Fabrice Kurmann fkurmann@ucsc.edu 3 May, 2021

CSE 13S Spring 2021 Assignment 5: Hamming Codes Design Document

Program Flow:

Encoder: Generate Hamming codes given input.

Accept command line options:

- -h print out program info message
- -i specify an infile other than stdin
- -o specify an outfile for the encoded codes only other than stdout.

Decoder: Decode the Hamming codes made by the encoder.

Print statistics (bytes processed, uncorrected and corrected errors, error rate <u>decimal to 6</u> <u>digits of precision</u>) <u>to stderr.</u>

Accept command line options:

- -h print out program info message
- -i specify an infile other than stdin
- -o specify an outfile for the decoded codes other than stdout.
- -v enables printing statistics to stderr

Matrices to store:

Generator matrix (G)

Transpose of parity checker matrix, (H^T)

Arrays/vectors to store:

Error reference table

Hamming code vector for input

Decoded vector for output

The Bit Vector ADT:

Represents a one dimensional array of bits.

For n items, use (n/8) + 1 uint 8s

Bitwise operations are required to handle the individual bits.

ADT variables:

Length, store as a uint32_t Vector, store as an array of uint8_t

Functions to implement:

by_create - return pointer to a bit vector, each bit is initialized to zero.

To initialize vector to zero, it's byte contents must all be zero.

bv_delete - destructor, free memory

bv_length - return length in bits, not bytes

bv_set_bit - set a specific bit to 1(?), crucial not to change any other bits*

bv_clear_bit - set a specific bit to 0(?), only change desired bit

bv_get_bit - get a specific bit's value

bv_xor_bit - XOR comparison of bv element i and given binary value

changing specific bit: byte = index/8

place in byte = index %. 8

Qet bit 13: = Q0101011010000101

byte
$$13/8 = 1.(timeste)$$

byte 13/8 = 5

points to 14th bit (at index 13)

Modify vector [byte_value] (bit value)

Breaking vint 8 into it's own array:

L setting a bit done by changing value of VINT.

Logical shifts
$$00110100_2$$
: 52
 00011010_2 : $2.8 \cdot 16 \cdot 26$
 00001101_2 : $1.44.8 \cdot 13$
 000001102 : 2.44 = 6 even, bit is 0

Logical shifts = using integer division on unit. - 3 times

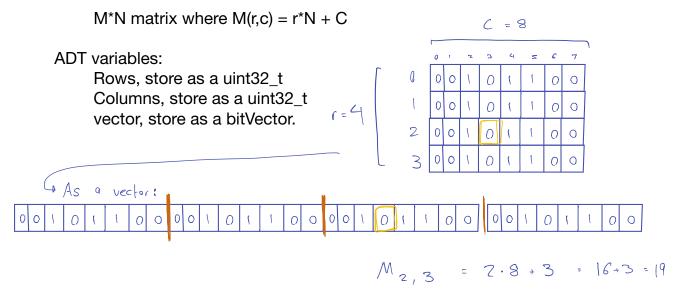
Wing addition to conduct bitwise xox : xox

bv_print - print the bits, separated by spaces, as they are sorted in vector array.

(Bit vector is read *left to right*) Bit 0 is leftmost, the lsb, unlike standard reading.

Bit Matrix ADT:

Represents a two dimensional array of bits - or an array of bit vectors.*



Functions to implement:

bm_create - return pointer to a bit matrix, number of bits = rows*cols
To initialize vector to zero, it's byte contents must all be zero.

bm_delete - destructor, free memory

bm rows - return num rows

bm cols - return num cols

bm_set_bit - set a specific bit to 1(?), crucial not to change any other bits* bm_clear_bit - set a specific bit to 0(?), only change desired bit bm_get_bit - get a specific bit's value

These 3 functions will simply call the bit vector functions for the correct bit, showcased

above.

*bm_from_data - make a 1 row bit matrix out of the first LENGTH number of bits in a matrix bm_to_data - return the first 8 bits of a bit matrix as a uint8_t Simply return the uint8_t in position 0 of the bit vector.

*bm_multiply - return multiply two bit matrices using matrix multiplication, return a new BM with the result.

NOTE, you multiply matrix A by matrix B MOD2!.

bm_print - print the bits, separated by spaces, as they are sorted in vector array.

The Hamming Code Module:

HAM_STATUS contains three outcomes:

 $HAM_OK = -3$ (no errors)

 $HAM_ERR = -2$ (Error not fixable)

HAM_CORRECT = -1 (Error fixed)

Ham functions:

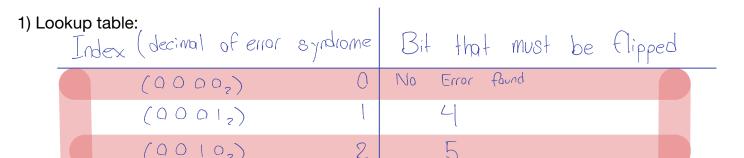
ham_encode - Given a nibble from a bitMatrix, adds the Hamming bits to return a fill byte (uint8_t) of Hamming code.

ham_decode - Return a nibble (decoded from a coded byte of Ham Code) to the beginning of a specified bitMatrix. Then also return the HAM STATUS from the decoding process.

Systemic Hamming Code:
$$(8,4)$$
Note, in the bt vector these bits are reflected.

Po: xor(Do Pr P3) Index
Pr: xor(Do Qr D3)
Pr: xor(Do Qr D3)
Processor (Or D2 D3)
Processor (Or D3 D3)
Fixample for 1 0 0 1 1 0 0 0 0 1

Prelab Questions:



		3
(0011_2)	3	Ham Error
$(0 00_3)$	4	6
(0101_2)	5	Ham Error
(01102)	6	Ham Error
$(0 1 1_2)$	7	3
(10002)	8	7
(10012)	9	Ham Error
(10103)	(0	Ham Error
(10112)	11	2
(11002)	12	Ham Error
(11012)	13	1
(11103)	14	0
(11112)	15	Ham Error

2) Error decoding:

2) Effort decoding.

(|1000|11|) • H^T = [|2333] %2 = [|0|1]

$$|10|_z = |3|_0 = Swap \text{ bit at 1}$$

Output = |100 + [|0|1]

2)
$$(00011011) \cdot H^{\tilde{1}} = [2121] \% 2 = [0101]$$

 $1010_2 = 10_{0} = HAM ERROR$
Output = unknown.