Derick Pan
Dpan7

Design

notes:

m = message 4 bits c = hamming code 8 bits G = vector-matrix multiplication e = error syndrome	XOR for addition OR for multiplication

X	Y	AND(X,Y)	OR(X,Y)	NAND(X,Y)	NOR(X,Y)	XOR(X,Y)
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Hamming(8,4)
8 = Hamming Code
4 = message

Encode and Decode will both be in hamming.c

File is just a sequence of bytes,

BitVector

Struct:

- Uint32_t length // Length in bits
- Uint8_t *vector // Array of bytes. It's uint8_t because we extended the last bit to be p3

16 bit vector

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

Setbit(n)

- Byte =n/8
  - Now we have to locate the specific bit inside of that byte.
- Offset n % 8
  - byte= Byte (OR) 1 << (offset)
- One line: v-> vector[n/8] |= (1 << (n%8)

Ex.

Set bit (14)

Byte = 14/8 = 1 // Byte 1

Offset = 14 %8 = 6// Bit 14 is in byte 1 and bit 6 offset

To set the 6th bit

MSB	1	0	1	1		0	1	1	0	LSB
	0	0	0	0		0	0	0	1	Shift it 6 times
OR	0	0	0	1		0	0	0	0	
=	1	1	1	1		0	1	1	0	

To set a bit in a byte, shift a "1" to the index then bitwise OR operation

= byte w/ bit set

Bv_clr_bit(n):

- Get 1's in every position BUT the bit
- MSB 0111 0110 LSB, Shift to the position, Invert it = 1101 1111 AND it, to become 0101 0110.
- v->vector[n/8] &= ~(1<< (n%8))

Bv_get_bit(n):

- v->vector[n/8] >> (n %8) & 0x1

MSB 0011 0010 LSB LS AND 0001 0000 0001 0000 Now shift right 4 back to position	MSB 0011 0010 LSB Right shift 4, 0000 0011 AND 0000 0001 = 0000 0001

XOR_BIT:

- MSB 0011 1100 LSB

MSB 0011 0010 LSB LS to position XOR 0010 0000 0001 1100	

4x8 Matrix = 32 bits.
4x4 Matrix = 16 bits.
Bc we know num of rows and columns, we always know the amount of bits.

Bit Matrix:

Stores u32 rows
Store u32 Cols
Bitvector *vector

Bit Matrix serves as a wrapper over Bit Vector (~ one line
Bitmatrix = m rows, n columns
Rth row, Cth column
R*N + C

Bit matrix Create:

Bm rows

Bm cols

Bm set bit (rows, cols)

Bm from data - bm to data
Needed bc we need to multiply bit vectors/matrices
We need a way to convert bytes to specific matrix

Msg = MSB 0000 1101 LSB // We need to convert this into a bit matrix
We can loop through bits, clear bit, set bit.

Bitmatrix *M = bm_from_data(msg, 4(haming 4/8scheme))
To convert into bit matrix:
Code = MSB 1110 0000 LSB
Bit Matrix 8C = bm_from_data(code, 8)

Hamming encode
Allocate memory to Generated Matrix
To encode a message we need to multiply M*g
M = bm_from_data
C = m.g
return Code = bm_to_data

Hamming Decoding, Convert code into bit matrix

Code = bm_from_data()
Error syndrome = c*H^t
If e == 0
    No error return Ham ok
Else
    lookup(e)

Encoder Main()

    getopt() loop
    Create generator Matrix G // This is going to be a bit matrix
    while fgetc != EOF
        byte = MSB 1100 0110 LSB
        msg1 = low nibble of byte
        msg2 = high nibble of byte

    code 1 = ham_encode (G, message to encode); // 8 bits = 1 byte
    code 2 = ham_encode (G, msg2); // 8 bits = 1 byte

    We can use fputc // Writing a character OUT to a file stream
    and free memory later

Decoder Main()

    getopt loop()
    create H transpose (H^t)

    while fgetc != EOF
        When we're decoding, every byte is a code.

        How much code do we need to convert back to a single byte of data? 2.
        get code 1
        get code 2
        msg1= decode (H^t, code 1) // nibble
        msg2 = decode(H^t, code 2) //Nibble
        Now we have to combine them and pact them together using pack_byte.

Pact the nibbles into a byte.
fputc(packed nibbles)