Background Paper Presentation

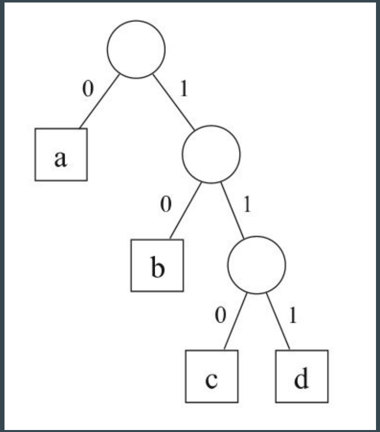
Blanchette, J.C.: *Proof pearl: Mechanizing the textbook proof of Huffman's algorithm.*
J. Autom. Reason. 2009

- Huffman Coding
- Huffman's Algorithm
- Proof of Optimality
- Conclusion
- Our Project

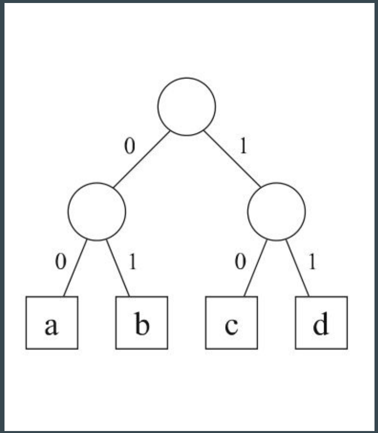
Daniel Filipe Nunes Silva & Samuel Chassot

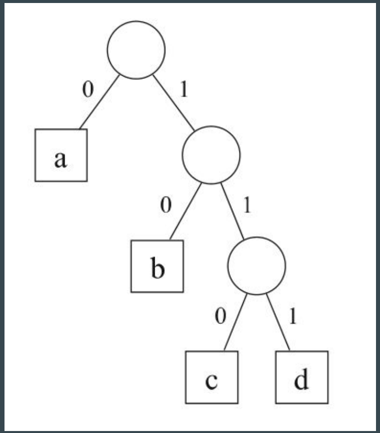
Huffman Coding

- Code, $\boldsymbol{C} = \{ a \rightarrow 0, b \rightarrow 10, c \rightarrow 110, d \rightarrow 111 \}$
- Full binary tree
- Minimize encoded string length
- Compression



