nach rechts entsteht.

*Bemerkung 19.* Beachte:

1. Verschiedene Permutationen  $\sigma \in \mathcal{S}_n$  können den gleichen Suchbaum liefern!
2.  $v^\sigma$  ist durch  $t^\sigma$  eindeutig bestimmt, also in diesem Kontext der Komplexitätsuntersuchung eigentlich redundant!
3. Für einen festen  $A$ -Suchbaum  $(t, v)$  ist die (mittlere) innere Weglänge  $w_i(t)$  bzw.  $\bar{w}(t) = w_i(t)/n$  das Maß für den Suchaufwand.

*Beispiel 14.* Die Suchbäume zu den Permutationen aus  $\mathcal{S}_3$  sind in Abbildung 26 dargestellt. Es gilt

$$w_i(t^{123}) = w_i(t^{132}) = w_i(t^{312}) = w_i(t^{321}) = 3, w_i(t^{213}) = w_i(t^{231}) = 2$$

Abbildung 26: Permutationen und ihre Suchbäume für  $n = 3$ 

also

$$w_3 = \frac{1}{3!} \sum_{\sigma \in \mathcal{S}_3} w_i(\sigma) = \frac{3 + 3 + 2 + 2 + 3 + 3}{6} = \frac{8}{3},$$

$$\bar{w}_3 = \frac{1}{3} w_3 = \frac{8}{9}.$$

Die für die Beurteilung der Suchkomplexität relevanten Begriffe sind die *mittlere innere Weglänge* und die *mittlere innere Weglänge pro innerem Knoten*, wobei die Mittelbildung über alle Permutationen  $\sigma \in \mathcal{S}_n$  läuft. Jeder binäre Suchbaum  $(t, v) \in \mathcal{BS}_n$  tritt also mit der Häufigkeit in der Mittelbildung auf, mit der als Suchbaum von Permutationen  $\sigma \in \mathcal{S}_n$  konstruiert wird.

**Definition 26.** 1. Die *mittlere innere Weglänge* für binäre Suchbäume mit  $n$  inneren Knoten ist definiert als

$$w_n = \frac{1}{n!} \sum_{\sigma \in \mathcal{S}_n} w_i(t^\sigma).$$

2. Der *mittlere Suchaufwand* (= *mittlere innere Weglänge pro Knoten*) für binäre Suchbäume mit  $n$  inneren Knoten ist definiert als

$$\bar{w}_n = \frac{1}{n!} \sum_{\sigma \in \mathcal{S}_n} \bar{w}(t^\sigma) = \frac{1}{n} w_n.$$

Ziel dieses Abschnitts ist der Beweis des grundlegenden

**Theorem 58.** Die *mittlere innere Weglänge*  $w_n$  für binäre Suchbäume mit  $n$  inneren Knoten genügt der *Quicksort-Rekursion*:

$$w_n = n - 1 + \frac{2}{n} \sum_{k=0}^{n-1} w_k, \quad w_0 = 0.$$

Die exakten Werte der  $w_n$  und somit deren asymptotisches Verhalten sind also aus Abschnitt ?? bekannt:

$$w_n = 2(n+1)H_n - 4n \sim 2n \ln n.$$

Daraus ergibt sich:

**Folgerung 59.** *Der mittlere Suchaufwand bei Verwendung von binären Suchbäumen im Permutationsmodell ist*

$$\bar{w}_n = \frac{1}{n} w_n = \left(2 + \frac{2}{n}\right) H_n - 4 \sim 2 \ln n.$$

Für den Beweis ist es erforderlich zu untersuchen, wieviele Permutationen  $\sigma \in \mathcal{S}_n$  den gleichen binären Suchbaum erzeugen. Überraschenderweise kann man dies in einer Formel ganz präzise angeben. Dazu eine Definition:

**Definition 27.** Für einen Binärbaum  $t$  und einen inneren Knoten  $b \in I(t)$  bezeichne  $t_b$  den Teilbaum von  $t$ , der  $b$  als Wurzel hat. Dann sei

$$\nu(t) := \prod_{b \in I(t)} i(t_b)$$

das Produkt über die Anzahlen innerer Knoten aller Teilbäume von  $t$ . Man bezeichnet  $\nu(t)$  auch als die *Hakenlänge(hooklength)* von  $t$ .

**Theorem 60.** *Für jeden Binärbaum  $t \in \mathcal{B}_n$  gibt es genau*

$$\frac{n!}{\nu(t)} = \frac{n!}{\prod_{b \in I(t)} i(t_b)}$$

*Permutationen  $\sigma \in \mathcal{S}_n$ , die diesen Binärbaum als ihren binären Suchbaum erzeugen, für die also  $t^\sigma = t$  gilt.*

*Beispiel 15.* Als Beispiel: der Binärbaum in Abbildung 27 hat also Folge der Anzahlen innerer Knoten für die Teilbäume (aufsteigende geordnet)

$$((i(t_b))_{b \in I(t)} = (1, 1, 1, 2, 2, 3, 4, 8).$$

Somit ist

$$\nu(t) = 1 \cdot 1 \cdot 1 \cdot 2 \cdot 2 \cdot 3 \cdot 4 \cdot 8 = 384.$$

Die Anzahl der Permutationen  $\in \mathcal{S}_8$ , die genau diesen Binärbaum als ihren binären Suchbaum haben, ist also

$$\frac{8!}{1 \cdot 1 \cdot 1 \cdot 2 \cdot 2 \cdot 3 \cdot 4 \cdot 8} = 105.$$

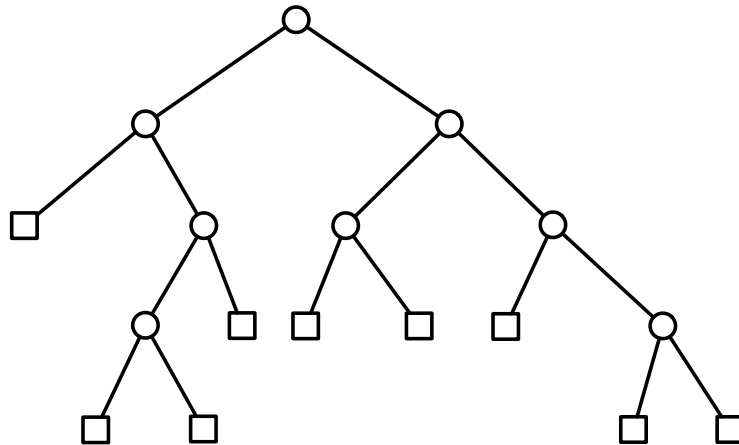


Abbildung 27: Binärbaum

### 6.2.5 Beweis von Theorem 60 und von Theorem 58

Zunächst wird Funktion  $\mathbf{p} : \mathcal{B} \rightarrow \mathbb{R}_{>0}$  definiert, die auf jeder Menge  $\mathcal{B}_n$  eine Wahrscheinlichkeitsverteilung ist.

**Definition 28.**  $\mathbf{p} : \mathcal{B} \rightarrow \mathbb{R}_{>0}$  ist definiert durch die Rekursion

$$\mathbf{p}(t) = \begin{cases} 1 & \text{falls } t = \square \\ \frac{1}{i(t)} \cdot \mathbf{p}(t_\ell) \cdot \mathbf{p}(t_r) & \text{falls } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

**Lemma 61.** Für die Funktion  $\mathbf{p} : \mathcal{B} \rightarrow \mathbb{R}_{>0}$  gilt:

1. Für jedes  $n \geq 0$  ist  $\mathbf{p}$  eine Wahrscheinlichkeitsverteilung auf  $\mathcal{B}_n$ .
2. Für jeden Binärbaum  $t \in \mathcal{B}$  ist  $\mathbf{p}(t) = \nu(t)^{-1}$ .

*Beweis.* Beide Aussagen werden per Induktion bewiesen.

1. Aus der Definition folgt unmittelbar, dass alle  $\mathbf{p}(t) \geq 0$  sind. Sei nun  $p_n := \sum_{t \in \mathcal{B}_n} \mathbf{p}(t)$ .

$p_0 = 1$  ist per Definition klar.

Für  $n > 0$  verwendet man die Beziehung

$$\mathcal{B}_n \leftrightarrow \sum_{1 \leq k \leq n} \mathcal{B}_{k-1} \times \mathcal{B}_{n-k}.$$

Dann ist

$$\begin{aligned}
 p_n &= \sum_{t \in \mathcal{B}_n} \mathbf{p}(t) \\
 &= \sum_{1 \leq k \leq n} \sum_{t_\ell \in \mathcal{B}_{k-1}} \sum_{t_r \in \mathcal{B}_{n-k}} \frac{1}{n} \cdot \mathbf{p}(t_\ell) \cdot \mathbf{p}(t_r) \\
 &= \frac{1}{n} \cdot \sum_{1 \leq k \leq n} \sum_{t_\ell \in \mathcal{B}_{k-1}} \mathbf{p}(t_\ell) \cdot \sum_{t_r \in \mathcal{B}_{n-k}} \mathbf{p}(t_r) \\
 &= \frac{1}{n} \cdot \sum_{1 \leq k \leq n} p_{k-1} \cdot p_{n-k} \\
 &= \frac{1}{n} \cdot \sum_{1 \leq k \leq n} 1 \cdot 1 = 1.
 \end{aligned}$$

2. Für den Baum  $t = \square$  ist  $I(\square) = \emptyset$ , In diesem Fall ist die Behauptung richtig, denn ein leeres Produkt ist  $= 1$ .

Ist  $t = \langle \circ, t_\ell, t_r \rangle$ , so gilt wegen  $I(t) = \{\circ\} \cup I(t_\ell) \cup I(t_r)$  offensichtlich

$$\nu(t) = i(t) \cdot \prod_{b \in I(t_\ell)} i(t_b) \cdot \prod_{c \in I(t_r)} i(t_c) = i(t) \cdot \nu(t_\ell) \cdot i(t_r).$$

Damit ist aber auch

$$\frac{1}{\nu(t)} = \frac{1}{i(t)} \cdot \frac{1}{\nu(t_\ell)} \cdot \frac{1}{i(t_r)}.$$

Das zeigt aber, dass die  $\nu(t)^{-1}$  die gleiche Rekursion erfüllen, wie die  $\mathbf{p}(t)$ . Also müssen sie gleich sein.  $\square$

**Definition 29.** Das *Shuffle-Produkt*  $\mathbf{a} \amalg \mathbf{b}$  von Wörtern  $\mathbf{a} = a_1 a_2 \dots a_m$  und  $\mathbf{b} = b_1 b_2 \dots b_n$  über einem Alphabet  $\Sigma$  besteht aus der Menge alle Wörter  $\mathbf{w} = w_1 w_2 \dots w_{m+n}$ , für die gilt

- es gibt Indices  $1 \leq i_1 < i_2 < \dots < i_m$  mit  $a_s = w_{i_s}$  ( $1 \leq s \leq m$ ),
- es gibt Indices  $1 \leq j_1 < j_2 < \dots < j_n$  mit  $b_t = w_{j_t}$  ( $1 \leq t \leq n$ ),
- $\{i_1, \dots, i_m\} \cap \{j_1, \dots, j_n\} = \emptyset$ .

Das Shuffle-Produkt von formalen Sprachen  $L, M \subseteq \Sigma^*$  ist

$$L \amalg M := \bigcup_{\mathbf{a} \in L, \mathbf{b} \in M} \mathbf{a} \amalg \mathbf{b}.$$

In den zu  $\mathbf{a} \amalg \mathbf{b}$  gehörenden treten also in “gemischter Reihenfolge” die Buchstaben aus  $\mathbf{a}$  und  $\mathbf{b}$  auf, wobei die interne Reihenfolge für jedes der beiden Wörter  $\mathbf{a}$  und  $\mathbf{b}$  erhalten bleibt.

