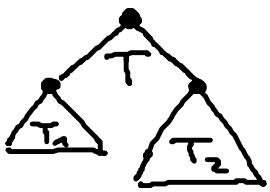
Algorithmus 3.1 (Huffman Algorithmus)

1. *Fasse jedes Zeichen $c \in C$ als einelementigen Baum auf und füge es in eine Priority Queue Q ein, wobei die Häufigkeit $f(c)$ als Schlüsselwert dient.*

2. *Solange Q mehr als einen Baum enthält:*

   - *Wähle die beiden Bäume T_1 und T_2 mit den kleinsten Häufigkeiten (muss nicht eindeutig sein).*
   - *Entferne sie aus Q.*
   - *Konstruiere einen neuen Baum aus den T_1, T_2 als Teilbäume unter einer neuen Wurzel und gebe ihm die Häufigkeit $f(T_1) + f(T_2)$.*
   - *Füge diesen Baum in Q ein.*

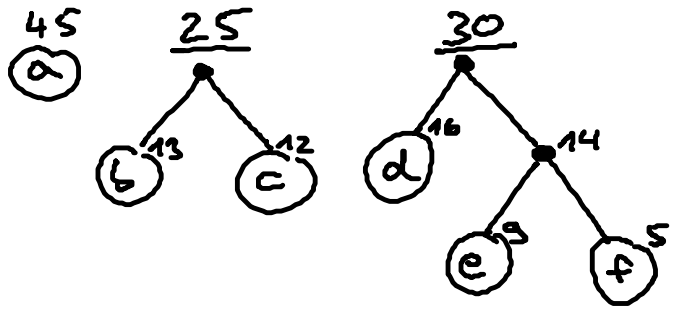
3. *Gebe (den einzig übrig gebliebenen Baum) T in Q zurück. Dieser Baum (der so genannte* Huffman Baum *oder* Huffman Code*) ist ein optimaler Präfixcode.*

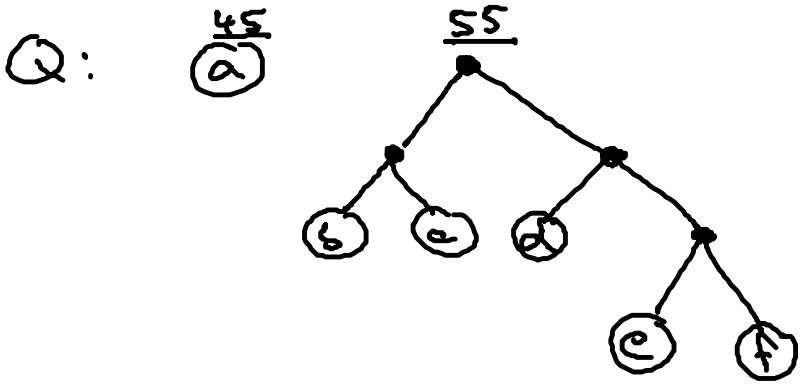


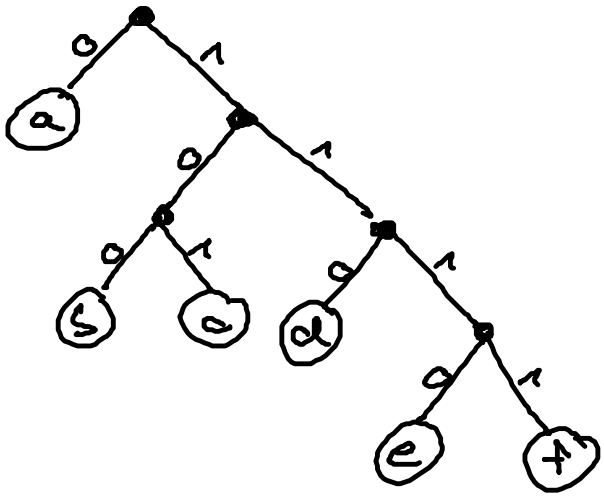
$f(T) = f(T_1) + f(T_2)$

Beispiel:

...

