Solution Set

CIS 1210—Data Structures and Algorithms—Fall 2024

Huffman Coding—Tuesday, October 8 / Wednesday, October 9

Readings

- Lecture Notes Chapter 15: Huffman Coding

Review: Huffman Coding

The motivation behind Huffman Coding is to encode and decode characters as bits, minimizing the average bits per letter (ABL). Furthermore, we seek a prefix-free code, where no encoding is a prefix of another — implying that a bit sequence can be parsed and decoded without any ambiguity.

The Huffman algorithm is a greedy algorithm that does this. Given a set of characters and their frequencies, the algorithm outputs an encoding by repeatedly merging the 2 nodes with the smallest frequency values until only one node remains. This one node is the root of the Huffman tree, whose leaves are characters and each root-to-leaf path is an encoding of that character. Furthermore, this tree is a full binary tree, where each internal node has exactly 2 children. Therefore, the Huffman algorithm produces an optimal and prefix-free encoding that minimizes the ABL.

The running time of the Huffman algorithm is $O(n \log n)$ if we utilize a min-heap to find the 2 nodes with minimum frequency in each step, as seen in the pseudocode. This is because at each step, we perform a constant number of Extract-Min and Insert operations, which take $O(\log n)$ time, and we repeat this for $O(n)$ iterations.

Problems

Problem 1

Construct an optimal Huffman coding for the following alphabet and frequency table S:

Character:	A	B	C	D	E
Frequency:	0.4	0.3	0.15	0.1	0.05

What is the ABL, or average bits per letter, for this encoding?

Solution

The following tree T would be produced:

```
 /\
A /\
 B /\
  C /\
   E  D
```

$$\text{ABL}(T) = \sum_{x \in S} f_x \cdot \text{depth}_T(x) = 0.4 \cdot 1 + 0.3 \cdot 2 + 0.15 \cdot 3 + 0.1 \cdot 4 + 0.05 \cdot 4 = \boxed{2.05}$$

Problem 2

You have an alphabet with $n > 2$ letters and frequencies. You perform Huffman encoding on this alphabet, and notice that the character with the largest frequency is encoded by just a 0. In this alphabet, symbol i occurs with probability $p_i; p_1 \geq p_2 \geq p_3 \geq ... \geq p_n$.

Given this alphabet and encoding, does there exist an assignment of probabilities to p_1 through p_n such that $p_1 < \frac{1}{3}$? Justify your answer.

Solution

There does not exist an assignment of probabilities to p_1 through p_n such that $p_1 < \frac{1}{3}$. Assume for the sake of contradiction that there exists an assignment such that $p_1 < \frac{1}{3}$. Consider the last step of the Huffman algorithm when our two final nodes are merged into one node. Let these two final nodes be x and y. Because character 1, which has the highest frequency, has an encoding length of 1, it must have been merged via this step. WLOG, let x be this character 1. Since this is the final step of Huffman, we know $p_x + p_y = 1$. From our assumption that $p_1 < \frac{1}{3}$, we know $p_y > \frac{2}{3}$.

Because $n > 2$, y must be a node representing at least 2 characters. Consider the time when y was created. Let the nodes that were merged to become y be a and b. We know $p_a + p_b > \frac{2}{3}$, which implies that $max\{p_a, p_b\} > \frac{1}{3}$. Here, we have reached a contradiction. Huffman always merges the two smallest frequency nodes, but when y was created, node x with $p_x < \frac{1}{3}$ was still available and unmerged. Hence, x would have been chosen instead of $max\{a, b\}$. Therefore, via contradiction, we have proved that the original claim is false and thus that there does not exist an assignment of probabilities as specified.