*Beispiel 16.* Ist  $\mathbf{a} = a_1a_2a_3$  und  $\mathbf{b} = b_1b_2$ , so besteht  $\mathbf{a} \amalg \mathbf{b}$  aus

$$\begin{array}{cccccc} a_1a_2a_3b_1b_2 & a_1a_2b_1a_3b_2 & a_1a_2b_1b_2a_3 & a_1b_1a_2a_3b_2 & a_1b_1a_2b_2a_3 \\ a_1b_1b_2a_2a_3 & b_1a_1a_2a_3b_2 & b_1a_1a_2b_2a_3 & b_1a_1b_2a_2a_3 & b_1a_1b_2a_2a_3 \end{array}$$

Wenn es aber Buchstaben gibt, die sowohl in  $\mathbf{a}$ , als auch in  $\mathbf{b}$  vorkommen, können einige dieser Wörter gleich sein.

Ist  $\mathbf{a} = 010$  und  $\mathbf{b} = 10$ , so besteht  $\mathbf{a} \amalg \mathbf{b}$  nur aus den Wörtern

$$01010 \quad 01100 \quad 10100 \quad 10010$$

Eine induktive Definition des Shuffle-Produkts: für  $\mathbf{a}, \mathbf{b} \in \Sigma^*, x, y \in \Sigma$  gilt

$$\begin{aligned} \mathbf{a} \amalg \varepsilon &= \{\mathbf{a}\} \\ \varepsilon \amalg \mathbf{b} &= \{\mathbf{b}\} \\ (\mathbf{a} \cdot x) \amalg (\mathbf{b} \cdot y) &= (\mathbf{a} \amalg (\mathbf{b} \cdot y)) \cdot x \cup ((\mathbf{a} \cdot x) \amalg \mathbf{b}) \cdot y \end{aligned}$$

**Lemma 62.** 1. Das Shuffle-Produkt  $\amalg$  ist kommutativ und assoziativ.

2. Haben  $\mathbf{a} = a_1a_2 \dots a_m$  und  $\mathbf{b} = b_1b_2 \dots b_n$  keinen Buchstaben gemeinsam, d.h. ist  $\{a_1, \dots, a_m\} \cap \{b_1, \dots, b_n\} = \emptyset$ , so gilt

$$\#(\mathbf{a} \amalg \mathbf{b}) = \binom{m+n}{m} = \frac{(m+n)!}{m! \cdot n!}$$

Die Bedeutung des Shuffle-Produkts für die binären Suchbäume ergibt sich aus folgendem Satz, der eine unmittelbare Konsequenz von Lemma 57 ist:

**Satz 63.** Ist  $\mathbf{a} = a_1a_2 \dots a_n$  eine Permutation von  $A$  und  $\mathbf{b}$  eine Permutation von  $A \setminus \{a_1\}$ , so sind folgende Aussagen äquivalent:

1.  $\mathbf{a} \equiv a_1 \mathbf{b}$
2.  $\mathbf{b} \in \mathbf{c} \amalg \mathbf{d}$ , wobei  $\mathbf{c}, \mathbf{d}$  Permutationen<sup>10</sup> sind mit  $\mathbf{c} \equiv \mathbf{a}_{<}$  und  $\mathbf{d} \equiv \mathbf{a}_{>}$ .

<sup>10</sup>In dieser Situation ist  $\mathbf{c}$  eine Permutation von  $\{x \in A; x < a_1\}$  und  $\mathbf{d}$  eine Permutation von  $\{x \in A; x > a_1\}$ .

**Folgerung 64.** Bezeichnet für einen Binärbaum  $t = \langle \bigcirc, t_\ell, t_r \rangle \in \mathcal{B}_n$

$$\#t = \#\{\sigma \in \mathcal{S}_n; t^\sigma = t\}$$

die Anzahl den Permutationen von  $\{1, 2, \dots, n\}$ , die  $t$  als ihren binären Suchbaum erzeugen, so gilt, falls  $t_\ell \in \mathcal{B}_{k-1}$  und  $t_r \in \mathcal{B}_{n-k}$ :

$$\#t = \binom{n-1}{k-1} \cdot \#t_\ell \cdot \#t_r.$$

scheint man dies in der Form

$$\frac{1}{n!} \#t = \frac{1}{n} \cdot \frac{1}{(k-1)!} \#t_\ell \cdot \frac{1}{(n-k)!} \#t_r,$$

so erkennt man, dass die Zahlen  $\frac{1}{n!} \#t$  der gleichen Rekursion genügen, wie die  $\mathbf{p}_n(t)$ . Also gilt

$$\#t = n! \cdot \mathbf{p}_n(t) = \frac{n!}{\nu(t)},$$

oder anders formuliert

- Für jeden Binärbaum  $t \in \mathcal{B}_n$  ist

$$\mathbf{p}_n(t) = \frac{n!}{\nu(t)}$$

die Wahrscheinlichkeit dafür, dass bei zufälliger Wahl einer Permutation  $\sigma \in \mathcal{S}_n$  dieses  $t$  als binärer Suchbaum erzeugt wird, d.h.  $t^\sigma = t$ .

Damit ist Theorem 60 bewiesen. □

*Beweis.* (von Theorem 58)

$$\begin{aligned} w_n &= \frac{1}{n!} \sum_{\sigma \in \mathcal{S}_n} w_i(t^\sigma) \\ &= \sum_{t \in \mathcal{B}_n} \mathbf{p}_n(t) \cdot w_i(t) \\ &= \sum_{k=1}^n \sum_{\substack{t_\ell \in \mathcal{B}_{k-1} \\ t_r \in \mathcal{B}_{n-k}}} \frac{1}{n} \cdot \mathbf{p}_{k-1}(t_\ell) \cdot \mathbf{p}_{n-k}(t_r) (w_i(t_\ell) + w_i(t_r) + n - 1) \\ &= \frac{1}{n} \sum_{k=1}^n (w_{k-1} + w_{n-k} + n - 1) \\ &= n - 1 + \frac{2}{n} \sum_{k=0}^{n-1} w_k \quad \square \end{aligned}$$

## 6.3 Sortierkomplexität und Information

### 6.3.1 Mittlere Höhe von Binärbäumen

In Lemma 54 wurde bezüglich der Höhe  $h(t)$  und der Anzahl der Blätter  $e(t)$  eines Binärbaums  $t$  die einfache herzuleitende, aber fundamentale Ungleichung

$$h(t) \geq \log e(t)$$

festgehalten. Dies wird nun ergänzt und verschärft durch eine analoge Aussage für die mittlere Höhe, die ebenfalls fundamental ist, deren Beweis aber nicht ganz so einfach ist.

**Definition 30.** Die *mittlere Höhe* eines Binärbaumes  $t$  ist definiert als

$$\bar{h}(t) = \frac{1}{e(t)} \sum_{b \in E(t)} h(b, t).$$

*Beispiel 17.* Abbildung 28 zeigt ein Beispiel. Abbildung 29 illustriert, wie sich die mittlere Höhe eines Binärbaumes aus der mittleren Höhe seiner Teilbäume berechnet.

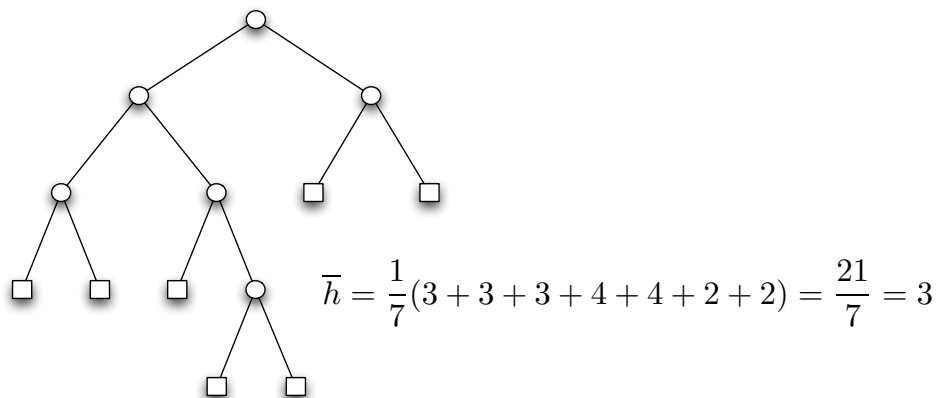


Abbildung 28: Mittlere Höhe eines Binärbaumes

**Lemma 65.** Für die mittlere Höhe von Binärbäumen gilt die rekursive Beziehung

$$\bar{h}(t) = \begin{cases} 0 & \text{falls } t = \square \\ 1 + \frac{e(t_\ell)}{e(t)} \cdot \bar{h}(t_\ell) + \frac{e(t_r)}{e(t)} \cdot \bar{h}(t_r) & \text{falls } t = \langle \circ, t_\ell, t_r \rangle \end{cases}$$

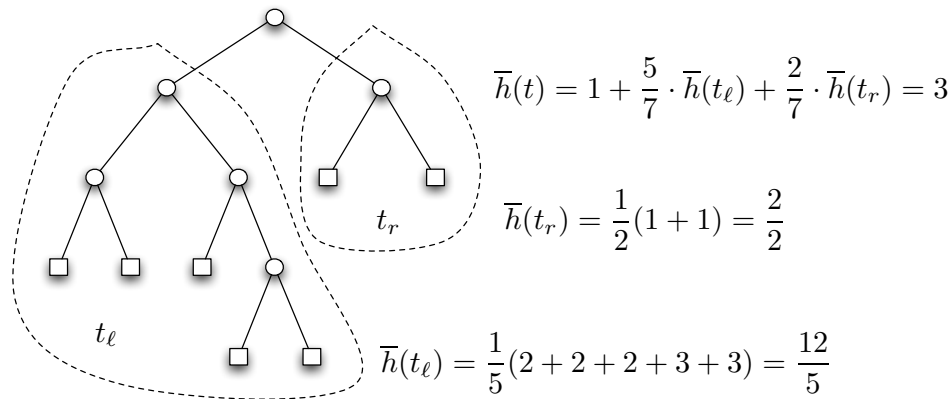


Abbildung 29: Rekursive Struktur der mittleren Höhe

*Beweis.* Der Induktionsanfang mit  $t = \square$  ist klar. Der Induktionsschritt:

$$\begin{aligned} \bar{h}(\langle \circ, t_\ell, t_r \rangle) &= \frac{1}{e(t)} \sum_{b \in E(t)} h(b, t) \\ &= \frac{1}{e(t)} \left( \sum_{b \in E(t_\ell)} [h(b, t_\ell) + 1] + \sum_{b \in E(t_r)} [h(b, t_r) + 1] \right) \\ &= \frac{e(t_\ell) + e(t_r)}{e(t)} + \frac{e(t_\ell)}{e(t)} \cdot \bar{h}(t_\ell) + \frac{e(t_r)}{e(t)} \cdot \bar{h}(t_r) \\ &= 1 + \frac{e(t_\ell)}{e(t)} \cdot \bar{h}(t_\ell) + \frac{e(t_r)}{e(t)} \cdot \bar{h}(t_r) \quad \square \end{aligned}$$

Ein fundamentale Ungleichung für Binärbäume ist nun

**Theorem 66.** Für jeden Binärbaum  $t$  gilt:

$$\bar{h}(t) \geq \log e(t)$$

*Beweis.* Dies folgt per Induktion über der rekursiven Aufbau von Binärbäumen:

$$\begin{aligned}
 \bar{h}(\square) &= 0 = \log 1 \\
 \bar{h}(\langle \circ, t_\ell, t_r \rangle) &= 1 + \frac{e(t_\ell)}{e(t)} \cdot \bar{h}(t_\ell) + \frac{e(t_r)}{e(t)} \cdot \bar{h}(t_r) \\
 &\geq 1 + \frac{e(t_\ell)}{e(t)} \cdot \log e(t_\ell) + \frac{e(t_r)}{e(t)} \cdot \log e(t_r) \\
 &= 1 + \underbrace{\frac{e(t_\ell)}{e(t)} \cdot \log \frac{e(t_\ell)}{e(t)} + \frac{e(t_r)}{e(t)} \cdot \log \frac{e(t_r)}{e(t)}}_{-H\left(\frac{e(t_\ell)}{e(t)}, \frac{e(t_r)}{e(t)}\right)} + \log e(t) \\
 &= \underbrace{1 - H\left(\frac{e(t_\ell)}{e(t)}, \frac{e(t_r)}{e(t)}\right)}_{\geq 0} + \log e(t)
 \end{aligned}$$

Dabei ist

$$H(x, 1-x) = -x \log x - (1-x) \log(1-x)$$

die “Entropiefunktion” von Claude SHANNON, deren Eigenschaften in Abschnitt 6.3.4 diskutiert werden.  $\square$

Der Graph der Entropiefunktion  $H(x, 1-x)$  (gelbe Kurve) ist in Abbildung 30 wiedergegeben, zusammen mit Kreisbögen vom Radius  $1/2$  (grüne Kurve) bzw. 1 (rote Kurve) um das Zentrum ( $x = 1/2, y = 0$ ) in der Ebene.

