HAMMING CODE

CENG 325

Patnekar, Prarthavi P. - SDSMT Student

Contents

List of Figures

Description of Program

The program asks the user to specify the number of data bits in a message. Total number of parity bits are then calculated using this number. The program then generates G, H and R matrices based on the previous numbers. The G matrix is used to encode the data bits into a message. Then a random generator is used to generate a possible error in one of the bits of the message being send. So, the received message can contain 1 bit error. The program gets the parity bits and corrects the messages if needed using the H matrix and the parity check matrix. The matrix is then decoded using the R matrix.

Description of Algorithms

For getting the column that matches with the parity check in the H matrix, I go through each column until a match is found. The nested loop goes through each row of a column until there is a bit that doesn't match the parity check bit. If the second loop goes through all the rows in a column, it means the column matches the parity check vector and our work is done. The index of the matched column is the index of the error bit.

For converting from decimal to binary, we keep diving the number by two and storing the mod of the number by two.

To get better understanding of the hamming code, I utilized the "The Hamming Code with Matrices" article by Department of Electrical and Computer Engineering - University of New Brunswick, Fredericton, NB, Canada. The construction of my G and H matrices is based on the concepts mentioned the in the article above.

Description of Libraries

I used the armadillo library to help with construction and multiplication of the matrices required for the project. Armadillo is a linear algebra library (matrix maths) for the C++ language, distributed under the permissive Apache 2.0 license. I added it to the project using the NuGet Manager.

Description of Functions and Program Structure

The functions and structure utilized in the program are as follows:

GetUserInput

The function prompts the user for the number of data bits they wish to send. It then verifies if the input from the user is valid. If not, it prompts user for an answer again until a valid option has been entered. The function then returns that answer to the Main function.

CalculateParityNum

Calculates the number of parity bits required based on the number of data bits specified by the user.

DecToBinary

Converts a decimal input into a binary number of the required length and returns it in the form of a vector.

GenerateDataBits

Generates a vector of random data Bits whose length upon the number of databits specified by the user.The main then prints out the generated data bits as message.

GenerateGMatrix

The program then generates a G matrix based on the number of parity and data bits using this function. The leftmost columns of the matrix forms a (n * n) matrices where n is the number of data bits. Figure 1 shows the G matrix generated by this function for a message with 4 data bits and 3 parity bits.

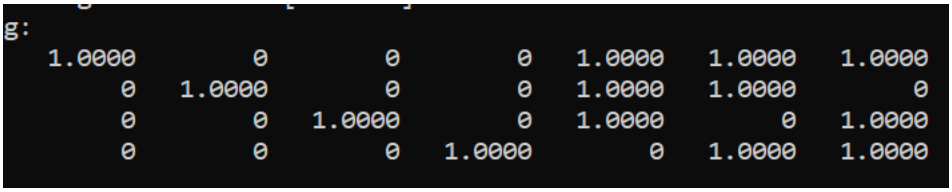

Figure 1: The G Matrix

GenerateXVector

Used to generate both the send vector and the decoded message vector. The function basically multiplies the two input matrices, mods the result by two, and returns the resultant matrix. To get the sending vector, g and p (message matrix) matrices are multiplied together. The main function then prints out the sending matrix.

GenerateError

The program then generates a possible error in the message through this function. The program randomly selects a bit from a range of 0 to the length of the sending message for error. If zero is selected, the message has no error is generated in the message. Otherwise, an error is introduced the randomly selected bit by flipping its binary value. The Main functions treats the value returned from this function as the received message. The main function displays the received message to the user.

ErrorCorrection

This function calls the generateHMatrix to get the H matrix. The H matrix is multiplied by the received message vector to get the parity check matrix, which is displayed to the user. The parity checks are then compared to each column of the H matrix, to find a match. The index of the matched column in the H matrix is the index of the error bit. So, that bit is flipped in the received message to get the corrected message. The corrected message is then displayed to the user. Even if there were no errors in the received message, it is still displayed as corrected message. The corrected message is returned to the main function.

GenerateHMatrix

Generates the H matrix. The rightmost columns of the H matrix form an identity matrix of (n * n) where n represents the number of parity bits. Figure 2 shows the H matrix generated by this function for a message with 4 databits and 3 parity bits.

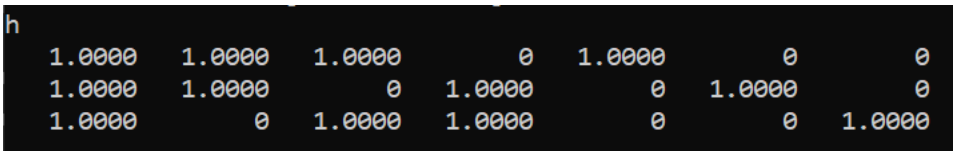

Figure 2: The H matrix

GenerateRMatrix

The function generates the R matrix, where the databits form the identity matrix. Figure 3 shows the R matrix generated by this function for a message with 4 databits and 3 parity bits. The resultant R matrix is multiplied with the received message to get the decoded message. The main function then displays the decoded message.

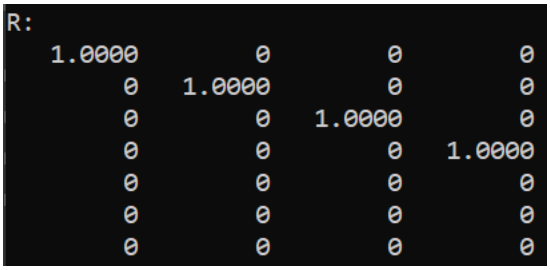

Figure 3: The R matrix

PrintBits

The function is used to print the matrices in the format specified by in the project document.

Description of Testing and Verification Process

For testing, I first manually entered the data bits instead of randomly generating data bits. I then tested the cases of having no error in the message, and then having an error in the first and the last bit. I then randomly generated the indices of error bit and tested those cases. Later, I randomly generated the data bits for the message of varying length and tested with random errors to ensure that the program was working correctly.

Description of Submission and How to Run

The submission contains this report along with a visual studio solution and folders for packages, project folder and other folders needed for the project. The main program name is Hamming.cpp To run the project, open the Hamming solution in visual studio and hit run. The program runs in while loop so if you want to exit the program, just close the console window. Note: The parity check here doesn't directly reflect the index of the error bit, instead it is used to get the index of the error bit. Also, the program checks for the user input, so if the user enters an invalid option, the user enters invalid characters, they will be prompted for a new databits number.

Level for Grading

- For an A [93 – 100 %]

Citation

Conrad Sanderson and Ryan Curtin.
Armadillo: a template-based C++ library for linear algebra.
Journal of Open Source Software, Vol. 1, pp. 26, 2016.

Conrad Sanderson and Ryan Curtin.
A User-Friendly Hybrid Sparse Matrix Class in C++.
Lecture Notes in Computer Science (LNCS), Vol. 10931, pp. 422-430, 2018.

"ECE4253 Hamming Code - a Matrix Approach." *Www.ece.unb.ca*, https://www.ece.unb.ca/tervo/ee4253/hamming2.shtml#:~:text=The%20Hamming%20code%20concepts%20can,result%20(syndrome)%20is%20zero. Accessed 14 Oct. 2022.