Lecture 38: Compression.

Compression model #1:

Algorithm C.

Bitstream B → → Compressed Bits C(B) → Algorithm C^{-1} → B

lossless = no information is lost.

- Text files are often compressible by 70% or more.

Prefix free code:

Characters = 8 bits long.

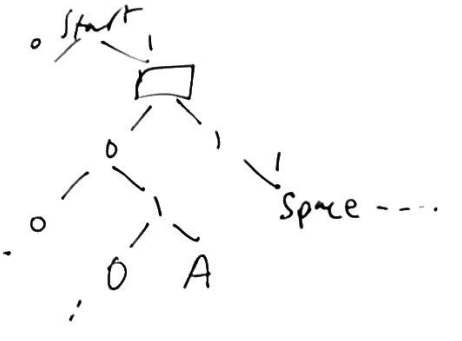
easy way to compress:

Use fewer than 8 bits for each letter.

Codewords go with symbols.

Prefix free code = no codeword is a prefix of any other

Bushy prefix-free tree:



Shannon–Fano Coding

- Split into left and right halves of roughly equal frequency.
  - ↳ left half gets a leading 0. Is the one w/ higher frequency.
  - ↳ right half gets a leading 1.
  - ↳ Repeat.

Huffman coding:

- Assign each symbol to a node w/ weight = frequency.
- Take the 2 smallest nodes & merge them into a super node. (weight = sum of child weights.)
- Repeat until everything is a part of the tree.

For encoding, use a map of <char> : <BitStream>.
This lets you map & find key/values easily.

For decoding, use a Trie. (Longest matching prefix).

Usually, for every possible input file, there is a unique code just for that file. The code gets sent with the compressed file.

Huffman coding:

1. Count frequencies.
2. Build encoding array and decoding trie.
3. Write decoding trie to output.huf
4. Write codeword for each symbol to output.huf.

Recap:
Given a X.test we need to compress into X.huf:

- Consider each b-bit symbol and count the occurrences of each of the 2^b possibilities where b is the size of each symbol in bits.
- Use Huffman code construction to make a decoding trie & encoding map. Store this trie at the beginning of X.huf.
- Use encoding map to write codeword of each symbol of input into X.huf.

To decompress:

- read in decoding trie
- Use longest prefix of until all bits have been converted.

Compression model:

As a model, let's treat the algorithm & the compressed bitstream as a single sequence of bits.