Assignment 5
Hamming Codes

DESIGN DOCUMENT

Zack Traczyk
CSE13S - Spring 2021

Due: May 9th at 11:59 pm

1 Objective

The main programs, encode and decode, encode and decode files using the Hamming(8,4) code. The provided code for the error program injects noise into an encoded file. Entropy calculates the entropy of a file.

2 Given

- Header files for stack, path, and graph
- Pseudocode for recursive search

3 Prelab Questions

a. Completed Lookup table

Table 1 on the following page shows which error vector values are associated with which bits/ error codes.

b. Decode the following codes. If it contains an error, show and explain how to correct it.

1110 0011 $\rightarrow e$ = (1233) = (1011) = b1101 = 13 = 2nd bit

1101 1000 $\rightarrow e$ = (2121) = (0101) = b1010 = 5 = HAM_ERR

Value	Bit
0	0
1	4
2	5
3	HAM_ERR
4	6
5	HAM_ERR
6	HAM_ERR
7	3
8	7
9	HAM_ERR
10	HAM_ERR
11	2
12	HAM_ERR
13	1
14	0
15	HAM_ERR

Table 1: Lookup Table

4 Programs

4.1 Encode

This program encodes the Hamming(8,4) code. Bytes can be encoded from stdin or a file, and to stdout or a file. The program arguments for encode are as follows:

- -h : Command line options
- -i infile : The file containing bytes to be encoded (default is stdin)
- -o outfile : The output file to store encoded bytes (default is stdout)

4.2 Decode

This program decodes the Hamming(8,4) code. As in the encode program, Bytes can be decoded from stdin or a file, and to stdout or a file. The program arguments for decode are as follows:

- -h : Command line options
- -v : Prints statists of the decoding process to stderr
- -i infile : The file containing bytes to be decoded (default is stdin)
- -o outfile : The output file to store decoded bytes (default is stdout)

4.3 Error

The source code for this program is provided in the class resources repository. Error adds noise to a file and is used to the efficacy of the encoding when decoded.

- -e : Percent of file to randomly add noise

4.4 Entropy

The source code for this program is provided in the class resources repository. Entropy calculates the entropy, or variation of contents, for a file.

5 Parse

Program arguments are parsed and stored using flags. A set is not used since the order of the inputs is irrelevant and the parsed data will ultimately be stored in variables anyway.

After program arguments are parsed the input file is parsed with encode or decode. Execution happens with every grab of a byte to the buffer. Refer to execute for how the algorithm works.

6 Execute

6.1 Encode

For encode, the parsed byte is divided into an upper and lower nibble. Then, the lower nibble is encoded and the byte is written to file out. The same is done for the upper byte. This process is repeated until the entire file is parsed. The pseudocode is shown in *alg.* (1).

```
initialize encode matrix G;
while buffer grabs byte from file_in if not end of file do
    lower = lower_nibble(buffer);
    encoded lower = encode(encode matrix G, lower);
    write encoded lower to out file;
    upper = upper_nibble(buffer);
    encoded upper = encode(encode matrix, upper);
    write encoded upper to out file;
end
delete encode matrix;
close files;
```

Algorithm 1: Encode

6.2 Decode

Decode is (unsurprisingly) the opposite of encode. After one byte is parsed, it is decoded and stored in a variable last. Every even odd iteration, a byte is parsed and decoded like before, but instead of storing in last, the function pack_byte is called and puts the last nibble and current nibble together and writes to stdout. The pseduocode is shown in *alg.* (2).

```
initialize decode matrix Ht;
initialize analytic variables;
while buffer grabs byte from file_in if not end of file do
    increment processed bytes;
    initialize message pointer;
    error = decode(decode matrix Ht, buffer, pointer to decoded message);
    error handling;
    if processed bytes is even then
        last = msg;
    else
        decoded = pack_byte(msg, last);
        write decoded to out file;
    end
end
delete encode matrix;
close files;
```

Algorithm 2: Decode