38.8 Exercises

Factual

1. Suppose we build a Shannon Fano or Huffman code for the text of this question including spaces and punctuation characters. Which characters would have the longest code?
2. What two ways could we represent a Huffman code for characters in Java?

<details>
<summary>Problem 1</summary>

`?` and `.`, since both are only used once in the above sentence.

</details>

<details>
<summary>Problem 2</summary>

A `HashMap<Character, BitSequence>` or a `BitSequence[]`. Note that the two are equivalent in Java because a Character is a number.

</details>

Procedural

1. Suppose we have a string `abcdefg` which repeats 1000 times. How many bits would be in the compressed bitstream?

<details>
<summary></summary>

Since all 8 characters are equal in frequency, we get a balanced binary tree as our Huffman encoding, so all codewords are 3 bits long. 1000 * 8 * 3 = 24000 bits.

</details>

Metacognitive

1. Using the idea of self-extracting bits, come up with an encoding for the sequence `abdefg` repeated 1000 times that uses less than 2000 bits.

Problem 1

The idea of self-extracting bits includes writing code or an interpreter that can generate the original uncompressed sequence. This can be done with the following code:

```
public class Sequence {
    public static void main(String[] args) {
        for (int i = 0; i < 1000; i++) {
            for (int j = 0; j < 8; j++) {
                System.out.print(String.format("%c", 'a' + j));
            }
        }
    }
}
```

This code uses exactly 239 characters, or 1912 bits. This demonstrates the power of the self-extracting bits model: compare this to the 24000 bits required for a Huffman code.

Previous
38.7 Summary

Next
39. Compression, Complexity, P = NP

Last updated 1 year ago