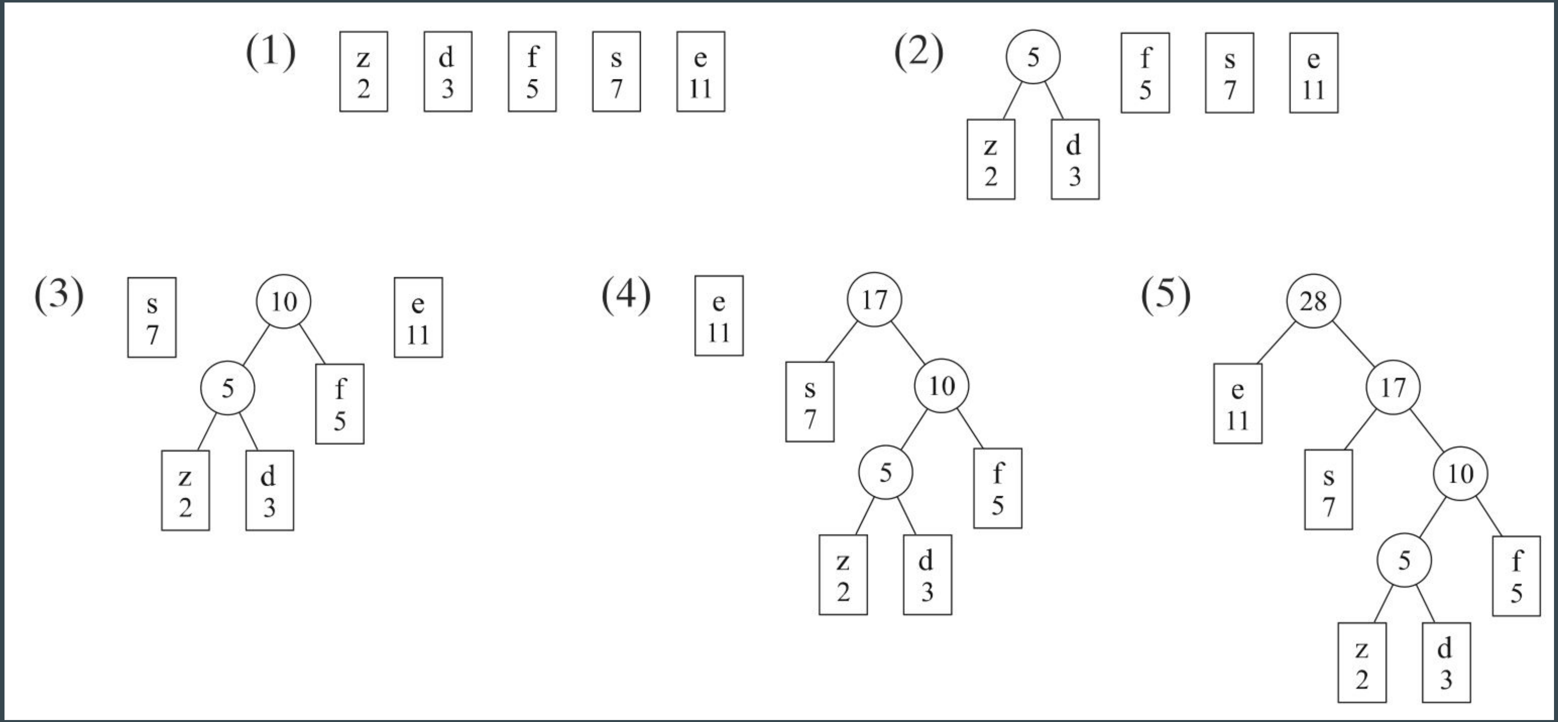
Huffman's Algorithm





- Fixed-length vs variable-length
- Optimal solution
- Example for string "abacabad"
- **Input**
  - Alphabet, $A = (a_1, a_2, \dots, a_n)$
  - Weights, $W = (w_1, w_2, \dots, w_n)$, where w_i is the weight of a_i
- **Output** : Code, $C = (c_1, c_2, \dots, c_n)$, where c_i is the codeword for a_i
- Minimal sum of $w_i \cdot \text{length}(c_i)$

Huffman's Algorithm Example



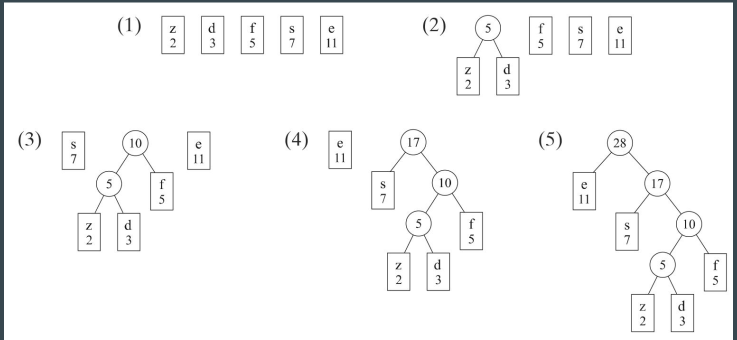
Functional Implementation

- Datatypes
  - **Leaf a w**
  - **InnerNode w t_1 t_2**
- Functions
  - **cachedWeight _**
  - **uniteTrees t_1 t_2**
  - **insortTree t f**

- Huffman's algorithm
  - **huffman** (t_1 t_2 ts) =

huffman(**insortTree**(**uniteTrees** t_1 t_2) ts)

  - **huffman** [t] = t



Basic auxiliary functions

- functions defined on trees
- mostly recursive
- goal → define ***optimality*** and lemmas to prove it

- *alphabet:*

$$\mathit{alphabet}\,(\mathit{Leaf}\ w\ a) = \{a\}; \quad \mathit{alphabet}\,(\mathit{InnerNode}\ w\ t_1\ t_2) = \mathit{alphabet}\ t_1 \cup \mathit{alphabet}\ t_2.$$

- *consistent:*

$$\begin{aligned} \mathit{consistent}\,(\mathit{Leaf}\ w\ a) &= \mathit{True} \\ \mathit{consistent}\,(\mathit{InnerNode}\ w\ t_1\ t_2) &= (\mathit{consistent}\ t_1 \wedge \mathit{consistent}\ t_2 \\ &\qquad\qquad \wedge\ \mathit{alphabet}\ t_1 \cap \mathit{alphabet}\ t_2 = \emptyset). \end{aligned}$$

