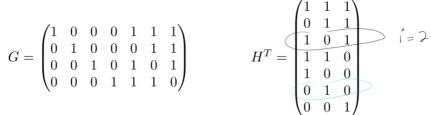
CSC 404 - ACTIVITY/PROJECT 14 - NAME: ()

Remark 1. The generating matrix G and transpose of the parity check matrix, H, in a [7,4] Hamming Code (modulo 2) is given by





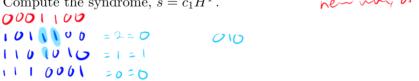
Problem 1. For the following, try to do a couple of them by hand. So, you can see the magician tricks in action.

- a. For a message $x = 1001 \sim (1, 0, 0, 1)$.
 - i. What is the resulting codeword c = xG? 11+00+01+ 11 = 2 = 0 11+01+00+11=2=0 11+01 + 01 + 10=1=1
 - ii. Show that the syndrome, $s = cH^T$, results in (0,0,0).

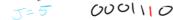
- b. Suppose you receive the codeword $c = 0111101 \sim (0, 1, 1, 1, 1, 0, 1)$
 - i. Compute the syndrome, $s = c_1 H^T$.

- ii. Determine the position, j, of the syndrome within H^T .
- iii. Change the jth bit in the received codeword. What was the original, k=4, bit message?

- c. Suppose you receive the codeword $c = 0001100 \sim (0,0,0,1,1,0,0)$ i. Compute the syndrome. $s = c_1 H^T$.



ii. Determine the position, j, of the syndrome within H^T .



iii. Change the jth bit in the received codeword. What was the original, k = 4, bit message?

Remark 2. You may have noticed this already, but there is a specific structure the matrix H^T takes. For r=4, each row/entry is a unique non-zero 4-bit number – i.e., it contains 0001 through 1111 – Neat! For H^T we place the entries with a single 1 on the bottom forming a 4×4 identity matrix and everything else lives on top. For a general r we construct H^T as $2^r - 1$ rows where each row/entry is a unique non-zero r bit number and place the identity matrix on the bottom!

We are actually free to rearrange the non identity matrix rows anyway we want (as long as we keep the same organization in the matrix G. An easy way to generate H^T is to cycle through numbers 1 to $2^r - 1$ and place their binary representation (with possible padded zeros) in to H^T making sure to place (append!) the single 1 terms in an 'identity matrix' shape on the bottom of H^T . See Replit Link for a version of this – I highly recommend trying to reconstruct a version of this.

Problem 2. (Bigger Hamming [n, k] Codes) Alright, let's work with some bigger Hamming Codes! How about we use r = 4 to allow $k = (2^4 - 1) - 5 = 11$ bit messages. (In general, $n = 2^r - 1$ and k = n - r.) Use technology to compute the following – you can do these by hand, but it is a pain :-).

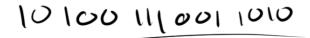
a. For the message x = 10101010101 compute the corresponding codeword c = xG. Then show the syndrome $s = cH^T$ results in 0000 (i.e., with no errors in the codeword we should have all zeros in the syndrome.).



b. Suppose you receive the codeword $c_1 = 101001110001010$. Compute the syndrome $s = c_1 H^T$.

c. Determine the position, j, of the syndrome within H^T . (See Above/Replit for H^T)

d. Change the jth bit in the received codeword and output the resulting k-bit message. (Bonus Fun - turn the resulting message back into a decimal number)



Mogic Mon ...