### 6.3.2 Untere Schranke der Sortierkomplexität

Vergleichbasierte Sortieralgorithmen können mittels beschrifteter binärer Bäume (Entscheidungs bäume) veranschaulicht werden:

- Eingaben sind Liste  $(x_1, x_2, \dots, x_n)$  der Länge  $n$ , wobei die  $x_i$  paarweise verschieden sein sollen (Permutationsmodell);
- innere Knoten der Entscheidungsbäumes sind mit Paaren  $(i, j)$  ( $1 \leq i < j \leq n$ ) beschriftet, die anzeigen, welcher Vergleich ausgeführt wird;
- äussere Knoten (Blätter) sind mit Permutationen (=lineare Anordnungen) von  $\{1, 2, \dots, n\}$  beschriftet, die zeigen, welche dieser Ordnungen identifiziert wurde;

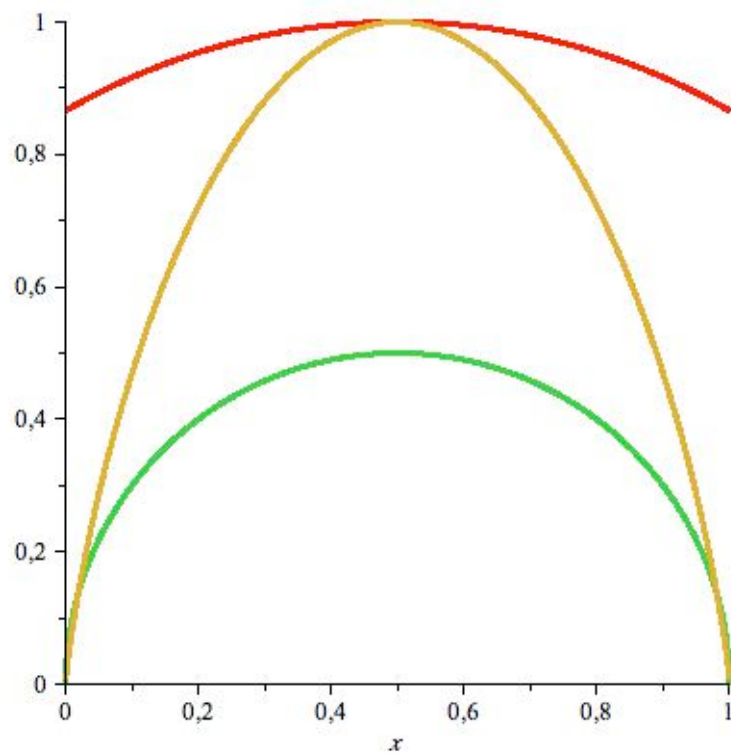


Abbildung 30: Graph der Entropiefunktion  $H(x, 1 - x)$  (gelb)

- jeder Weg von der Wurzel zu einem Blatt identifiziert schrittweise die Struktur der Eingabe:  
an einem mit  $(i, j)$  beschrifteten Knoten
  - gehe weiter in den linken Teilbaum, falls  $x_i < x_j$ ;
  - gehe weiter in den rechten Teilbaum, falls  $x_i > x_j$ ;
- an einem Blatt: die am Blatt notierte Permutation stellt die Ordnungsrelation der Eingabe dar.

*Beispiel 18.* Abbildung 31 zeigt den Entscheidungsbaum für MERGESORT für Listen der Länge 4. Der Baum hat  $4! = 24$  Blätter und Höhe 5.

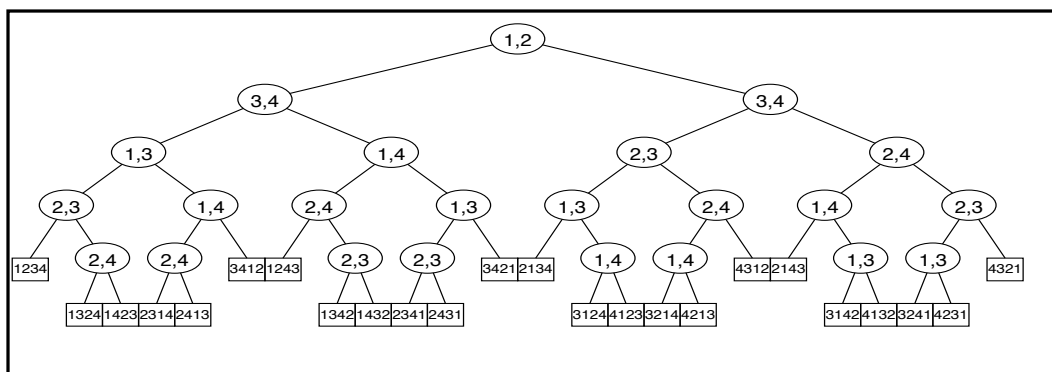


Abbildung 31: Entscheidungsbaum für Mergesort für Listenlänge 4

- Erinnerung an Lemma für jeden binären Baum  $t$  gilt

$$\log e(t) \leq h(t)$$

wobei  $e(t) =$  Anzahl der Blätter und  $h(t) =$  Höhe ist

- Operiert ein Sortieralgorithmus  $\mathcal{A}$  auf Eingaben der Länge  $n$ , so muss der zugehörige Entscheidungsbaum  $t_{\mathcal{A}}(n)$  mindestens  $n!$  Blätter haben, denn alle  $n!$  möglichen Permutationen müssen identifiziert werden können.
- Somit gilt für die Anzahl  $V_{\mathcal{A}}(n)$  im worst-case

$$V_{\mathcal{A}}(n) = h(t_{\mathcal{A}}(n)) \geq \lceil \log e(t_{\mathcal{A}}(n)) \rceil \geq \lceil \log n! \rceil$$

Erinnerung: STIRLINGS Formel

$$n! = \left(\frac{n}{e}\right)^n \sqrt{2\pi \cdot n} \left(1 + O\left(\frac{1}{n}\right)\right)$$

**Theorem 67.** *Jeder vergleichsbasierte Sortieralgorithmus benötigt auf Listen der Länge  $n$  im worst-case mindestens*

$$n \log n - n \log e + O(\log n) \approx n \log n - 1.44n \quad \text{Vergleiche.}$$

Nun zur average-case Analyse.

- Jeder Entscheidungsbaum für ein Sortierverfahren auf inputs der Länge  $n$  hat genau  $n!$  Blätter.
- Die mittlere Höhe eines Binärbaumes ist

$$\bar{h}(t) = \frac{1}{e(t)} \sum_{b \in E(t)} h(b, t),$$

wobei  $E(t)$  = Menge der Blätter von  $t$ ,  $h(b, t)$  = Höhe des Blattes  $b$  in  $t$

Erinnerung an Theorem 66: Für jeden Binärbaum gilt

$$\bar{h}(t) \geq \log e(t).$$

Daraus ergibt sich

**Theorem 68.** *Jeder vergleichsbasierte Sortieralgorithmus benötigt auf Listen der Länge  $n$  im average-case mindestens*

$$n \log n - n \log e + O(\log n) \approx n \log n - 1.44n \quad \text{Vergleiche.}$$

*Kommentar 20.* Das vorige Theorem bringt eine fundamentale *untere Schranke* der Komplexität des Sortierens zum Ausdruck: kein vergleichsbasiertes Sortierverfahren kann asymptotisch besser sein als  $n \log n$  Vergleiche (im *worst-case* und im *average-case*) auf Listen der Länge  $n$ . Wegen des Auftretens der Entropiefunktion im Beweis von Theorem 66 spricht man auch von der *informationstheoretischen Schranke für die Sortierkomplexität*.

- Sortieralgorithmen, die  $\Theta(n \log n)$  Vergleiche für inputs der Länge  $n$  benötigen (im worst-case bzw. average-case), sind *asymptotisch optimal*.
- *Mergesort* und *Heapsort* sind asymptotisch optimale Sortierverfahren im worst-case (und daher auch im average-case).
- *Selectionsort* (*Bubblesort*) und *Insertionsort* sind weder im worst-case, noch im average-case, asymptotisch optimal.
- *Quicksort* ist nur im average-case ein asymptotisch optimales Sortierverfahren.

- Es gibt andere Sortieralgorithmen (z.B. *Bucketsort*) mit asymptotisch linearer Laufzeit: die sind aber nicht vergleichsbasiert oder machen einschränkende Annahmen über die Natur bzw. Verteilung der zu sortierenden Elemente.

### 6.3.3 Die “wahre” untere Schranke der Sortierkomplexität

Es bezeichne (nur für diesen Abschnitt)

- $S(n)$  : die wirkliche minimale Anzahl von Vergleichen zum Sortieren von Listen der Länge  $n$  im worst case. Wir haben gesehen

$$S(n) \geq \lceil \log n! \rceil,$$

aber das heisst nicht, dass  $S(n) \geq \lceil \log n! \rceil$  ist!

- $M(n)$  : die minimale Anzahl von Vergleichen zum Sortieren von Listen der Länge  $n$  im worst case mittels *Mergesort*. Dafür wissen wir:

$$\begin{aligned} M(n) &= M(\lceil \frac{n}{2} \rceil) + M(\lfloor \frac{n}{2} \rfloor) + n - 1 \\ &= n \lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1 \end{aligned}$$

- $B(n)$  : die minimale Anzahl von Vergleichen zum Sortieren von Listen der Länge  $n$  im worst case mittels *Insertionsort* (mit binärem Einfügen). Dafür gilt:

$$\begin{aligned} B(n) &= \sum_{k=2}^n \lceil \log k \rceil \\ &= n \lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1 \end{aligned}$$

- $F(n)$  : die minimale Anzahl von Vergleichen zum Sortieren von Listen der Länge  $n$  im worst case mittels *merge insertion* (FORD-JOHNSON-Algorithmus, 1959)

$$\begin{aligned} F(n) &= \sum_{k=2}^n \lceil \log \frac{3}{4} k \rceil \\ &= n \lceil \log \frac{3}{4} n \rceil - \lfloor 2^{\lceil \log 6n \rceil} / 3 \rfloor + \frac{1}{2} \lceil \log 6n \rceil \end{aligned}$$

Die Werte für  $M(n)$ ,  $B(n)$ ,  $F(n)$  kann man für jedes  $n \in \mathbb{N}$  berechnen. Für  $S(n)$  sind die exakten Werte nur für  $n \leq 13$  bekannt. Für grössere  $n$  gibt es bislang nur Abschätzungen.

$n$	1	2	3	4	5	6	7	8	9	10	11	12	13
$M(n) = B(n)$	0	1	3	5	8	11	14	17	21	25	29	33	37
$F(n)$	0	1	3	5	7	10	13	16	19	22	26	30	34
$S(n)$	0	1	3	5	7	10	13	16	19	22	26	30*	34**
$\lceil \log n! \rceil$	0	1	3	5	7	10	13	16	19	22	26	29	33

\* : M. Wells, 1965; \*\* : M. Peczarski, 2002.