- *depth:*

$$
\begin{aligned}
\mathit{depth}\ (\mathit{Leaf}\ w\ b)\ a &= 0 \\
\mathit{depth}\ (\mathit{InnerNode}\ w\ t_1\ t_2)\ a &= (\text{if } a \in \mathit{alphabet}\ t_1 \text{ then } \mathit{depth}\ t_1\ a + 1 \\
&\quad\ \text{else if } a \in \mathit{alphabet}\ t_2 \text{ then } \mathit{depth}\ t_2\ a + 1 \\
&\quad\ \text{else } 0).
\end{aligned}
$$

- *height:*

$$
\begin{aligned}
\mathit{height}\ (\mathit{Leaf}\ w\ a) &= 0 \\
\mathit{height}\ (\mathit{InnerNode}\ w\ t_1\ t_2) &= \mathit{max}\ (\mathit{height}\ t_1)\ (\mathit{height}\ t_2) + 1.
\end{aligned}
$$

- *frequency:*

$$
\begin{aligned}
\mathit{freq}\ (\mathit{Leaf}\ w\ b)\ a &= (\text{if } a = b \text{ then } w \text{ else } 0) \\
\mathit{freq}\ (\mathit{InnerNode}\ w\ t_1\ t_2)\ a &= \mathit{freq}\ t_1\ a + \mathit{freq}\ t_2\ a.
\end{aligned}
$$

- *weight:*

$$
\begin{aligned}
\mathit{weight}\ (\mathit{Leaf}\ w\ a) &= w \\
\mathit{weight}\ (\mathit{InnerNode}\ w\ t_1\ t_2) &= \mathit{weight}\ t_1 + \mathit{weight}\ t_2.
\end{aligned}
$$

- *cost:*

$$\begin{aligned} cost\ (Leaf\ w\ a) &= 0 \\ cost\ (InnerNode\ w\ t_1\ t_2) &= weight\ t_1 + cost\ t_1 + weight\ t_2 + cost\ t_2. \end{aligned}$$

- *optimum:*

$$\begin{aligned} optimum\ t\ =\ (\forall u.\ & consistent\ u \longrightarrow alphabet\ t = alphabet\ u \longrightarrow freq\ t = freq\ u \\ & \longrightarrow cost\ t \le cost\ u). \end{aligned}$$

- *swapFourSyms:*

$$\begin{aligned} \mathit{swapFourSyms}\ t\ a\ b\ c\ d \ =\ & (\text{if } a = d \text{ then } \mathit{swapSyms}\ t\ b\ c \\ & \ \text{else if } b = c \text{ then } \mathit{swapSyms}\ t\ a\ d \\ & \ \text{else } \mathit{swapSyms}\ (\mathit{swapSyms}\ t\ a\ c)\ b\ d). \end{aligned}$$

Exchange symbols so that a & b occupies the positions of c & d

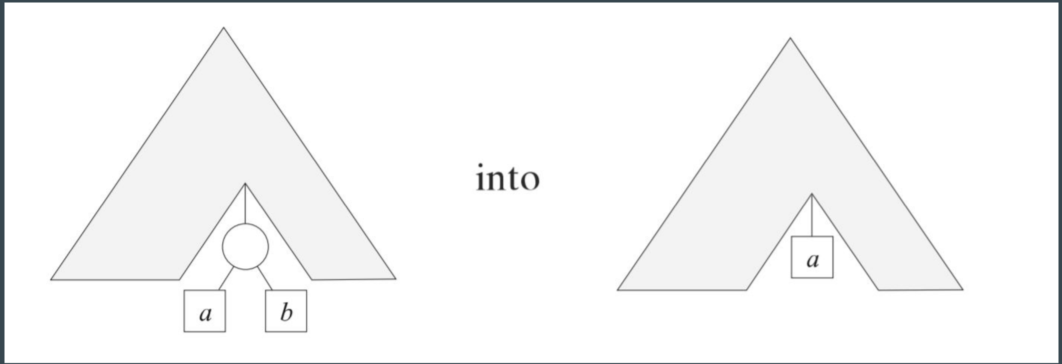
$$\begin{aligned} \mathit{swapLeaves}\ (\mathit{Leaf}\ w_c\ c)\ w_a\ a\ w_b\ b \ =\ & (\text{if } c = a \text{ then } \mathit{Leaf}\ w_b\ b \\ & \ \text{else if } c = b \text{ then } \mathit{Leaf}\ w_a\ a \\ & \ \text{else } \mathit{Leaf}\ w_c\ c) \\ \mathit{swapLeaves}\ (\mathit{InnerNode}\ w\ t_1\ t_2)\ w_a\ a\ w_b\ b \ =\ & \mathit{InnerNode}\ w\ (\mathit{swapLeaves}\ t_1\ w_a\ a\ w_b\ b) \\ & \qquad (\mathit{swapLeaves}\ t_2\ w_a\ a\ w_b\ b) \\ \mathit{swapSyms}\ t\ a\ b \ =\ & \mathit{swapLeaves}\ t\ (\mathit{freq}\ t\ a)\ a\ (\mathit{freq}\ t\ b)\ b \end{aligned}$$

