Huffman Encoding

Let some text be given:

How is such a text usually encoded?
-> e.g.: Subset of ASCII-letters

What might be an „optimal“ code?

Assumptions:

- every letter s_i in the original text is replaced by a code l_i .
- We are looking for an *optimal code* in the sense that *this code minimizes the averaged code word length.*

The averaged code word length L is compiuted as follows:

$$L = \sum_{i=1}^{n} p_i \cdot l_i$$

Huffman Encoding

Rough description of the algorithm:

1.) examine, how often each letter occurs in the original text.

2.) build a so called Huffman Tree

3.) build a table with so called Huffman Codes

Huffman Encoding

1.) examine, which letter occurs how often in the given text

go through the input text and count the occurrences of each letter.

Example.: „test_string“

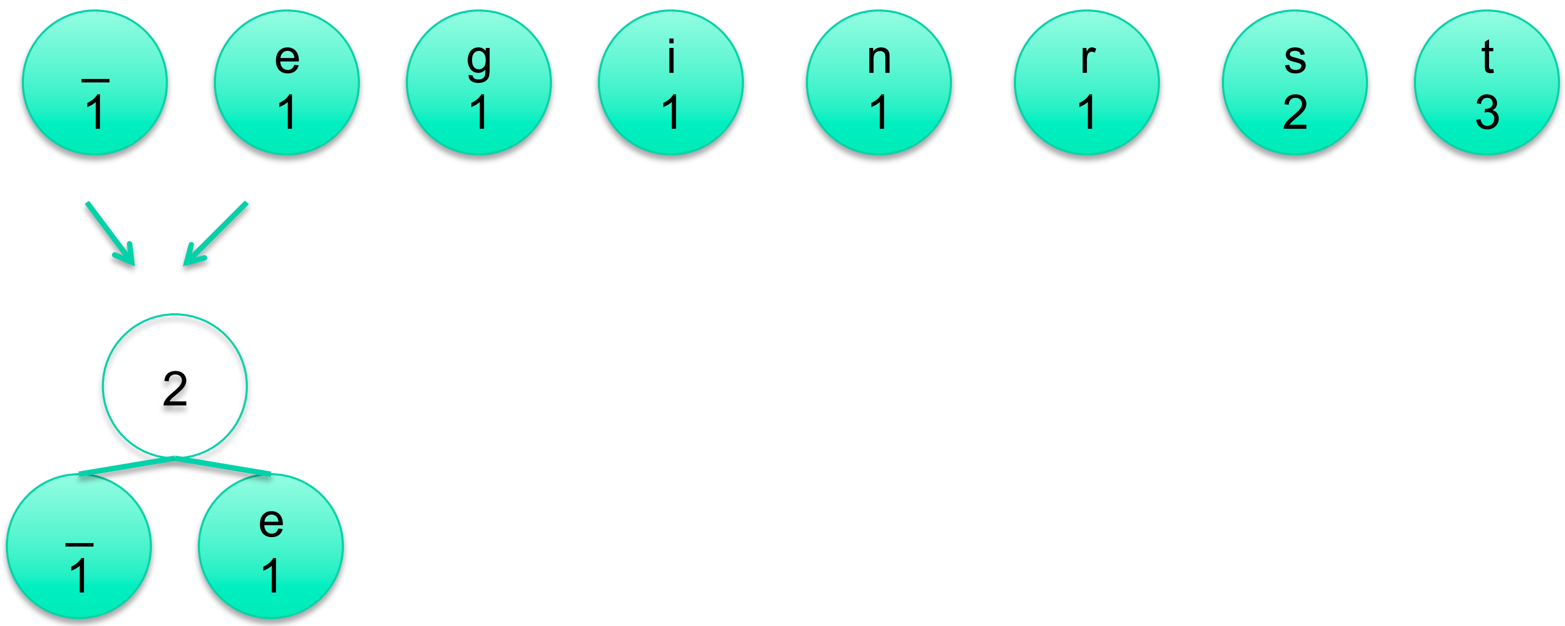
letters	_	e	g	i	n	r	s	t
occurrences	1	1	1	1	1	1	2	3