Es bleibt festzuhalten:

- *Mergesort* ist für  $n \geq 5$  nicht mehr (strikt) optimal.
- Für  $n = 12$  ist erstmals  $S(n) > \lceil \log n! \rceil$ .
- Die Tabelle suggeriert, dass der FORD-JOHNSON-Algorithmus im strikten Sinne optimal ist. Aber man weiss heute, dass das für grosse  $n$  nicht richtig ist.

Nähere Auskunft dazu findet man (wie zu erwarten) in Band 3 von KNUTHS TAOCP.

### 6.3.4 Entropie

$X$  sei eine Zufallsvariable, die endlich-viele Werte, z.B.  $\in \{1, 2, \dots, n\}$  oder  $\in \{0, 1, \dots, n-1\}$  annimmt. Mit  $p_k = P[X = k]$  wird die Wahrscheinlichkeit für Eintreten des Ereignisses “ $X = k$ ” notiert. Es gilt also (falls die Werte in  $\{1, 2, \dots, n\}$  liegen):

$$p_k \geq 0 \quad (1 \leq k \leq n) \quad \text{mit} \quad \sum_{k=1}^n p_k = 1.$$

Der Vektor  $\mathbf{p} = \langle p_1, p_2, \dots, p_n \rangle$  ist eine Wahrscheinlichkeitsverteilung auf der Menge  $\{1, 2, \dots, n\}$  oder einer anderen  $n$ -elementigen geordneten Menge.  $\mathcal{W}^{(n)}$  bezeichne die Menge dieser Wahrscheinlichkeitsverteilungen. Speziell bezeichne

- $\mathbf{u}^{(n)} = \langle \frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n} \rangle$  die Gleichverteilung (“uniform”) auf  $\{1, 2, \dots, n\}$ ;
- $\delta_k^{(n)} = \langle \delta_{1,k}, \delta_{2,k}, \dots, \delta_{n,k} \rangle$  die Verteilung mit einer Wahrscheinlichkeit = 1 im Punkt  $k$  und der Wahrscheinlichkeit = 0 in den übrigen Punkten  $1 \leq j \leq n, j \neq k$ .

**Definition 31.** Der *Informationsgehalt* oder kurz: die *Information* des Ereignisses “ $X = k$ ” ist

$$I[X = k] = -\log p_k.$$

Die *Entropie* von  $X$  (oder  $\mathbf{p}$ ) ist definiert als der *Erwartungswert der Information*:

$$H(\mathbf{p}) = H(p_1, \dots, p_n) = \mathbf{E}_{\mathbf{p}}(I) = -\sum_{k=1}^n p_k \log p_k.$$

Beachte: ein Ereignis mit  $p_k = 0$  liefert zu  $H(\mathbf{p})$  den Beitrag 0.

*Kommentar 21.* Anschaulich beschreibt  $H(\mathbf{p})$  den *mittleren Informationsgewinn* bei Durchführen eines Zufallsexperimentes, dessen Ausgang durch die Wahrscheinlichkeiten  $\mathbf{p}$  regiert wird. In dieser Betrachtung haben die meisten der Aussagen des folgenden Satzes einen guten Sinn und man kann sie als plausible Forderungen an einen solchen Informationsbegriff ansehen.

**Satz 69.** *Eigenschaften der Entropiefunktionen  $H(p_1, p_2, \dots, p_n)$  für  $n \geq 1$  sind:*

1.  $H(\mathbf{p}) \geq 0$
2.  $H(\mathbf{p}) = 0 \Leftrightarrow \mathbf{p} = \boldsymbol{\delta}_k^{(n)}$  für ein  $k \in \{1, \dots, n\}$ .
3.  $H(\mathbf{p}) \leq H(\mathbf{u}^{(n)})$ , d.h. die Gleichverteilung maximiert  $H(\mathbf{p})$ .
4.  $H(\mathbf{p})$  ist invariant unter Permutation der Komponenten  $p_1, p_2, \dots, p_n$ .
5.  $H(\mathbf{p}, 0) = H(\mathbf{p})$
6.  $H(\mathbf{u}^{(n)}) \leq H(\mathbf{u}^{(n+1)})$
7.  $H(\mathbf{p})$  ist eine stetige Funktion der Variablen  $p_1, p_2, \dots, p_n$ .
8.  $H(\mathbf{u}^{(mn)}) = H(\mathbf{u}^{(m)}) + H(\mathbf{u}^{(n)})$
9. Für  $\mathbf{p} \in \mathcal{W}^{(m)}$ ,  $\mathbf{q} \in \mathcal{W}^{(n)}$  und  $(\lambda, \mu) \in \mathcal{W}^{(2)}$  gilt

$$H(\lambda \mathbf{p}, \mu \mathbf{q}) = H(\lambda, \mu) + \lambda H(\mathbf{p}) + \mu H(\mathbf{q}).$$

Hierbei ist  $\langle \lambda \mathbf{p}, \mu \mathbf{q} \rangle = \langle \lambda p_1, \dots, \lambda p_m, \mu q_1, \dots, \mu q_n \rangle$  als ein Element von  $\mathcal{W}^{(m+n)}$  aufzufassen.

Fast alle diese Eigenschaften sind offensichtlich oder leicht nachzurechnen. Ausnahmen bilden lediglich die Punkte 3. und 9. Zunächst zu Punkt 3.

Folgende Aussage bezeichnet man in der Informationstheorie als “key lemma” oder als “Ungleichung von GIBBS”:

**Satz 70.** Für  $\mathbf{p} = (p_1, \dots, p_n) \in \mathcal{W}^{(n)}$  und  $\mathbf{q} = (q_1, \dots, q_n) \in \mathcal{W}^{(n)}$  gilt

$$H(p_1, \dots, p_n) \leq - \sum_{k=1}^n p_k \cdot \log q_k,$$

und es gilt „=“ genau dann, wenn  $\mathbf{p} = \mathbf{q}$  ist.

Als Konsequenz hat man, indem man für  $\mathbf{q}$  die Gleichverteilung  $\mathbf{u}^{(n)}$  mit wählt:

**Folgerung 71.** Für alle  $\mathbf{p} \in \mathcal{W}^{(n)}$  gilt

$$H(\mathbf{p}) \leq \log n,$$

und Gleichheit gilt genau im Fall  $\mathbf{p} = \mathbf{u}^{(n)}$ .

*Beweis.* (“key lemma”)

Die Logarithmusfunktion ist konvex und es gilt daher

$$\ln x \leq x - 1 \quad (x > 0)$$

mit Gleichheit genau dann, wenn  $x = 1$ . Also ist

$$\ln \frac{q_k}{p_k} \leq \frac{q_k}{p_k} - 1 \quad (1 \leq k \leq n).$$

Summiert man diese Ungleichungen, so erhält man

$$\sum_{1 \leq k \leq n} p_k \ln \frac{q_k}{p_k} \leq \sum_{1 \leq k \leq n} q_k - \sum_{1 \leq k \leq n} p_k = 0$$

und das ist gleichwertig zu

$$\sum_{1 \leq k \leq n} p_k \ln q_k \leq \sum_{1 \leq k \leq n} p_k \ln p_k,$$

mit Gleichheit genau dann, wenn  $\mathbf{p} = \mathbf{q}$ . □

*Bemerkung 22.* Die Aussage des “key lemmas” hat eine wichtige Konsequenz: man kann auf der Menge  $\mathcal{W}^{(n)}$  eine Abstandsmessung einführen: für  $\mathbf{p} = (p_1, \dots, p_n)$  und  $\mathbf{q} = (q_1, \dots, q_n) \in \mathcal{W}^{(n)}$  definiert man

$$D(\mathbf{p}||\mathbf{q}) = \sum_{1 \leq k \leq n} p_k \cdot \log \frac{p_k}{q_k},$$

genannt *Divergenz* oder *KULLBACK-LEIBLER-Distanz* von  $\mathbf{p}$  und  $\mathbf{q}$ . Diese Größe ist ein Maß dafür, wie sehr sich  $\mathbf{p}$  und  $\mathbf{q}$  unterscheiden. Das “key lemma” kann man auch so formulieren:

$$D(\mathbf{p}||\mathbf{q}) \geq 0 \quad \text{und “} = 0 \text{” gilt genau dann, wenn } \mathbf{p} = \mathbf{q}.$$

Vorsicht ist aber geboten: dieses Abstandsmaß ist keine *Metrik* im üblichen Sinn, denn es ist nicht symmetrisch, d.h. i.a. ist

$$D(\mathbf{p}||\mathbf{q}) \neq D(\mathbf{q}||\mathbf{p}).$$

Die KULLBACK-LEIBLER-Distanz spielt eine grosse Rolle in Anwendungen, wie z.B. in der Informationstheorie, der statistischen Lerntheorie und der Mustererkennung.  $\square$

Nun zu Punkt 9. Das kann man nachrechnen:

$$\begin{aligned} -H(\lambda \mathbf{p}, \mu \mathbf{q}) &= \sum_{1 \leq i \leq m} \lambda p_i \log(\lambda p_i) + \sum_{1 \leq j \leq n} \mu q_j \log(\mu q_j) \\ &= \sum_{1 \leq i \leq m} \lambda p_i (\log \lambda + \log p_i) + \sum_{1 \leq j \leq n} \mu q_j (\log \mu + \log q_j) \\ &= \lambda \log \lambda + \mu \log \mu + \lambda \sum_{1 \leq i \leq m} p_i \log p_i + \mu \sum_{1 \leq j \leq n} q_j \log q_j \\ &= -H(\lambda, \mu) - \lambda H(\mathbf{p}) - \mu H(\mathbf{q}). \quad \square \end{aligned}$$

*Bemerkung 23.* Die Aussage von Punkt 9. kann man ohne Schwierigkeiten verallgemeinern. Dazu

- sei  $(A_i)_{i \in I}$  eine Familie von paarweise disjunkten (endlichen) Mengen;
- bezeichne  $A = \bigoplus_{i \in I} A_i$  die disjunkte Vereinigung der  $A_i$ ;
- sei  $\boldsymbol{\rho} = (\rho_i)_{i \in I}$  eine Familie von Wahrscheinlichkeitsverteilungen, wobei  $\rho_i$  auf der Menge  $A_i$  definiert ist.
- sei  $\boldsymbol{\lambda} = (\lambda_i)_{i \in I}$  eine Wahrscheinlichkeitsverteilung auf  $I$ .

Auf ganz natürliche Weise ist dann

$$\boldsymbol{\lambda} \circ \boldsymbol{\rho} = \bigoplus_{i \in I} \lambda_i \cdot \rho_i$$

eine Wahrscheinlichkeitsverteilung auf  $A$ : für alle  $i \in I$  und  $a \in A_i$  ist

$$(\boldsymbol{\lambda} \circ \boldsymbol{\rho})(a_i) = \lambda_i \cdot \rho_i(a).$$

Für die Entropie dieser Wahrscheinlichkeitsverteilung gilt:

$$H[\lambda \circ \rho] = H[\lambda] + E_\lambda[\rho],$$

wobei

$$E_\lambda[\rho] = \sum_{i \in I} \lambda_i H[\rho_i]$$

ist. □

Ein wichtiger und prominenter Spezialfall hiervon ist folgende Aussage:

**Satz 72.** *Sind  $X, Y$  zwei Zufallsvariable, so gilt*

$$H[X, Y] = H[X] + H[Y | X].$$

*Hierbei ist  $H[X, Y]$  die Entropie der gemeinsamen Verteilung von  $X$  und  $Y$  und  $H[Y | X]$  ist die "bedingte Entropie" von  $Y$ , gegeben  $X$ . □*

Die Aussagen von Satz 69 sind nicht nur von praktischem Interesse, sondern sie dienen auch dazu, die SHANNONSche Definition der Entropie zu rechtfertigen. Wenn man sie für plausible Forderungen an einen quantitativen Entropiebegriff akzeptiert, hat man keine Wahl, denn es gilt:

**Theorem 73 (SHANNON).** *Die Eigenschaften 1.-9. von Satz 69 bestimmen die Entropiefunktion eindeutig (bis auf einen konstanten Faktor), d.h. es gilt:*

*Eine Funktion mit den Eigenschaften 1.-9. ist notwendig von der Form*

$$H(p_1, \dots, p_n) = -c \sum_{k=1}^n p_k \log p_k$$

*mit einer Konstanten  $c > 0$ .*

Tatsächlich sind diese neun Eigenschaften nicht unabhängig voneinander und man kommt bereits mit wenigen der Voraussetzungen zur gleichen Folgerung. Genauer gesagt gilt: die Eigenschaften 3, 5, 7, und 9 von Satz 69 implizieren die übrigen Eigenschaften.