Q: 

Q: 

Q: 

e: 1110

Laufzeit des Algo.

1.	alle Zeichen einfügen in Q:	$O(n)$
2.	$n-1$ Phasen:	$n-1$ mal:
	- die beiden Kleinsten aus Q entfernen:	$O(2\log n)$
	- neuen Baum bauen:	$O(1)$
	- wieder einfügen in Q:	$O(\log n)$
3.	Baum zurück geben:	$O(1)$
insgesamt:		$O(n \log n)$

zusätzlich: Bestimmen Zeichensatz C und

Häufigkeiten $f(c)$ durch einmaliges Lesen der gegebenen Textdatei: $O(\text{Dateigröße})$

Optimalitätsbeweis:

Zeige, dass Huffman-Alg. opt. Präfixcode T findet, d.h.

$$B(T) = \sum_{c \in C} f(c) \cdot h_T(c) \leq B(T')$$

für alle Präfixcodes T'.

Nutze für den Beweis zwei Lemmata:

Lemma 1: Es seien $x, y \in C$ die beiden Zeichen mit den kleinsten Häufigkeiten, d.h. $f(x) \leq f(y) \leq f(c) \quad \forall c \in C \setminus \{x, y\}$.

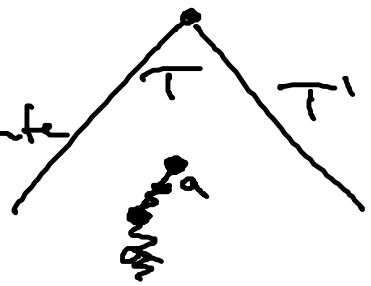
Dann gibt es opt. Präfixcode T, in dem x und y auf der tiefsten Schicht liegen d.h. $h_T(x) = h_T(y) \geq h_T(c) \; \forall c \in C \setminus \{x, y\}$ und Geschwister sind (d.h. gemeinsamen Mutterknoten).

(Achtung: Das gilt nicht für jeden opt. Präfixcode)

Beweis: Es sei: T ein optimaler Präfixcode.
Es sei: $a \in C$ ein Blatt mit größter Höhe
$h_T(a) \geq h_T(c) \quad \forall c \in C$.

Beh.: Dann hat a einen Bruder $b \in C$,

denn:
Sonst betrachte
Baum T'



mit $B(T') < B(T)$ ↯ Optimalität von T.

Falls $\{a,b\} = \{x,y\}$, dann fertig.
Falls nicht, dann modifizere T geringfügig ohne Optimalität zu verlieren.
$\rightarrow T'$ mit $B(T') = B(T)$ und
T' erfüllt Eigenschaft aus Lemma.

Modifizere in 2 Schritten:
1) Vertausche x mit a $T \to T_1$
2) " y mit b $T_1 \to T'$

$$B(T) - B(T_1) = \sum_{c \in C} f(c) \cdot h_T(c) - \sum_{c \in C} f(c) \cdot h_{T_1}(c)$$

$$= f(a) \cdot h_T(a) + f(x) \cdot h_T(x) - f(a) \cdot \underbrace{h_{T_1}(a)}_{= h_T(x)} - f(x) \cdot \underbrace{h_{T_1}(x)}_{= h_T(a)}$$

$$= \underbrace{(f(a) - f(x))}_{\geq 0} \cdot \underbrace{(h_T(a) - h_T(x))}_{\geq 0} \geq 0$$

Opt. von T $\Rightarrow$ $B(T_1) = B(T)$

$B(T_1) - B(T') \geq 0$ (genauso)

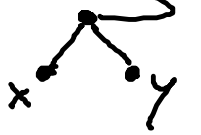
$\Rightarrow$ $B(T') = B(T_1) = B(T)$. □

