Introduction

Huffman coding is a common technique used to losslessly compress text. It uses relative character frequency to encode text such that the least possible space is taken on average. In this lab we will be focusing on encoding text into binary, though later you will learn other methods.

Implementation

We represent an encoding as a tree, with each leaf representing a character, and the path from the root to the leaf describing its encoding. Each edge in the tree is assigned either 1 or 0. By convention, each left edge is 0. We can construct such a tree as follows:

As input, assume that we have access to the relative probabilities with which each character appears. Treat each character as a tree of one node, with weight equal to its frequency, and construct a min-heap of the trees. Now we perform the following steps:

1. Remove the lowest frequency item from the heap (tree $A$), then remove the new lowest frequency item (tree $B$).
2. Construct a new tree by creating a root node with two children: $A$ as the left child, and $B$ as the right child.
3. Add this tree back into the heap, with weight equal to the combined weights of $A$ and $B$.
4. If the heap is of size greater than 1, repeat from step 1.

When this process is complete, the only remaining item in the heap is our final Huffman coding tree.

Expected Encoding Length

Using the frequency with which each character appears, and the length of the compressed characters, we can find the expected length of one encoded character.

Definition 1. The expected value of a variable is the sum of its possible values multiplied by their respective probabilities ($E = \sum x_p$)

It is important to note that Huffman encoding results in optimal expected encoding length. That is to say, no other encoding will produce results requiring fewer average bits to encode text.

Practice Problems

Problem 1. Construct an alphabet $A$ with frequencies such that in an optimal Huffman coding there exist at least two encodings of length exactly $(n-2)$, where $n$ is the size of the alphabet. $n$ must be at least 5.

Problem 2. Construct an optimal Huffman coding for the following alphabet and frequency table:

<table>
<thead>
<tr>
<th>Character</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0.3</td>
<td>0.3</td>
<td>0.15</td>
<td>0.15</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Problem 3. Find the average encoded character length of the following alphabet and frequency table:

1. This direction of this step is convention; the reverse is just as valid, but for this course we will expect you to abide by this standard.
<table>
<thead>
<tr>
<th>Character</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0.4</td>
<td>0.3</td>
<td>0.15</td>
<td>0.1</td>
<td>0.04</td>
<td>0.01</td>
</tr>
</tbody>
</table>