CSC 310, Spring 2004 — Assignment #4

Due 4:00pm April 8 (drop it off in my office, SS 6016A).
You're also welcome to hand it in at the start of lecture on April 7.
Worth 10% of the course grade.

Note that this assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own.

As discussed in lectures, maximum likelihood decoding of a linear code for the Binary Erasure Channel (BEC) can be done by solving a system of linear equations. The same method can be used for a channel in which data is transmitted in packets of a few thousand bytes, some of which may be lost (but which will be received correctly if they are received at all). We might, for example, distribute a movie to many receivers by broadcasting one million packets that are one thousand bytes in size, some of which may be lost. We might group the one million packets into one thousand blocks of one thousand packets, and encode a block of a thousand packets with some linear code, which operates in parallel on all the bits in the packet. (A related, and possibly better, method for solving this problem is described in Chapter 50 of MacKay's book.)

For an $[N, K]$ code, decoding for the BEC by solving a system of $N-K$ equations will take time proportional to $(N-K)^3$ using the most obvious algorithm. This time can be reduced somewhat by using cleverer algorithms, but for values of $N-K$ in the thousands, the time could still be quite substantial. One way around this is to use a code with a sparse parity check matrix, and to decode using an iterative "message passing" algorithm, which takes time roughly linear in $N-K$. Using a sparse parity check matrix reduces the error correction capability of the code somewhat, and the message passing algorithm is not guaranteed to always find the maximum likelihood decoding, but these disadvantages may be outweighed by the much lower decoding time.

In this assignment, you will write a program to implement such a message passing decoder (for the BEC, in which single bits are erased, not packets). You will try it out for a code and a set of received messages that are provided to you, and determine how many of these received messages can be decoded using the message passing algorithm. Finally, you should find out how many of the received messages that couldn't be completely decoded would have been decoded completely using a maximum likelihood decoder.

The message received from a BEC is a vector of N symbols. Those received symbols that are 0 or 1 are equal to the corresponding transmitted symbol. An "erasure", which will be represented as symbol 2, can be received when either a 0 or 1 was transmitted. The decoder attempts to replace the erasure symbols with the symbol (either 0 or 1) for that position in the codeword that was transmitted.

The "message passing" decoding algorithm can be visualized in terms of messages passed from bits in the codeword to parity checks, and from parity checks to bits in the codeword. If the correct value for a bit in a codeword is known — either because it was received correctly through the channel, or because its value has been determined in the decoding process — this bit in the codeword sends a message to all parity checks of which it is a part, telling these parity checks what its value is. A parity check waits until it has received messages from all but one of the bits that participate in the parity check, at which point it can send a message to the remaining bit, telling it that its value must be the value needed for the parity check to come out correctly (given the values of the other bits, which are now known). This process of exchanging messages continues until no further messages can be sent. At this point, the codeword bits may all be determined, in which case decoding was successful, or some codeword bits may still be unknown, in which case decoding was not completely successful (though some of the erased bits may have been filled in with their correct values).

Here is another equivalent way of thinking of the algorithm. Let the vector r, of length N, initially contain the received symbols, some of which are equal to what was transmitted (0 or 1), but some of which may be erasures (represented as 2). Let the code be represented by the parity check matrix H. The decoder looks repeatedly at the rows of H, which correspond to parity checks, and may change r if this parity check now allows an erased bit to be determined. This process continues until no further changes to r are being made. When the decoder looks at a row of H, it first checks how many of the bits participating in this parity check (identified by the 1s in that row of H) are still erasures. If exactly one bit is an erasure, the decoder changes this erasure in r to the correct bit, which is the sum (in Z_2) of the other bits participating in the parity check (since this will result in the parity check being satisfied). This procedure is less efficient than passing messages, since it keeps looking at parity checks even when they are no longer needed, and since it doesn't take advantage of the sparseness of H, but that's OK for this assignment.

You should write a program to implement this procedure (in any common programming language). Your program should read the K by N matrix H from a file (as a series of 0s and 1s, in order from left to right and top to bottom). It should also read a series of vectors, r, of length N, in which the elements are 0, 1, or 2 (representing an erasure). For each such received vector, it should apply the decoding algorithm described above, and print the result, which may still contain values of 2 if the decoding was not completely successful. If the message was decoded completely, the result should be a codeword, which satisfies all the parity checks specified by H. You should check whether or not this is true, and print a message if the decoded vector has no erasures, but is not a codeword.

The file `/u/radford/310/a4-H` on CDF contains the parity check matrix, H, for the code you should use. This code has rate 3/7, with $N = 35$ and $K = 15$, so that H is a 20 by 35 matrix. Your program should be able to handle any linear code, but you may fixed the values of N and K as constants in your program (rather than allowing them to be set at run time). The file `/u/radford/310/a4-r` contains 50 received messages, obtained by erasing the bits in 50 randomly chosen codewords with probability 0.3. These files are also available from the web page.

You should try to decode the 50 messages in `a4-r`. Those that are decoded completely should be codewords. For those that aren't decoded completely, you should determine whether or not a maximum likelihood decoder could have decoded them completely. You can do this by manually changing one of the bits in the message that the decoder left as an erasure to 0, and then to 1, and for both of these, try decoding again. If you find that both of these altered messages can be decoded to codewords, then no decoder could have determined which of these codewords was actually transmitted. If you find that one of the messages is decoded completely to a codeword, but the other ends up not being a codeword, then you can conclude that the codeword found is the only codeword that could have produced the received vector, and that a maximum likelihood decoder could therefore have decoded this message completely.

You should hand in a listing of your program, which should be written in good programming style, the results of decoding the 50 messages in `a4-r`, and the results you used to determine for which of these messages a maximum likelihood decoder would have succeeded.