Lemma 2: Es seien $x, y \in C$ Zeichen mit geringsten Häufigkeiten und T ein Präfixcode, in dem x und y Geschwister sind mit gemeinsamem Mutter z. Es entstehe T' aus T durch Weglassen der Blätter x und y und $C' = (C \setminus \{x,y\}) \cup \{z\}$ und

$$f'(c) = \begin{cases} f(c) & \text{falls } c \neq z \\ f(x)+f(y) & \text{u } c = z \end{cases} \quad \forall c \in C'.$$

Dann gilt:
T ist optimal für C und f genau dann, wenn T' optimal ist für C' und f'.

Beweis: Es sei: H die Menge aller Präfixcodes für C, die die Eigenschaft besitzen, dass x und y Geschw. sind mit Mutter z. Es sei: H' die Menge aller Präfixcodes für C'.
Dann beschreibt $T \mapsto T'$ eine Abbildung von H nach H'. Diese Abbildung ist bijektiv

(wenn man die Anordnung der Kinder x, y ignoriert oder oBdA 

)

injektiv: klar.

surjektiv: Zu gegebenem T' konstruiere T durch Anhängen von x und y an das Blatt z. $\Rightarrow T \mapsto T'$.

zeige jetzt: $B(T) = B(T') + k$ (↑ Konstante)

(dann gilt offensar

$B(T)$ minimal $\Leftrightarrow$ $B(T')$ minimal)

$$B(T) - B(T') = \sum_{c \in C} f(c) \cdot h_T(c) - \sum_{c \in C'} f(c) \cdot h_{T'}(c)$$

$$= f(x) \cdot h_T(x) + f(y) \cdot h_T(y) - f(z) \cdot h_{T'}(z)$$

$$= (f(x) + f(y)) \cdot h_T(x) - (f(x) + f(y)) \cdot (h_T(x) - 1)$$

$$= f(x) + f(y) =: k \text{ konstant.}$$ □

Theorem: Der Huffman Algo. konstruiert einen optimalen Präfixcode.

Beweis: Ind. über $n := |C|$

Ind. Anf.: $n = 2$

opt. Präfixcode:  ≙ Ausgabe des Algo.

Ind. Vor.: Der Alg. findet einen opt. Präfixcode für $n-1$ Zeichen für festes $n \in \mathbb{N}$.

Ind. Schluss auf n:

In erster Iteration des Algo.: Ermittle Zeichen x, y mit geringsten Häufigkeit und konstruiere 

und füge den neuen Baum in Q ein.

$\rightarrow Q$ enthält $n-1$ Bäume. Interpretiere 

als neues Zeichen z mit

$$f(z) = f(x) + f(y).$$

Im weiteren läuft der Huffman Alg. genau so, wie er für C' und f' läuft.

Nach Ind. Vor. liefert er also einen optimalen Präfixcode T' für C' und f'!

Lemma 2 $\Rightarrow T$ ist ebenfalls optimal für C und f. $\square$

Bemerkungen zu Huffman Codes und Alternativen:

Huffman	alternativ
zeichenweise Kodierung	nicht zeichenweise (kodiere ganze Strings mit einem Codewort)

statisch: zunächst gesamte Ausgangsdatei lesen und Häufigkeiten analysieren.	dynamisch: "on the fly": Codeworte werden beim Lesen der Datei erzeugt und dynamisch verändert ("adaptiv") → höheres Optimierungspotential
verlustfrei	verlustbehaftet.

Suchbäume

Problemstellung:

Halte dynamische Daten im Speicher und unterstütze die folgenden Operationen:

- suchen
- einfügen
- löschen

} möglichst schnell.

Dafür kennen wir bislang nur Listen:

- suchen $O(n)$
- einfügen $O(1)$
- löschen $O(1)$ (ohne Suchen)

Deutliche Verbesserung durch Suchbäume:

$O(\log n)$ Aufwand für alle 3 Operationen