- *mergeSiblings:*

$$mergeSibling\ (Leaf\ w_b\ b)\ a = Leaf\ w_b\ b$$

$$\begin{aligned} mergeSibling\ (InnerNode\ w\ \ (Leaf\ w_b\ b) & \\ (Leaf\ w_c\ c))\ a & = \begin{array}{l}(\text{if } a = b \vee a = c \text{ then } Leaf\ (w_b + w_c)\ a \\ \ \text{else } InnerNode\ w\ (Leaf\ w_b\ b)\ (Leaf\ w_c\ c))\end{array} \end{aligned}$$

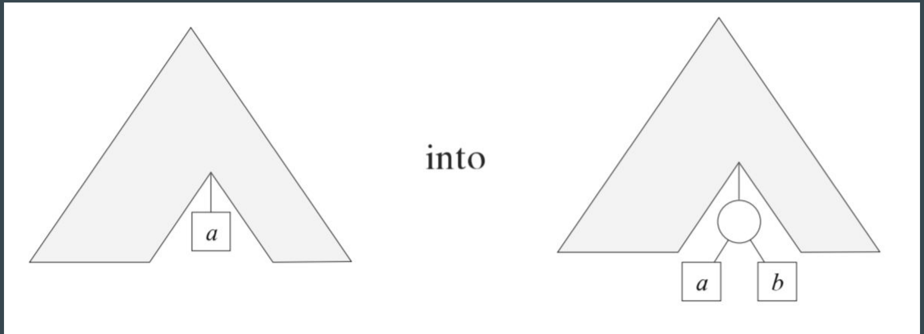
$$\begin{aligned} mergeSibling\ (InnerNode\ w\ t_1\ t_2)\ a = InnerNode\ w\ \ & (mergeSibling\ t_1\ a) \\ & (mergeSibling\ t_2\ a). \end{aligned}$$



- *splitLeaf:*

$$
\begin{aligned}
\mathit{splitLeaf}\,(\mathit{Leaf}\,w_c\;c)\;w_a\;a\;w_b\;b \;&=\; (\text{if } c = a \;\text{ then }\; \mathit{InnerNode}\,w_c\;\; (\mathit{Leaf}\,w_a\;a)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\mathit{Leaf}\,w_b\;b)\\
&\qquad\qquad\quad\;\;\text{else }\; \mathit{Leaf}\,w_c\;c)\\
\mathit{splitLeaf}\,(\mathit{InnerNode}\,w\;t_1\;t_2)\;w_a\;a\;w_b\;b \;&=\; \mathit{InnerNode}\,w\;\;(\mathit{splitLeaf}\,t_1\;w_a\;a\;w_b\;b)\\
&\qquad\qquad\qquad\quad\;(\mathit{splitLeaf}\,t_2\;w_a\;a\;w_b\;b).
\end{aligned}
$$