### 6.3.5 Binäre Bäume mit Gewichten (auf den Blättern)

Wir betrachten nun Paare  $\langle t, \mathbf{p} \rangle$ , wobei  $t \in \mathcal{B}$  ein binärer Baum ist und  $\mathbf{p} = (p_b)_{b \in E(t)}$  eine Wahrscheinlichkeitsverteilung auf den Blättern von  $t$ . Es interessiert

nun die mittlere Höhe der Blätter im Baum  $t$ , wobei aber auf den Blättern nicht mehr die Gleichverteilung gegeben ist, sondern eben die Wahrscheinlichkeitsverteilung  $\mathbf{p} = (p_b)_{b \in E(t)}$ .

**Definition 32.** Für einen gewichteten Baum  $\langle t, \mathbf{p} \rangle$  ist

$$\bar{h}(t, \mathbf{p}) = \sum_{b \in E(t)} p_b \cdot h(b, t)$$

die *gewichtete mittlere Höhe* oder auch *gewichtete externe Pfadlänge*.

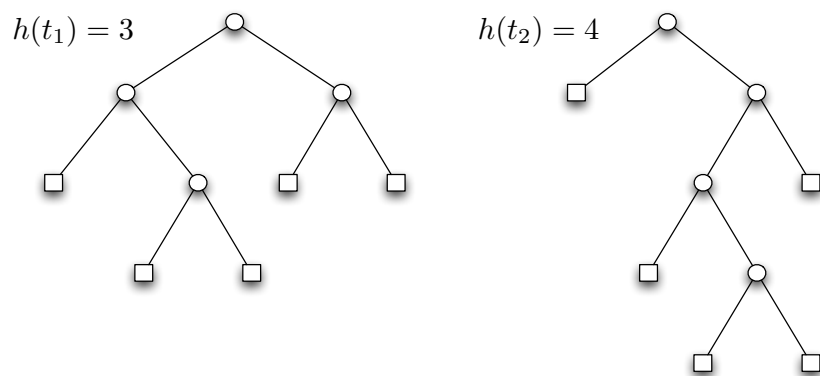
*Bemerkung 24.* Für den Fall der Gleichverteilung auf  $E(t)$  ist das gerade  $\bar{h}(t)$ .

Zur Erinnerung: im Zusammenhang mit der Sortierkomplexität spielte die Ungleichung (“information theory bound”)

$$h(t) \geq \log e(t)$$

die entscheidende Rolle.

*Beispiel 19.* Beispiele für mittlere Höhe und gewichtete mittlere Höhe in den Abbildungen 32, 33 und 34.

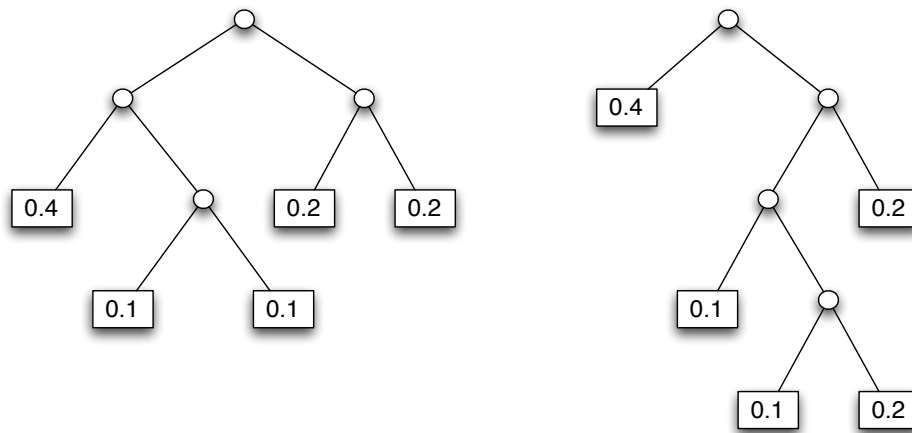


$$\bar{h}(t_1) = \frac{1}{5}(2 + 3 + 3 + 2 + 2) = \frac{12}{5} \quad \bar{h}(t_2) = \frac{1}{5}(1 + 3 + 4 + 4 + 2) = \frac{14}{5}$$

Abbildung 32: Mittlere Höhe

Ein fundamentales Problem der Informationstheorie ist nun die Frage:

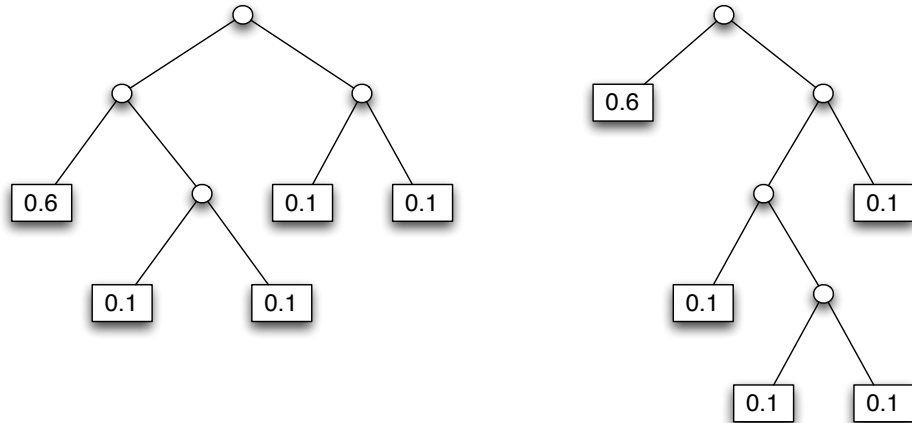
- Wie klein kann  $\bar{h}(t, \mathbf{p})$  bei gegebenem  $\mathbf{p}$  gemacht werden, indem man den Binärbaum  $t$  geeignet wählt?



$$h(t_1, p) = 0.4 \cdot 2 + 0.1 \cdot 3 + 0.1 \cdot 3 + 0.2 \cdot 2 + 0.2 \cdot 2 = 2.2$$

$$h(t_2, p) = 0.4 \cdot 1 + 0.1 \cdot 3 + 0.1 \cdot 4 + 0.2 \cdot 4 + 0.2 \cdot 2 = 2.3$$

Abbildung 33: Gewichtete mittlere Höhe



$$h(t_1, q) = 0.6 \cdot 2 + 0.1 \cdot 3 + 0.1 \cdot 3 + 0.1 \cdot 2 + 0.1 \cdot 2 = 2.2$$

$$h(t_2, q) = 0.6 \cdot 1 + 0.1 \cdot 3 + 0.1 \cdot 4 + 0.1 \cdot 4 + 0.1 \cdot 2 = 1.9$$

Abbildung 34: Gewichtete mittlere Höhe

Die Antwort darauf gibt ein Resultat von SHANNON, das neben seinem berühmten Kanalcodierungstheorem ein Grundpfeiler der Informationstheorie ist.

**Theorem 74** (SHANNONS Quellcodierungstheorem). *Für jede Wahrscheinlichkeitsverteilung  $\mathbf{p} \in \mathcal{W}^{(n)}$  ist*

- Untere Schranke: Für jeden Binärbaum  $t \in \mathbb{B}_{n-1}$  gilt

$$H(\mathbf{p}) \leq \bar{h}(t, \mathbf{p}).$$

- Obere Schranke: Es gibt einen Binärbaum  $t \in \mathbb{B}_{n-1}$  mit

$$\bar{h}(t, \mathbf{p}) < H(\mathbf{p}) + 1.$$

Insgesamt gilt also die Ungleichung

$$H(\mathbf{p}) \leq \bar{h}(t, \mathbf{p}) < H(\mathbf{p}) + 1,$$

wobei die linke Ungleichung eine *universelle* Aussage ist und die rechte Ungleichung eine *existentielle*!

*Beweis.* Der Beweis für den ersten Teil (untere Schranke für  $\bar{h}(t, \mathbf{p})$ ) wird hier geführt. Er verläuft genauso wie der Beweis für  $\log e(t) \leq \bar{h}(t)$  im Fall der Gleichverteilung, siehe Abschnitt 6.3.1. Der Beweis der oberen Schranke folgt im Abschnitt 6.4.3.

Sei  $t = \langle \bigcirc, t_\ell, t_r \rangle$  Binärbaum mit Knotengewichten

- $(p_1, \dots, p_m)$  auf  $E(t_\ell) = (b_1, \dots, b_m)$
- $(q_1, \dots, q_n)$  auf  $E(t_r) = (c_1, \dots, c_n)$

Dabei sind

- $\mathbf{p} = (p_1, \dots, p_m, q_1, \dots, q_n)$
- $\mathbf{p}_\ell = (p_1/p, \dots, p_m/p)$  mit  $p = p_1 + \dots + p_m$
- $\mathbf{p}_r = (q_1/q, \dots, q_n/q)$  mit  $q = q_1 + \dots + q_n$

Wahrscheinlichkeitsverteilungen auf  $E(t)$  bzw.  $E(t_\ell)$  bzw.  $E(t_r)$ .

Dann gilt (Induktion!)

$$\begin{aligned}
 \bar{h}(t, \mathbf{p}) &= \sum_{b \in E(t)} p_b \cdot h(b, t) \\
 &= \sum_{b \in E(t_\ell)} p_b \cdot h(b, t) + \sum_{c \in E(t_r)} p_c \cdot h(c, t) \\
 &= \sum_{1 \leq i \leq m} p_i \cdot (h(b_i, t_\ell) + 1) + \sum_{1 \leq j \leq n} q_j \cdot (h(c_j, t_r) + 1) \\
 &= p + q + p \cdot \bar{h}(t_\ell, \mathbf{p}_\ell) + q \cdot \bar{h}(t_r, \mathbf{p}_r) \\
 &\geq 1 + p \cdot H(\mathbf{p}_\ell) + q \cdot H(\mathbf{p}_r) \\
 &= \underbrace{1 - H(p, q)}_{\geq 0} + H(\mathbf{p}) \geq H(\mathbf{p})
 \end{aligned}$$

wegen der Eigenschaften 3. und 9. der Entropiefunktion. □

## 6.4 Quellcodierung

### 6.4.1 Codes variabler Länge

Es sei nun  $A = \{a, b, c, \dots\}$  eine endliche Menge von “Nachrichten” (Quellalphabet) und  $\mathbb{B} = \{0, 1\}$  das “Kanalalphabet”. Nachrichten sollen binär, also durch Wörter über dem Alphabet  $\mathbb{B}$  codiert werden.

**Definition 33.** Eine (binäre) *Codierung* ist injektive Abbildung

$$\Phi : A \rightarrow \mathbb{B}^+$$

falls für die Fortsetzung zu einem Homomorphismus

$$\Phi : A^* \rightarrow \mathbb{B}^* : a_1 a_2 \dots a_n \mapsto \Phi(a_1) \Phi(a_2) \dots \Phi(a_n)$$

die Decodierbedingung (*unique decipherability*)

$$(UD) \quad \text{für jedes } w \in \mathbb{B}^* \text{ gilt } \#\Phi^{-1}(w) \leq 1$$

erfüllt ist. Die Menge  $\Phi(A) = \{\Phi(a), \Phi(b), \Phi(c), \dots\}$  der Codewörter wird dann auch als *Code* bezeichnet.

