

1. **Compression** [10 pts]

If a prefix code has $n = 2^k$ codewords and at least one codeword is shorter than k bits, prove that there must be at least two codewords longer than k bits.

2. **Huff and puff.** In class, we used Huffman's algorithm to construct optimal prefix codes for a fixed message probability distribution. However, most practical compression algorithms will update the message probability distribution as they encode the input, which means that the Huffman code has to be updated on every message of the input sequence. A naive solution to this issue is to rerun Huffman's algorithm from scratch on every step, which can be very costly. In this problem, we will study a more efficient way of updating the Huffman code.

Consider the *Huffman tree* produced by Huffman's algorithm. Notice that instead of using message probability to determine the weight of nodes (as we did in class), we can use message *frequency* (number of occurrences) without changing any other aspect of the algorithm or its analysis. Let $w(u)$ denote the weight of node u .

Consider the following property: a binary tree with n leaves has the *sibling property* if and only if:

- (i) the n leaves a_1, a_2, \dots, a_n have positive integer weights $w(a_1), w(a_2), \dots, w(a_n)$, and the weight of each internal node is equal to the sum of its children's weights; and
- (ii) there exists a *labeling function* f that labels each node with a distinct number in $\{1, 2, \dots, 2n - 1\}$ such that for all nodes u, v : (1) $f(u) < f(v)$ implies $w(u) \leq w(v)$ (i.e. non-decreasing weight); and (2) $f(u) = 2j - 1$ and $f(v) = 2j$ for some $j \in \{1, 2, \dots, n - 1\}$ iff u and v are siblings.

Notice that a binary tree is a Huffman tree if and only if it satisfies the sibling property (this is not hard to prove, but you can assume it).

- (a) Let $l(u)$ denote the *level* of node u , i.e., its distance from the root. Show that $l(u) \geq l(v)$ if $w(u) < w(v)$ for any two nodes u and v in the tree. Hint: you can use the fact that Huffman codes optimal.
- (b) Let u and v be two nodes with the same weight $w(u) = w(v)$. Show that u and v cannot be more than one level apart, i.e. $|l(u) - l(v)| \leq 1$.
- (c) Suppose that we want to update Huffman tree T by processing the message a , and suppose that T already contains a leaf for message a . Consider the following naive algorithm: increment by 1 the weight of the leaf corresponding to a and all of its ancestors in T . Give a concrete example or describe a condition under which the resulting tree is not a Huffman tree.
- (d) Consider the following two-phase procedure.

Input: Huffman tree T , message a .

Phase 1: modify tree.

$u \leftarrow$ leaf in T corresponding to a

while $u \neq \text{root}(T)$ **do**

$v \leftarrow$ node v with the maximum $f(v)$ such that $w(u) = w(v)$

Exchange the subtree rooted at u with the subtree rooted at v
Swap the labels of u and v
 $u \leftarrow$ parent of u

Phase 2: update weights

$u \leftarrow$ leaf in T corresponding to a

while $u \neq \text{root}(T)$ **do**

$w(u) \leftarrow w(u) + 1$

$u \leftarrow$ parent of u

$w(\text{root}(T)) \leftarrow w(\text{root}(T)) + 1$

Show that the resulting tree is a Huffman tree.

3. Perfect arithmetic coding [10 pts]

Suppose that a message set S has probabilities of the form $1/2^k$ where k is a positive integer. For example, there can be four messages with probabilities $1/2, 1/4, 1/8, 1/8$. Show that a slight modification to arithmetic coding achieves an expectation of $n \cdot H(S)$ bits to send n messages (instead of at most $n \cdot H(S) + 2$ from lecture). In other words, it is optimal and matches the lower bound. (Hint: encode each interval more efficiently.)