Huffman Encoding

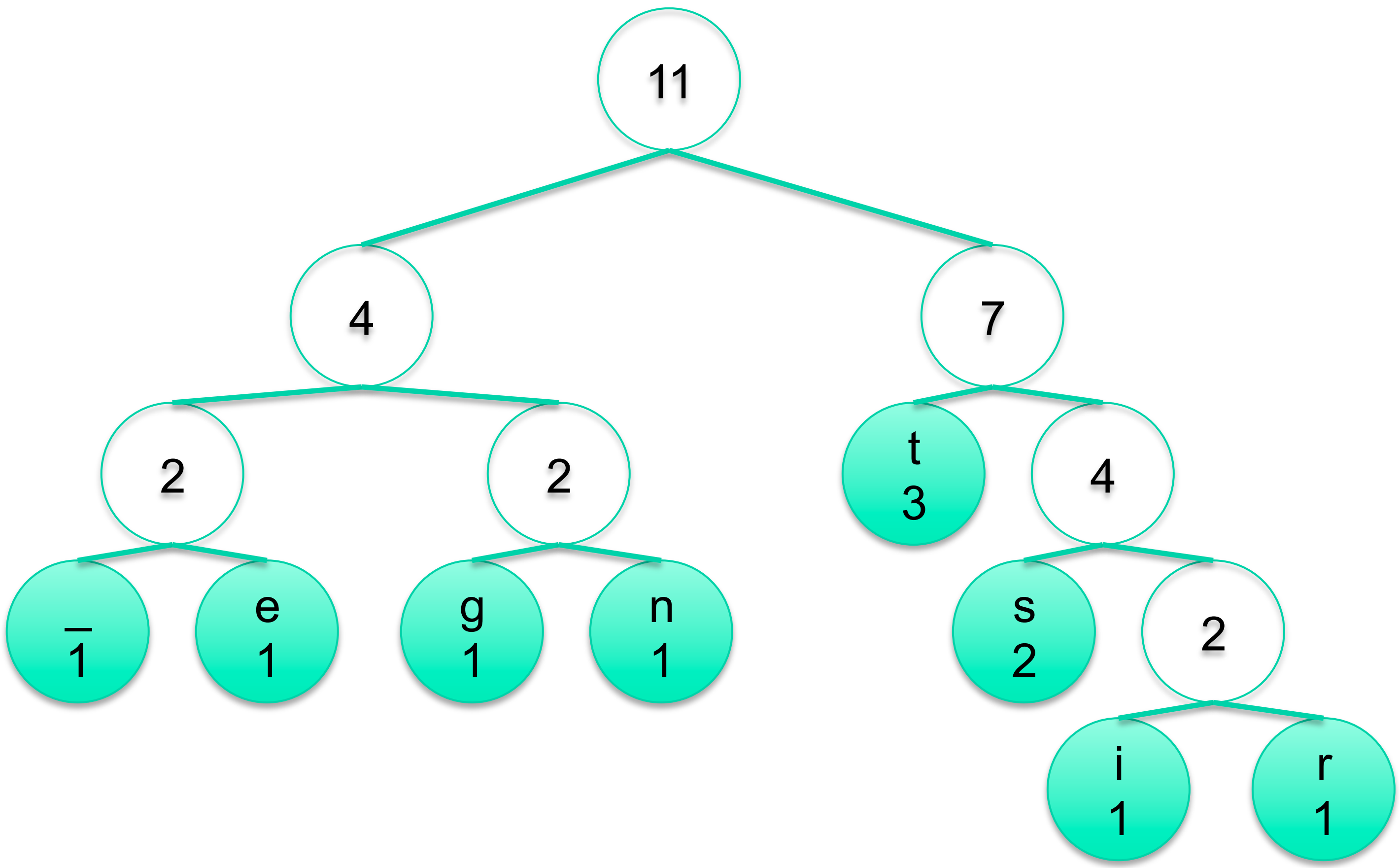
2.) build the so called Huffman Tree

Build the tree as follows: Firstly, each occuring letter is caught in its own tree. Thereafter, those two trees that have the smallest number of occurrences are brought together. The sum of the occurrences of the old roots is written into a new root node.

Example.:



Huffman Encoding



Huffman Coding

TECHNISCHE UNIVERSITÄT DARMSTADT

3.) build a table with the final Huffman Codes

000	_
001	e
010	g
011	n
10	t
110	s
1110	i
1111	r

Encoded text:

10001110100001101011111110011010

Observation: No code is prefix of another code.

Let Σ be the alphabet for which the code is to be generated. It contains $|\Sigma| = n$ letters (characters).

Lemma 1: Every inner node in a minimal prefix tree possesses two children.

Proof: Let us assume that a minimal tree T, which possesses an inner node with only one child, exists. Then, we construct a tree T‘ with one node less: We remove the single successor and replace it by its child-node.

For this new tree is valid: some encodings of some letters have been shortened. This is a contradictoin to the assumption that the tree T was minimal.

Lemma 2: Let s_i and s_j be those letters with smallest occuring probability. Then, s_i and s_j have maximum depth in T.

Proof:
Assumption: there is a letter s that is placed in maximum depth, but not having smallest occuring probability.
Then we exchange s with s_i or with s_j and receive a smaller total encoding.

Optimality of Huffman-Coding

Theorem: The Huffman-Coding has minimal expected encoding length.
Proof by induction over $|\Sigma|$.

- Induction start for $|\Sigma| \leq 2$ is clear.
- Now, let $|\Sigma|>2$ and let T be a tree, representing the optimal prefix code for Σ.
  - 1st observation: Every inner node in T has two children (otherwise contradiction to optimality).
  - 2nd observation: Let s_i and s_j be the letters with smallest occuring probability. Then s_i and s_j are in maximum depth in T (otherwise contradiction to optimality).
- Thus: s_i and s_j are in T as in the Huffman-Tree
- Replace s_i and s_j with a new letter s with $\text{Prob}(s) = \text{Prob}(s_i) + \text{Prob}(s_j)$.
- Induct.-assumption.: Remaining Huffman-Tree for new Σ is optimal

$\Rightarrow$ induction step