

Encoding documents, music, images, and video for the purposes of compression is a consistent theme in this course. Our primary focus on compression is through the aperture of Huffman's algorithm which finds optimal binary prefix-free codes for a set of  $n$  symbols  $s_1, \dots, s_n$  which occur  $o_1, \dots, o_n$  times respectively. The procedure creates a set of trees – one for each symbol – and initializes the weight  $w_i$  of each tree to  $o_i$ . It then selects two trees with minimum weight, and replaces them with a new tree composed of the subtrees. This new tree has weight equal to the sum of weights of its subtrees. The procedure ends when only a single tree remains.

## Huffman Tree Cost

The cost of a Huffman tree is equivalent to the size of the document when it's encoded using the code defined by the tree. In other words, if symbol  $s_i$  appears  $o_i$  times and  $s_i$  is a leaf at depth  $d_i$  in the tree then each occurrence of  $s_i$  in the document<sup>1</sup> is encoded with  $d_i$  bits. As a result,  $s_i$  contributes cost  $o_i \times d_i$  to the total cost of the encoded document. In general the total cost of the encoded document (and hence the tree) is

$$o_1 \times d_1 + o_2 \times d_2 \cdots + o_n \times d_n \quad (1)$$

For example, consider the following document (of length 11) composed of 5 symbols:

ABRACADABRA

Table 1 lists each symbol and its number of occurrences.

Symbol	A	B	R	C	D
Occurrences	5	2	2	1	1

Table 1: A list of symbols and their respective occurrences for the document ABRACADABRA

Running Huffman's algorithm on the table might produce the following Huffman tree:

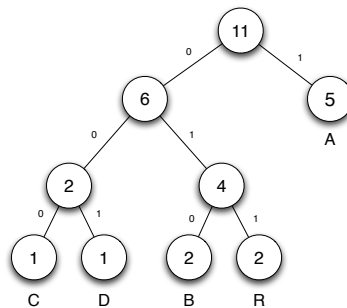


Figure 1: A possible tree produced by Huffman's algorithm on the input given in Table 1

Encoding the document according to the code defined by the Huffman tree yields

10100111000100110100111

which has a length of 23. In other words, the cost of the tree is 23. We can verify this is indeed the cost of the code by noting that in the tree,  $A$  has depth 1 and the remaining symbols have depth 3. Hence, using just the tree, we can also arrive at the cost by using Equation 1.

$$1 \times 3 + 1 \times 3 + 2 \times 3 + 2 \times 3 + 5 \times 1 = 3 + 3 + 6 + 6 + 5 = 23$$

Surprisingly, one does not need to actually create a tree to compute the cost of a Huffman code. The idea is to keep a running total  $T$  of the current cost of encoding the document. To see how, imagine that we are creating the Huffman

<sup>1</sup>document here is really just a place holder for music, image or video – that is a series of symbols

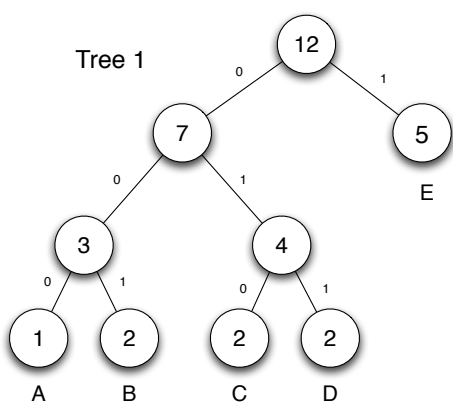
tree given in Figure 1. Initially, our running total is  $T = 0$ . Every time we merge two trees, the codewords for the symbols in those trees increase by 1 bit. Our procedure works by adding in this additional codeword cost at the time of a merge.

- We begin by merging C and D. We now know that each occurrence of C and D will be replaced by codewords with length at least 1, so each occurrence of C and each occurrence of D will contribute cost at least 1 to the overall cost. Since C and D each occur only once,  $T = T + 2 = 0 + 2 = 2$ .
- Next we merge B and R. Since B and R will be replaced by codewords with length at least 1, each occurrence of B and each occurrence of R will contribute cost at least 1 to the overall cost. Since B and R each occur twice they will contribute an additional cost of  $2 + 2 = 4$  to  $T$ . Hence,  $T = 6$ .
- Next we merge the CD tree with the BR tree. The CD tree has weight 2. This means C and D occur 2 times collectively in the document. The BR tree has weight 4. This means B and R occur 4 times collectively in the document. Since we are merging these two nodes, we know the codewords for C and D will be 1 bit longer in the document. The same is true for B and R. Since C and D occur twice collectively, this merge adds a cost of 2 to the total. Similarly, since B and R occur 4 times collectively, the merge adds a cost of 4 to the total. Hence the total is now  $T = T + 2 + 4 = 6 + 2 + 4 = 12$ .
- Finally, we merge the CDBR node with the A node. Each occurrence of A will be encoded with a codeword of length 1 and the symbols in the CDBR node each require an additional bit in their keywords. Since A occurs 5 times and C,D,B, and R occur 6 times collectively, our total is  $T = T + 5 + 6 = 12 + 5 + 6 = 23$ .

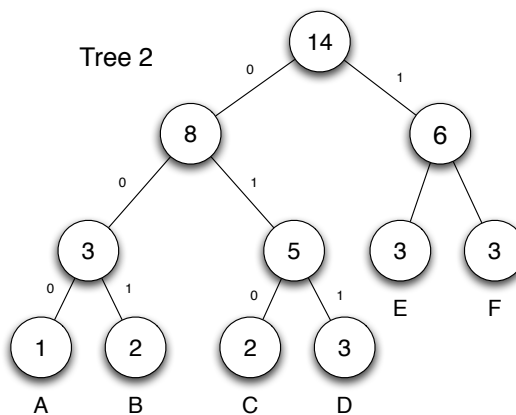
Note that while we discussed computing the running total in terms of tree merges, we don't actually need to create or merge any trees; we just need to find the two smallest weights and replace them with their sum in the current list of weights. This means each merge operation is really just a sum operation. Each sum operation has an associated update to the running total so it makes sense to talk about the value of the running total after each sum operation. For example, the running total after each sum operation for the Huffman tree in Figure 1 is given below:

Sum Operation	1	2	3	4
Running Total	2	6	12	23

**Question 1.** Below are two Huffman trees called Tree 1 and Tree 2 respectively. We would like you to show us the running total of the current cost after each sum operation for each tree. We have provided running total tables for you, so all you need to do is fill them in and hand in this sheet of paper.



Sum Operation	1	2	3	4
Running Total				



Sum Operation	1	2	3	4	5
Running Total					