*Beispiel 20.* In den folgenden Beispielen ist  $A = \{a, b, c, d\}$ .

$$\Phi_1 : \begin{cases} a \mapsto 00 \\ b \mapsto 01 \\ c \mapsto 10 \\ d \mapsto 11 \end{cases} \quad \Phi_2 : \begin{cases} a \mapsto 0 \\ b \mapsto 111 \\ c \mapsto 110 \\ d \mapsto 101 \end{cases} \quad \Phi_3 : \begin{cases} a \mapsto 01 \\ b \mapsto 011 \\ c \mapsto 110 \\ d \mapsto 101 \end{cases}$$

1.  $\Phi_1$  ist Codierung mit konstanter Länge (“Blockcode”)

$$\Phi_1(abadc) = 00 \cdot 01 \cdot 00 \cdot 11 \cdot 10 = 0001001110$$

2.  $\Phi_2$  ist Codierung mit variabler Länge

$$\Phi_2(abadc) = 0 \cdot 111 \cdot 0 \cdot 101 \cdot 110 = 01110101110$$

3.  $\Phi_3$  ist keine Codierung!

$$\Phi_3(bda) = 011 \cdot 101 \cdot 01 = 01110101 = 01 \cdot 110 \cdot 101 = \Phi_3(acd)$$

**Definition 34.** Eine Codierung  $\Phi$  (ein Code  $\Phi(A)$ ) hat die *Präfix-Eigenschaft* (oder: ist ein *Präfixcode*), wenn gilt:

(PP) kein Codewort ist Präfix eines anderen Codewortes

*Beispiel 21.* 1.  $\Phi_1$  (und allgemein alle Codes konstanter Länge) sowie  $\Phi_2$  haben die (PE), für die Abbildung  $\Phi_3$  gilt (PE) nicht.

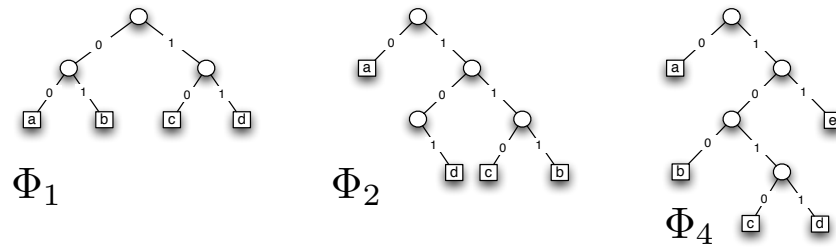
2. Es gilt offensichtlich: (PP)  $\Rightarrow$  (UD), aber (UD)  $\not\Rightarrow$  (PP).

3. Zwei Beispiele mit  $A = \{a, b, c, d, e\}$ :

$$\Phi_4 : \begin{cases} a \mapsto 0 \\ b \mapsto 100 \\ c \mapsto 1010 \\ d \mapsto 1011 \\ e \mapsto 11 \end{cases} \quad \Phi_5 : \begin{cases} a \mapsto 00 \\ b \mapsto 101 \\ c \mapsto 010 \\ d \mapsto 001 \\ e \mapsto 11 \end{cases}$$

$\Phi_4$  hat (PP),  $\Phi_5$  hat (UD), aber nicht (PP).

*Bemerkung 25.* Präfixcodes sind binäre Bäume in einem erweiterten Sinn: innere Knoten können einen (rechten oder linken) oder zwei (rechten und linken) Nachfolger haben.



### 6.4.2 Quellen und Datenkompression

Vergleiche die Codierungen  $\Phi_4$  und  $\Phi_5$ :

$$\begin{array}{rcc}
 & \Phi_4 & \Phi_5 \\
 aaea & \mapsto & 000110 \\
 abaedbc & \mapsto & 0100011101110101011 \quad 0010001100101010
 \end{array}$$

Welche Codierung ist “besser”?

Das hängt davon ab, mit welchen Wahrscheinlichkeiten die “Quellsymbole”  $a, b, c, d, e$  vorkommen!

**Definition 35.** Eine *Quelle*  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  ist ein Paar, bestehend aus

- einem *Quellalphabet*  $A = \{a, b, c, \dots\}$
- einer Wahrscheinlichkeitsverteilung  $\mathbf{p} = (p_a, p_b, p_c, \dots)$  auf  $A$ .

Für Codierungen  $\Phi : A \rightarrow \{0, 1\}^+$  einer Quelle  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  ist

$$\mu(\mathcal{Q}, \Phi) = \sum_{x \in A} p_x \cdot |\Phi(x)|$$

*mittlere* oder *erwartete* Codewortlänge des Codes  $\mathcal{C} = \Phi(A)$ .

*Bemerkung 26.* Für Präfixcodes gilt:

$$\text{mittlere Codewortlänge} \equiv \begin{cases} \text{gewichtete mittlere Höhe} \\ \text{des entsprechenden Binärbaumes} \\ \text{(im erweiterten Sinn)} \end{cases}$$

Diese Grösse ist ein Maß dafür, welche *Kompression* bei Codierung mit diesem Code erreicht wird.

*Beispiel 22.* 1.  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  mit  $A = \{a, b, c, d\}$  und

$$\mathbf{p} = (p_a, p_b, p_c, p_d) = (0.9, 0.05, 0.025, 0.025)$$

$$- \Phi_1 : a \mapsto 00, b \mapsto 01, c \mapsto 10, d \mapsto 11$$

$$\mu(\mathcal{Q}, \Phi_1) = 0.9 \cdot 2 + 0.05 \cdot 2 + 0.025 \cdot 2 + 0.025 \cdot 2 = 2$$

$$- \Phi_2 : a \mapsto 0, b \mapsto 111, c \mapsto 110, d \mapsto 101$$

$$\mu(\mathcal{Q}, \Phi_2) = 0.9 \cdot 1 + 0.05 \cdot 3 + 0.025 \cdot 3 + 0.025 \cdot 3 = 1.2$$

2.  $\mathcal{Q} = \langle A, \mathbf{q} \rangle$  mit  $A = \{a, b, c, d\}$  und

$$\mathbf{q} = (q_a, q_b, q_c, q_d) = (0.35, 0.25, 0.25, 0.15)$$

$$- \Phi_1 : a \mapsto 00, b \mapsto 01, c \mapsto 10, d \mapsto 11$$

$$\mu(\mathcal{Q}, \Phi_1) = 0.35 \cdot 2 + 0.25 \cdot 2 + 0.25 \cdot 2 + 0.15 \cdot 2 = 2$$

$$- \Phi_2 : a \mapsto 0, b \mapsto 111, c \mapsto 110, d \mapsto 101$$

$$\mu(\mathcal{Q}, \Phi_2) = 0.35 \cdot 1 + 0.25 \cdot 3 + 0.25 \cdot 3 + 0.15 \cdot 3 = 2.3$$

**Folgerung 75** (SHANNON). *Bei gegebener Quellverteilung  $\mathbf{p} = (p_a, p_b, p_c, \dots)$  gibt die Entropie  $H(\mathbf{p})$  ein Maß dafür an, welche Kompression (mittlere gewichtete Wortlänge) bei keiner Codierung unterschritten werden kann.*

*Bemerkung 27.* 1. Für optimale Codierung kann man sich auf die Verwendung von (echten) Binärbäumen beschränken.

2. Präfixcodes lassen sich “online” decodieren, bei Codes ohne (PP) geht das i.a. nicht.

3. Verwendung von (UD)-Codes, die nicht die (PP) haben, kann die Situation bezüglich Kompression nicht verbessern, denn es gilt der Satz von MACMILLAN:

Zu jeder Quelle  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  und zu jedem (UD)-Code  $\Phi(A)$  gibt es einen (PP)-Code  $\Psi(A)$  mit  $\mu(\mathcal{Q}, \Phi) = \mu(\mathcal{Q}, \Psi)$ .

4. Die Konstruktion optimaler Präfixcodes leistet ein Verfahren von HUFFMAN, siehe Abschnitt 6.4.4.

### 6.4.3 Ungleichung von Kraft

Die Frage nach der *Existenz* von Präfix-Codes mit gegebenen Wortlängen  $\ell_1, \ell_2, \dots, \ell_n$  wird durch folgendes Resultat beantwortet:

**Theorem 76** (Ungleichung von KRAFT). *Notwendig und hinreichend für die Existenz eines binären Präfixcodes  $\{w_1, w_2, \dots, w_n\} \subset \mathbb{B}^*$  mit Wortlängen  $|w_k| = \ell_k$  ( $1 \leq k \leq n$ ) ist*

$$\sum_{1 \leq k \leq n} 2^{-\ell_k} \leq 1.$$

Beachte: ist  $t$  ein Binärbaum (im ursprünglichen Sinn), so gilt immer

$$\sum_{b \in E(t)} 2^{-h(b,t)} = 1.$$

*Beweis.* 1. Die Bedingung ist *notwendig*, denn existiert ein solcher Präfixcode und ist  $\ell_n = \max_{1 \leq k \leq n} \ell_k$ , so sind die Wortmengen

$$w_k \cdot \mathbb{B}^{\ell_n - \ell_k} \subseteq \mathbb{B}^{\ell_n} \quad (1 \leq k \leq n)$$

wegen (PP) paarweise disjunkt, und deshalb gilt

$$\sum_{1 \leq k \leq n} 2^{\ell_n - \ell_k} \leq 2^{\ell_n}.$$

2. Die Bedingung ist *hinreichend*: Dies wird mittels Induktion über  $n$  bewiesen

- Für  $n = 1$  bzw.  $n = 2$  leisten  $\{0^{\ell_1}\}$  bzw.  $\{0^{\ell_1}, 1^{\ell_2}\}$  das Gewünschte
- Sei die Behauptung für  $n > 1$  bewiesen und genügen die  $1 \leq \ell_1 \leq \dots \leq \ell_n \leq \ell_{n+1}$  der Ungleichung von KRAFT. Die gilt die Ungleichung von KRAFT auch für die  $n + 1$  Zahlen

$$\ell_1, \ell_2, \dots, \ell_{n-1}, \ell_n, \ell_n$$

und wegen  $2^{-\ell_n} + 2^{-\ell_n} = 2^{-(\ell_n - 1)}$  auch für die  $n$  Zahlen

$$\ell_1, \ell_2, \dots, \ell_{n-1}, \ell_n - 1.$$

Es existiert also ein Präfixcode

$$\{w_1, w_2, \dots, w_n\} \text{ mit } |w_k| = \ell_k \text{ (} 1 \leq k < n \text{) und } |w_n| = \ell_n - 1.$$

Der Präfixcode

$$\{w_1, w_2, \dots, w_{n-1}, w_n 0, w_n 1^{\ell_{n+1} - \ell_n + 1}\}$$

leistet das Verlangte.  $\square$

*Beispiel 23.* Zur Ungleichung von KRAFT

Wortlängen	Code
2, 3, 3, 3, 5, 7	{00, 010, 110, 111, 01110, 0111111}
2, 3, 3, 3, 4	{00, 010, 110, 111, 0111}
2, 2, 3, 3	{00, 01, 110, 111}
2, 2, 2	{00, 01, 11}
1, 2	{0, 11}
0	{ $\epsilon$ }

*Beweis.* Oberere Schranke von SHANNONS Quellcodierungstheorem

$\mathcal{Q} = \langle A, \mathbf{p} \rangle$  sei eine Quelle mit Alphabet  $A = \{a_1, \dots, a_n\}$  und Wahrscheinlichkeiten  $\mathbf{p} = (p_1, p_2, \dots, p_n)$ ,

wobei o.E.  $p_k > 0$  ( $1 \leq k \leq n$ ) sein soll. Mit

$$\ell_k = \lceil -\log p_k \rceil \quad (1 \leq k \leq n)$$

gilt

$$-\log p_k \leq \ell_k < 1 - \log p_k \quad (1 \leq k \leq n)$$

und somit folgt aus der linken Ungleichung

$$\sum_{1 \leq k \leq n} 2^{-\ell_k} \leq 1.$$

Wegen Theorem 76 existiert also ein Präfixcode

$$\Phi : A \rightarrow \{w_1, w_2, \dots, w_n\} : a_i \mapsto w_i \quad (1 \leq i \leq n)$$

mit den Wortlängen

$$|w_k| = \ell_k = 2^{\lceil -\log p_k \rceil} \quad (1 \leq k \leq n).$$

Eine Abschätzung für die mittlere Wortlänge dieses Codes ergibt sich nun aus der rechten Ungleichung

$$\mu(\mathcal{Q}, \Phi) = \sum_{1 \leq k \leq n} p_k \cdot \ell_k < \sum_{1 \leq k \leq n} p_k \cdot (1 - \log p_k) = 1 + H(\mathbf{p}). \quad \square$$

#### 6.4.4 HUFFMANS Konstruktion optimaler (Präfix-)Codes

Eine *optimale* Codierung  $\Phi$  für eine Quelle  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  liegt vor, wenn

$$\mu(\mathcal{Q}, \Phi) = \min\{\mu(\mathcal{Q}, \Psi) ; \Psi \text{ Codierung für } \mathcal{Q}\}$$

gilt. Das Theorem von SHANNON garantiert für optimales  $\Phi$

$$H(\mathbf{p}) \leq \mu(\mathcal{Q}, \Phi) < 1 + H(\mathbf{p})$$

Die Konstruktion eines optimalen  $\Phi$  kann mit Hilfe eines "GREEDY"-Algorithmus ausgeführt werden, der sich am Beweis der Ungleichung von KRAFT orientiert.

