HUFFMAN ENCODING YEAH HOURS

Alexander De Baets

Cynthia Spent a lot of time working out some examples, so be sure to watch lecture if you missed it!

The Problem

byte	1	2	3	4	5	6	7	8	9	10
char	'a'	'b'	' '	'a'	'b'	' '	'c'	'a'	'b'	EOF
ASCII	97	98	32	97	98	32	99	97	98	256
binary	01100001	01100010	00100000	01100001	01100010	00100000	01100011	01100001	01100010	N/A

So, how do I do that?

1. **Count character frequencies:** Make a quick pass through the file to be encoded, counting how many of each distinct character you find in the text.
2. **Build encoding tree:** Build a binary tree using a specific queue-based algorithm.
3. **Build encoding map:** Traverse the binary tree to record the binary encodings of each character in a map for future quick-access use.
4. **Encode the file:** Do a second pass through the file, looking up each byte (each character) in the map from Step 3, to get the encoding for each byte.

Step 1: Count the character frequencies

byte	1	2	3	4	5	6	7	8	9	10
char	'a'	'b'	' '	'a'	'b'	' '	'c'	'a'	'b'	EOF
ASCII	97	98	32	97	98	32	99	97	98	256
binary	01100001	01100010	00100000	01100001	01100010	00100000	01100011	01100001	01100010	N/A

Step 2: Building the Encoding Tree

(Example on board)

THe Huffman Node Struct

```
struct HuffmanNode {
    int character;          // character being represented by this node
    int count;              // number of occurrences of that character
    HuffmanNode* zero;      // 0 (left) subtree (NULL if empty)
    HuffmanNode* one;       // 1 (right) subtree (NULL if empty)
    ...
};
```

Step 3: Building the encoding map

(on the board)

Reading and Writing to Files

`obitstream` (bit output stream) member	Description
`void writeBit(int bit)`	writes a single *bit* (0 or 1) to the output stream

`ibitstream` (bit input stream) member	Description
`int readBit()`	reads a single *bit* (0 or 1) from input; -1 at end of file

`ostream` (output stream) member	Description
`void put(int byte)`	writes a single *byte* (character, 8 bits) to the output stream

`istream` (input stream) member	Description
`int get()`	reads a single *byte* (character, 8 bits) from input; -1 at EOF

Step 4: Encode the Text Data!

(on board)

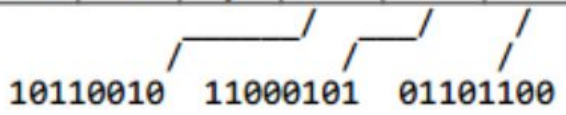
Decoding the Data

(on board)

The Header

byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'{'	'3'	'2'	':'	'2'	','	' '	'9'	'7'	':'	'3'	','	' '	'9'	'8'	':'	'3'
byte	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
	','	' '	'9'	'9'	':'	'1'	','	' '	'2'	'5'	'6'	':'	'1'	'}'	*	*	*

```
// output << frequencyTable;     input >> frequencyTable;
// output.writeBit(...);         input.readBit(); ...
```



Putting it all together!

- Compress
- Decompress
- FreeTree