Algorithmics - Tutorial Sheet 2

Strings and text algorithms

---

1. [**Work in pairs**] Let s be the following string:

   `dogs do not spot hot pots or cats`

   (a) Draw the character frequency table for the string s.

   (b) Draw a Huffman tree encoding the string s.

   (c) Write the Huffman codes for the characters c and d based on your Huffman tree drawn for the previous task.

2. Explain how the Huffman-tree-building algorithm can be extended so that it also determines the weighted path length of the tree that it constructs.

   Recall: the weighted path length (`WPL`) of a tree T is the sum of

   $$(\text{weight}) \times (\text{distance from root})$$

   over all leaf nodes of the tree.

3. [**Work in pairs**] Trace the `LZW` algorithm by hand on the text below. Generate a representation of the compressed file (with code words represented by integers), and a representation of the dictionary trie at the end of the compression. (Assume the dictionary initially contain the 128 characters of the basic `ASCII` character set, represented by the code words $0, 1, \ldots, 127$.)

   `peter piper picked a peck of pickled pepper`

   For information: the coding of relevant `ASCII` characters are:

   "space" : 32 `a` : 97 `c` : 99 `d` : 100 `e` : 101 `f` : 102 `i` : 105
   `k` : 107 `l` : 108 `o` : 111 `p` : 112 `r` : 114 `t` : 116

   and the next available code word is 128.

4. Suppose that the `LZW` compression algorithm is applied to the text:

   `abababababab...`

   of length 100 (i.e. the text that contains an alternating sequence of a's and b's and 100 characters in total).

   Determine the resulting compression ratio and saved space (as a percentage of the original file size). Assume that each character in the source file occupies 8 bits, that the initial dictionary size is 128, and that the initial codeword size is 8.

   Hint: apply the `LZW` algorithm until a pattern in the amount of text used up in each step emerges. This pattern can then be used to compute the number of steps required to reach the 100th (and last) character. From the number of steps you can then find the number of code words required to encode the text.