Normally, $w_a + w_b = freq(t, a)$

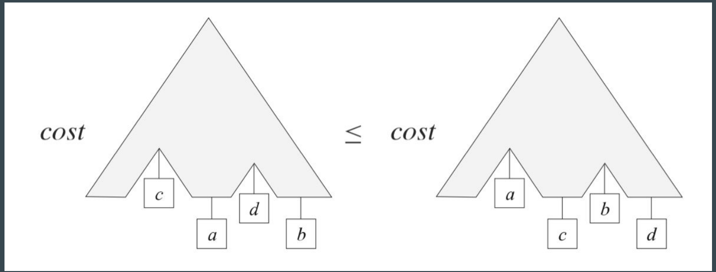


Lemma 8.5 - Leaf Split Optimality

If $consistent\ t$, $optimum\ t$, $a \in alphabet\ t$, $b \notin alphabet\ t$, $freq\ t\ a = w_a + w_b$, $\forall c \in alphabet\ t.\ w_a \leq freq\ t\ c \wedge w_b \leq freq\ t\ c$, *and* $w_a \leq w_b$, *then* $optimum\ (splitLeaf\ t\ w_a\ a\ w_b\ b)$.

Intermediary lemmas - intuition

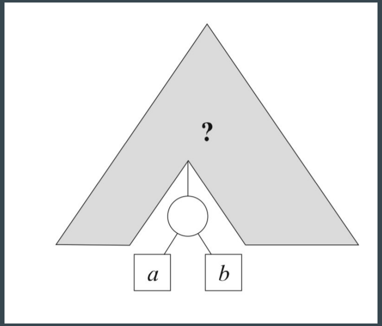
- **Lemma 8.4:** If ***a*** and ***b*** are the minima of the tree:



- **Lemma 7.2:** merging two siblings ***a*** and ***b*** decreases the ***cost*** by $w_a + w_b$

Lemma 8.5 - Leaf Split Optimality - Proof

- assume $\forall u, \mathit{cost}(t) \leq \mathit{cost}(u)$
- $\mathit{height}(t) > 0$: exists c and d, two sibling symbols at the bottom of u.
- swap c and d with a and b, minima of u.
  - We obtain a new tree v.

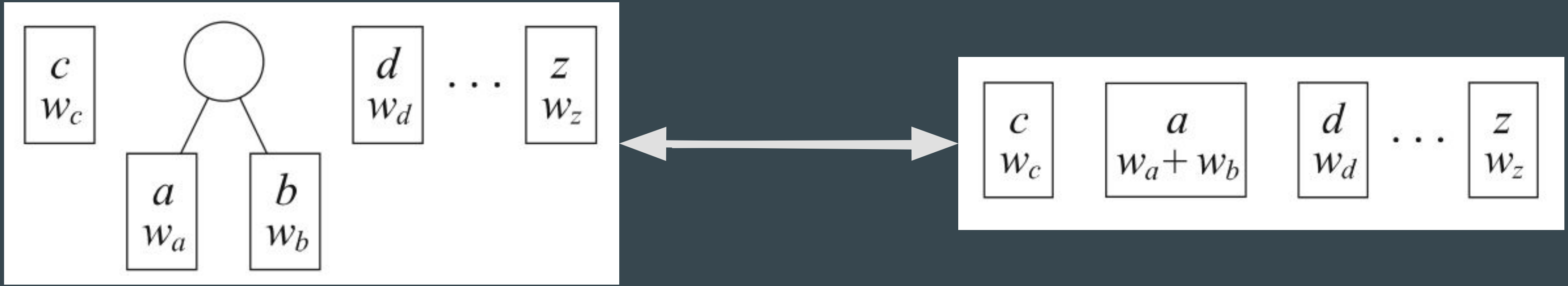


$$
\begin{aligned}
& \mathit{cost}\ (\mathit{splitLeaf}\ t\ a\ w_a\ b\ w_b) \\
= \quad & && \text{by Lemma 7.3} \\
& \mathit{cost}\ t + w_a + w_b \\
\leq \quad & && \text{by assumption} \\
& \mathit{cost}\ \overbrace{(\mathit{mergeSibling}\ (\mathit{swapFourSyms}\ u\ a\ b\ c\ d)\ a)}^{v} + w_a + w_b \\
= \quad & && \text{by Lemma 7.2} \\
& \mathit{cost}\ (\mathit{swapFourSyms}\ u\ a\ b\ c\ d) \\
\leq \quad & && \text{by Lemma 8.4} \\
& \mathit{cost}\ u.
\end{aligned}
$$