**Lemma 77.** *Ist  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  ein Quelle mit  $\#A \geq 2$  und  $\Phi$  eine optimale Codierung für  $\mathcal{Q}$ , so gilt:*

$$\text{Für } a, b \in A \text{ mit } p_a > p_b \text{ ist } |\Phi(a)| \leq |\Phi(b)|.$$

*Beweis.* Gibt es  $a, b \in A$  mit  $p_a > p_b$  und  $|\Phi(a)| > |\Phi(b)|$ , so kann man die Codierungen von  $a$  und  $b$  vertauschen:

$$\Psi : \begin{cases} a \mapsto \Phi(b) \\ b \mapsto \Phi(a) \\ c \mapsto \Phi(c) \quad (c \in A \setminus \{a, b\}) \end{cases}$$

und damit die mittlere Codewortlänge verkleinern:

$$\begin{aligned} \mu(\mathcal{Q}, \Psi) &= \mu(\mathcal{Q}, \Phi) - p_a \cdot (|\Phi(a)| - |\Psi(a)|) - p_b \cdot (|\Phi(b)| - |\Psi(b)|) \\ &= \mu(\mathcal{Q}, \Phi) - \underbrace{(p_a - p_b) \cdot (|\Phi(a)| - |\Phi(b)|)}_{>0} \end{aligned}$$

□

**Lemma 78.** *Ist  $\Phi$  optimale Codierung für die Codierung für eine Quelle  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$ , so ist der Code  $\Phi(A)$  ein Binärbaum (im strikten Sinn).*

**Lemma 79.** *Ist  $\Phi$  optimale Codierung für  $\mathcal{Q}$ , so kann man annehmen (d.h., durch Umordnung erreichen), dass es Symbole  $a, b \in A$  mit minimalen Wahrscheinlichkeiten  $p_a, p_b$  gibt, d.h.*

$$p_a, p_b \leq \min_{c \in A \setminus \{a, b\}} p_c,$$

die als Geschwister codiert sind, d.h. es gibt ein  $w \in \{0, 1\}^*$  mit

$$\Phi(a) = w \cdot 0 \quad \text{und} \quad \Phi(b) = w \cdot 1.$$

*Beweis.* Knoten auf dem höchsten Niveau eines Binärbaumes (im engeren Sinne) treten immer als Geschwisterpaare auf.

Durch Umordnung der Wahrscheinlichkeiten auf dem höchsten Niveau kann man die angegebene Situation erreichen, ohne die mittlere Wortlänge zu ändern.  $\square$

**Definition 36.** Ist  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  eine Quelle mit  $\sharp A \geq 2$  und sind  $a, b \in A$  zwei Quellsymbole, so wir die durch *Fusion* der Quellsymbolen  $A$  und  $b$  entstehende Quelle  $\mathcal{Q}' = \langle A', \mathbf{p}' \rangle$  definiert durch

$$\begin{aligned} \mathcal{Q} &= \langle A, \mathbf{p} \rangle \\ A' &= (A \setminus \{a, b\}) \cup \{\alpha\} \\ p'_x &= \begin{cases} p_x & \text{für } x \in A \setminus \{a, b\} \\ p_a + p_b & \text{für } x = \alpha \end{cases} \end{aligned}$$

**Satz 80.** Mit den Bezeichnungen wie in der vorigen Definition gilt

1. Ist  $\Phi$  (striker) Präfixcode für  $\mathcal{Q}$ , bei dem  $\Phi(a)$  und  $\Phi(b)$  Geschwister sind, d.h.  $\Phi(a) = w \cdot 0$ ,  $\Phi(b) = w \cdot 1$  für ein  $w \in \{0, 1\}^*$ , und ist  $\Phi'$  definiert durch

$$\Phi' : A \rightarrow \{0, 1\}^+ : \begin{cases} x \mapsto \Phi(x) & \text{für } x \in A \setminus \{a, b\}, \\ \alpha \mapsto w, \end{cases}$$

so ist  $\Rightarrow \Phi'$  ein (striker) Präfixcode für  $\mathcal{Q}'$ .

2. Ist  $\Phi'$  ein (striker) Präfixcode für  $\mathcal{Q}'$  und ist  $\Phi$  definiert durch

$$\Phi : A \rightarrow \{0, 1\}^+ : \begin{cases} x \mapsto \Phi'(x) & \text{für } x \in A \setminus \{a, b\}, \\ a \mapsto \Phi'(\alpha) \cdot 0, \\ b \mapsto \Phi'(\alpha) \cdot 1, \end{cases}$$

so ist  $\Rightarrow \Phi$  ein (striker) Präfixcode für  $\mathcal{Q}$ , bei dem  $\Phi(a)$  und  $\Phi(b)$  Geschwister sind.

3. Für die mittleren Codewortlängen gilt dabei

$$\begin{aligned} \mu(\mathcal{Q}, \Phi) - \mu(\mathcal{Q}', \Phi') &= p_a \cdot |\Phi(a)| + p_b \cdot |\Phi(b)| - p'_\alpha \cdot |\Phi'(\alpha)| \\ &= p_a \cdot |\Phi(a)| + p_b \cdot |\Phi(b)| - (p_a + p_b) \cdot (|\Phi(a)| - 1) \\ &= p_a + p_b = p'_\alpha. \end{aligned}$$

**Folgerung 81.** Entsteht die Quelle  $\mathcal{Q}' = \langle A', \mathbf{p}' \rangle$  aus der Quelle  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  durch *Fusion* zweier Symbole  $a, b \in A$  mit minimalen Wahrscheinlichkeiten  $p_a, p_b$ , so gilt (mit dem obigen Zusammenhang zwischen  $\Phi$  und  $\Phi'$ ):

$$\Phi \text{ optimal für } \mathcal{Q} \Leftrightarrow \Phi' \text{ optimal für } \mathcal{Q}'$$

Diese Aussage erlaubt die rekursive Konstruktion optimaler Präfixcodes.

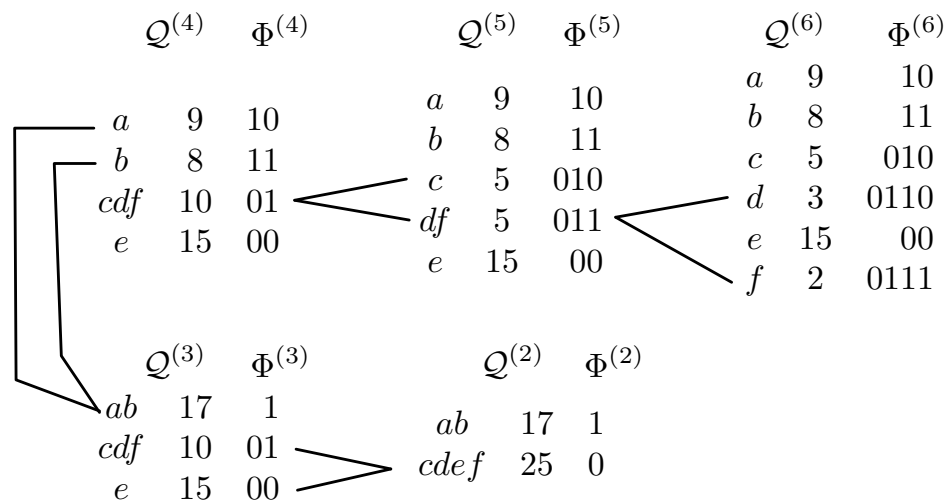
Die Realisierung dieser Idee – die HUFFMAN-Konstruktion – kann schematisch so dargestellt werden:

$$\begin{aligned} \mathcal{Q}^{(2)} &\leftarrow \mathcal{Q}^{(3)} \leftarrow \dots \leftarrow \mathcal{Q}^{(n-1)} \leftarrow \mathcal{Q}^{(n)} = \mathcal{Q} \\ \Phi^{(2)} &\rightarrow \Phi^{(3)} \rightarrow \dots \rightarrow \Phi^{(n-1)} \rightarrow \Phi^{(n)} = \Phi \end{aligned}$$

Dabei ist

- $\mathcal{Q}^{(k)}$  : Quelle mit  $k$  Symbolen
- $\Phi^{(k)}$  : optimaler Präfixcode für  $\mathcal{Q}^{(k)}$
- $\mathcal{Q}^{(k-1)} \leftarrow \mathcal{Q}^{(k)}$  : Fusion von zwei Symbolen mit minimaler Wahrscheinlichkeit
- $\Phi^{(k-1)} \rightarrow \Phi^{(k)}$  : Konstruktion entlang Umkehrung der Fusion
- $\mathcal{Q}^{(2)} = \langle \{a, b\}, (p_a, p_b) \rangle$
- $\Phi^{(2)} = \langle a \mapsto 0, b \mapsto 1 \rangle$

*Beispiel 24.* HUFFMAN-Konstruktion (mit Symbolhäufigkeiten an Stelle Wahrscheinlichkeiten)



$$\begin{aligned} \mu(\mathcal{Q}^{(6)}, \Phi^{(6)}) &= \frac{9 \cdot 2 + 8 \cdot 2 + 5 \cdot 3 + 3 \cdot 4 + 15 \cdot 2 + 2 \cdot 4}{42} \\ &= \frac{5 + 10 + 17 + 25}{42} + 1 = \frac{99}{42} = 2.35714284 \dots \end{aligned}$$

Beachte:

$$H\left(\frac{9}{42}, \frac{8}{42}, \frac{5}{42}, \frac{3}{42}, \frac{15}{42}, \frac{2}{42}\right) = 2.309050472\dots$$

### 6.4.5 Bottom-up Implementierung der HUFFMAN-Konstruktion

Gegeben ist die Quelle  $\mathcal{Q} = \langle A, \mathbf{p} \rangle$  mit  $\#A = n$  und  $\mathbf{p} = (p_1, \dots, p_n)$ . Man konstruiert eine Folge

$$F^{(n)}, F^{(n-1)}, \dots, F^{(3)}, F^{(2)}, F^{(1)},$$

wobei jedes

$$F^{(k)} = \{(t_1^{(k)}, g_1), \dots, (t_k^{(k)}, g_k)\}$$

ein Wald von  $k$  gewichteten Binärbäumen ist, wobei

$$g_1 \geq g_2 \geq \dots \geq g_k \text{ und } g_1 + \dots + g_k = 1.$$

Dabei ist das Gewicht eines gewichteten Binärbaumes ist die Summe der Gewichte seiner Blätter unter der Bewertung.

