Huffman Compression

Eric Gunn

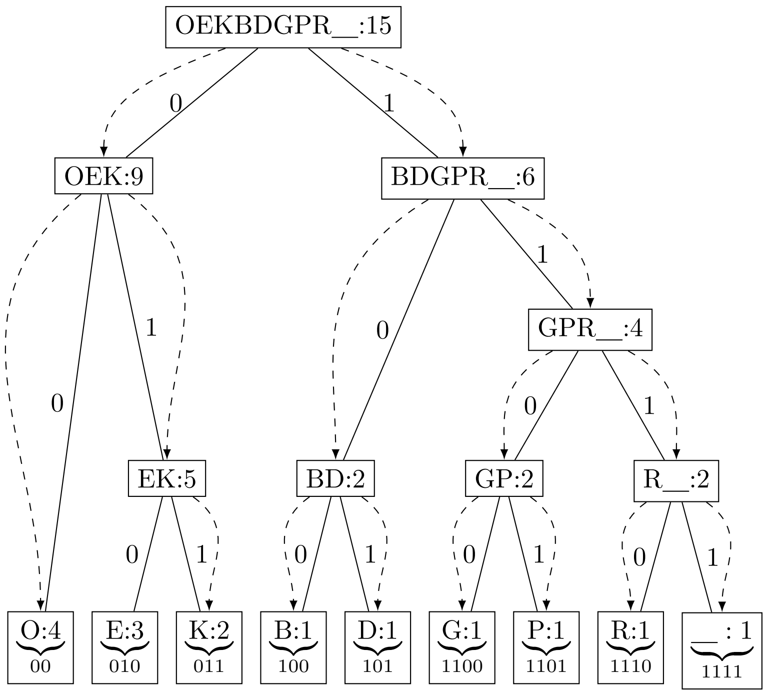
Huffman compression is an interesting way to use space efficiently by reducing the number of bits needed to represent a string of text. For example, take the string **GOOD BOOKKEEPER** and convert it to its ASCII binary representation:

Character	Binary ASCII Representation
G	01000111
O	01001111
O	01001111
D	01000100
	00100000
B	01000010
O	01001111
O	01001111
K	01001011
K	01001011
E	01000101
E	01000101
P	01010000
E	01000101
R	01010010

Each binary ASCII value is one byte (8 bits). Since there are 15 total characters in the string (including the space), we know that the string is 120 bits long. Huffman compression works by using fewer bits to represent the most frequent characters in a string. Here is a table of the characters in the string sorted by frequency in descending order:

Character	Frequency
O	4
E	3
K	2
B	1
D	1
G	1
P	1
R	1
	1

If the frequency table is taken and put it into a tree, with each node being a sum of its children, the following is the result (here, __ is used to represent a space. The character appears on the left with the frequency or sum of frequencies on the right, separated with a colon.):



Each node in the tree is assigned either a 0 if on the left, or a 1 if on the right. By traversing the tree to its leaves (endpoints) it is found that there is a unique path to each leaf, so each character can now assume a new (and smaller) binary representation.

Character	Frequency	Huffman Compression Representation
O	4	00
E	3	010
K	2	011
B	1	100
D	1	101
G	1	1100
P	1	1101
R	1	1110
	1	1111

After the compression process, the string is now 45 bits instead of 120 bits, yielding a 62.5% reduction in space.