$\square$

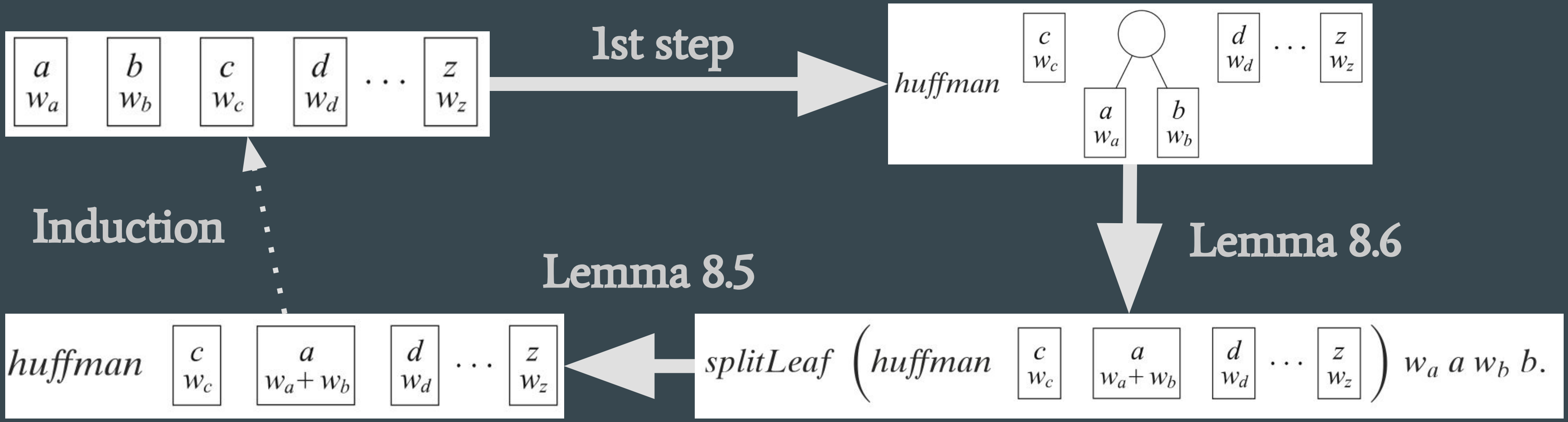
Leaf Split Commutativity

Lemma 8.6 (Leaf Split Commutativity) *If* $consistent_{\mathsf{F}}\ ts$, $ts \neq []$, *and* $a \in alphabet_{\mathsf{F}}\ ts$, *then* $splitLeaf\ (huffman\ ts)\ w_a\ a\ w_b\ b = huffman\ (splitLeaf_{\mathsf{F}}\ ts\ w_a\ a\ w_b\ b)$.



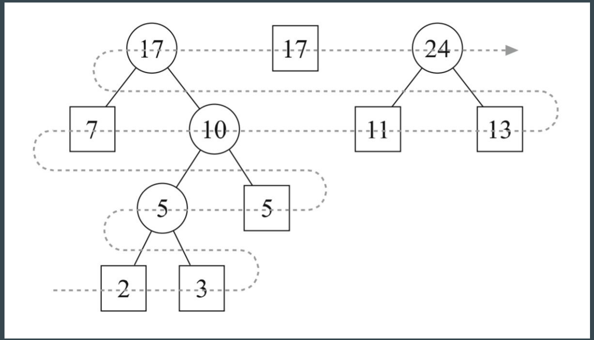
Huffman Optimality

Theorem 8.7 (Huffman Optimality) *If* $consistent_F$ *ts*, $height_F$ *ts* $= 0$, *sortedByWeight ts, and ts* $\neq []$, *then optimum* (*huffman ts*).



Conclusion

- They formalized and demonstrated the textbook's proof of the Huffman algorithm
- They found that custom induction rules simplify a lot the proof using Isabelle
- They defined Lemmas that were not seen in the literature
    - e.g. each step of the algorithm preserves this invariant:
        - the nodes of the forest are ordered by weight from left to right, bottom to top:
        - They could not prove the preservation though



Our project

- They proposed to then extend the proof's scope to the applications of the algorithm.
- That is what we will do:

  We will implement a pair of *encode/decode* functions using the tree produced by the Huffman's algorithm and prove that they are bijectives.
  i.e. that

$$\textit{decode}(\textit{encode}(x)) = x$$