- Die Konstruktion startet mit

$$F^{(n)} = \{(\square, p_1), \dots, (\square, p_n)\}$$

- Der Konstruktionsschritt  $F^{(k)} \rightarrow F^{(k-1)}$  fusioniert die beiden Bäume aus  $F^{(k)}$  mit dem geringsten Gewicht:

$$F^{(k-1)} = \left( F^{(k)} \setminus \{(t_{k-1}^{(k)}, g_{k-1}), (t_k^{(k)}, g_k)\} \right) \cup (s, h)$$

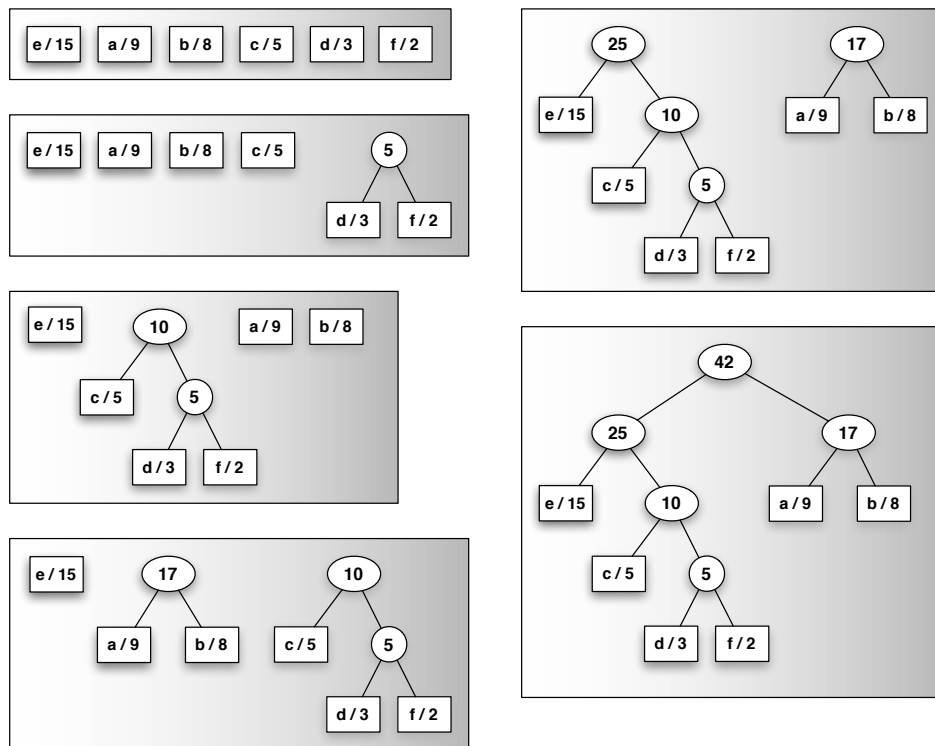
wobei

$$s = \langle \bigcirc, t_{k-1}^{(k)}, t_k^{(k)} \rangle, \quad h = g_{k-1} + g_k.$$

Die Bäume in  $F^{(k-1)}$  sind noch nach ihrem Gewicht zu ordnen.

- $F^{(1)} = \{t_1^{(1)}\}$  ist ein Baum und dies ist der HUFFMAN-Code. □

*Beispiel 25.* Obiges Beispiel in dieser Darstellung:



*Bemerkung 28.* Zur *Komplexität* der HUFFMAN-Konstruktion ist festzuhalten:

- Die Information über die Baum-Gewichte  $g_j$  kann man in einer *priority queue* organisieren! Diese Queue kann man als minimum-heap implementieren.
- Der Aufbau des heaps kostet  $\mathcal{O}(n)$  Aufwand.
- Die  $n-1$  Konstruktionsschritte erfordern heap-Operationen ( $2 \times$  DELETEMIN,  $1 \times$  REHEAP) mit einem Aufwand von jeweils  $\mathcal{O}(\log n)$ .
- Der Gesamtaufwand (an Vergleichsoperationen) ist somit  $\mathcal{O}(n \log n)$ .

*Kommentar 29.* Einige weiterführende Bemerkungen:

- HUFFMANS Konstruktion ist ein klassischer GREEDY-Algorithmus.
- Weitere bekannte GREEDY-Algorithmen
  - Zahldarstellung in Positionssystemen
  - Minimale Gerüste (Spannbäume) (KRUSKAL, PRIM)
  - Kürzeste Wege (DIJKSTRA)
  - Knapsack (“fractional”)

- GREEDY gehört mit DIVIDE-AND-CONQUER, DYNAMIC PROGRAMMING, BACK-TRACKING mit BRANCH-AND-BOUND, RANDOMIZATION zu den fundamentalen Entwurfsprinzipien für Algorithmen
- Das Typische an GREEDY-Problemen: optimale Lösungen von Teilproblemen lassen sich immer zu global-optimalen Lösungen fortsetzen
- GREEDY kann man nicht immer einsetzen, aber wenn ja, ist es sehr effizient!
- Man kann genau charakterisieren, in welchen Situationen GREEDY funktioniert (das führt auf den Begriff der “Matriode”).

## 6.5 Kommentare, Literatur, Ausblicke

### 6.5.1 Bäume und Suchbäume, Sortierkomplexität

Das Thema *Bäume in der Informatik* ist ein “weites Feld”. Bäume liefern allgegenwärtige Strukturierungs- und Analysekonzepte, von denen in diesem Kapitel nur Teilaspekte behandelt wurden. Die Literatur ist immens, deshalb sollen hier ganz wenige Hinweise genügen.

Eine umfassende Darstellung findet man, wie zu erwarten, in den beiden Bänden [8] und [9] von KNUTH. Speziell aus analytischer Sicht ist auch [20] eine Fundgrube. Die Lehrbuchliteratur widmet sich diesen Objekten natürlich auch intensiv, was man etwas an den Texten von CORMEN, LEISERSON, RIVEST, STEIN [3], HEUN [5], SCHÖNING [18] und SEDGEWICK [19].

Zu allen Aspekten des Sortierens ist natürlich das *opus magnum* [9] von KNUTH an erster Stelle zu nennen. In Lehrbuchform findet man u.a. in den schon vorher genannten Texten, sowie dem Buch von BRASSARD, BRATLEY [2] Darstellungen von unterschiedlichem Umfang und Tiefe.

### 6.5.2 Quellcodierung und Entropie

Die *Mathematische Theorie der Information* blickte schlagartig mit der epochalen Arbeit [21] von Claude SHANNON das Licht der Welt. Keine andere Arbeit hat die Entwicklung einer Theorie der Information und Informationsverarbeitung so sehr beeinflusst, wie diese. Sie ist in kommentierter Form als Büchlein [22] heute noch erhältlich. Die grundlegenden Resultate über Quell- und Kanalcodierung sind hier bereits formuliert und bewiesen. Kein Lehrbuch der Informations- und/oder Codierungstheorie kann daran vorbeigehen. Empfehlenswert sind die Bücher (in

alphabetischer Reihenfolge der Autorennamen) von APPLEBAUM [1], COVER und THOMAS [4], MACKAY [11], MCELIECE [12], ROTH [15], WELCH [23]. aber auch das ist nur eine subjektive Auswahl aus einer riesigen Literatur.

Die Original der Ungleichung von KRAFT für Präfixcodes findet sich in einer Masterthesis [10] vom MIT. Das Resultat wurde von MACMILLAN in [13] auf beliebige UD-Codes verallgemeinert. Der hier gegebene einfache Beweis geht auf [7] von KARUSH zurück.

Wer etwas über die Zusammenhänge zwischen dem physikalischen und dem informationstheoretischen Informations- und Entropiebegriff erfahren will, findet in dem ganz aktuellen Buch [14] von MÉZARD und MONTANARI eine faszinierend anregende Lektüre, die zudem mannigfache Anwendungen in der Informatik präsentiert.

### 6.5.3 Huffmans Algorithmus, Datenkompression

Die Konstruktion optimaler Präfixcodes nach HUFFMAN ist nicht nur ein "Klassiker" der Algorithmik, sondern begründet gleichzeitig das Gebiet der verlustfreien *Datenkompression*. SHANNON sagt, was möglich ist, HUFFMAN zeigt, wie es optimal gemacht wird. Nebenbei: auch SHANNON gibt ein Verfahren von Codes an, die seiner Entropie-Ungleichung genügen, aber der entstehende Code ist nicht immer optimal bezüglich der mittleren Wortlänge.

Die Arbeit [6] ist das HUFFMANSche Original. Darstellungen finden sich in allen Büchern der Informationstheorie, siehe vorigen Punkt, aber eben der Algorithmik. Als Belege seien wieder die (ohnehin empfehlenswerten) Lehrbücher von [3], [5], [18], [19] genannt.

Für das Gebiet der Datenkompression hat SALOMON das Kompendium [16] verfasst. Von ihm stammt auch ein schönes und aktuelles Buch [17] über Datenkompression mittels Codes variabler Länge.

### 6.5.4 Greedy-Algorithmen

Wie schon im Text erwähnt, ist der Algorithmus von HUFFMAN ein Paradebeispiel für einen "Greedy"-Algorithmus. Entsprechend häufig kommt er auch in der Algorithmik-Literatur unter dieser Überschrift vor, so beispielsweise bei BRASSARD, BRATLEY [2] CORMEN, LEISERSON, RIVEST, STEIN [3] SCHÖNING [18] Hinter der Idee der Greedy-Algorithmen steckt ein abstraktes mathematisches

Konzept, das der *Matroide*. Speziell in [3] und [18] erfährt man etwas über diese Zusammenhänge.

## Literatur

- [1] David Applebaum. *Probability and Information*. Cambridge University Press, Cambridge, second edition, 2008. An integrated approach.
- [2] Gilles Brassard and Paul Bratley. *Algorithmics*. Prentice Hall Inc., Englewood Cliffs, NJ, 1988. Theory and practice, Translated from the French.
- [3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001.
- [4] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, second edition, 2006.
- [5] Volker Heun. *Grundlegende Algorithmen*. Vieweg, 2. edition, 2003.
- [6] David A. Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(10):1089–1101, 1952.
- [7] J. Karush. A simple proof of an inequality of McMillan. *IRE Trans. Info. Theory*, IT-7(2):118, 1961.
- [8] Donald E. Knuth. *The Art of Computer Programming. Vol. 1. Fundamental algorithms*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, third edition, 1997.
- [9] Donald E. Knuth. *The Art of Computer Programming. Vol. 3. Sorting and Searching*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, second edition, 1998.
- [10] L. G. Kraft. A device for quantizing, grouping and coding amplitude modulated pulses. Master's thesis, Department of Electrical Engineering, MIT, Cambridge, MA, 1949.
- [11] David J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, New York, 2003.
- [12] Robert J. McEliece. *The Theory of Information and Coding*. Number 3 in Encyclopedia of Mathematics and its Applications. Addison-Wesley Publishing Co., Reading, Mass., 1977.

- 
- [13] Brian McMillan. The basic theorems of information theory. *Ann. Math. Stat.*, 24(2), 1953.
- [14] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford University Press, 2009.
- [15] Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.
- [16] David Salomon. *Data Compression — The Complete Reference*. Springer-Verlag, 1997.
- [17] David Salomon. *Variable-length Codes for Data Compression*. Springer-Verlag, 2007.
- [18] Uwe Schöning. *Algorithmik*. Spektrum Akademischer Verlag, 2001.
- [19] Robert Sedgewick. *Algorithms*. Addison-Wesley Publishing Co., Reading, Mass., 1988.
- [20] Robert Sedgewick and Philippe Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Co., Reading, Mass., 1996.
- [21] Claude Elwood Shannon. A mathematical theory of communication. *Bell Sys. Tech. J.*, 28:379–423, 623–656, 1948.
- [22] Claude Elwood Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.
- [23] Dominic Welsh. *Codes and Cryptography*. Oxford University Press, 1988.