Asgn5 Writeup by Derick Pan
Derick Pan - dpan7
Cse13s

- Graphs showing the amount of entropy of data before and after encoding it.
- Analysis of the graphs you produce.
- show error rate and stuff and entropy

Firstly, let's define a few terms and the things that's going on in this writeup.

What is Entropy?	Entropy, as defined by Claude Shannon, quantifies the amount of information produced in a process. Simplified to terms of this assignment; Entropy signifies how much chaos we have in a file. A file of all 0's would have an entropy of 0, because there is no chaos. Whereas, a file of repeating "01" would have a lot of chaos.
What does the "error rate" mean?	In this assignment we have a program called error that injects noise into our hamming codes. By injecting noise, I mean flipping a % of all the bits in the file. An error rate of 1% flips all the bits, and an error rate of 0% flips none the bits.
What is a Hamming Code?	Hamming codes is a linear error-correcting code able to decode and correct data even when there's a lot of interference.

First, we look at a few Entropy data points of files before and after encoding it.

Entropy of Random	Entropy of Bible.txt	Entropy of world192.txt
Before encoding: 5.999488 After encoding: 3.534526	Before encoding: 4.342751 After encoding: 3.341338	Before encoding: 4.998314 After encoding: 3.504212

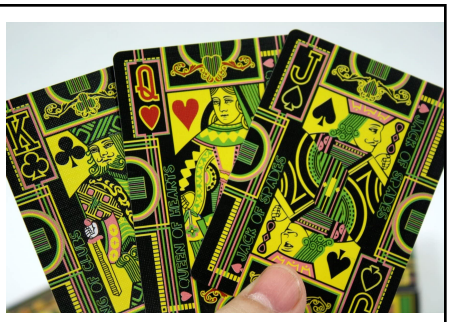
All three entropy values are different because all of the text files are different.

We Notice a common theme that there's less entropy after encoding the file, but why is that? I will use an example to explain my idea.

For example, I have two standard 52-card decks of cards in order.

The deck on the left is very clean and simple.

The deck on the right has each card individually designed with complex color mixing.



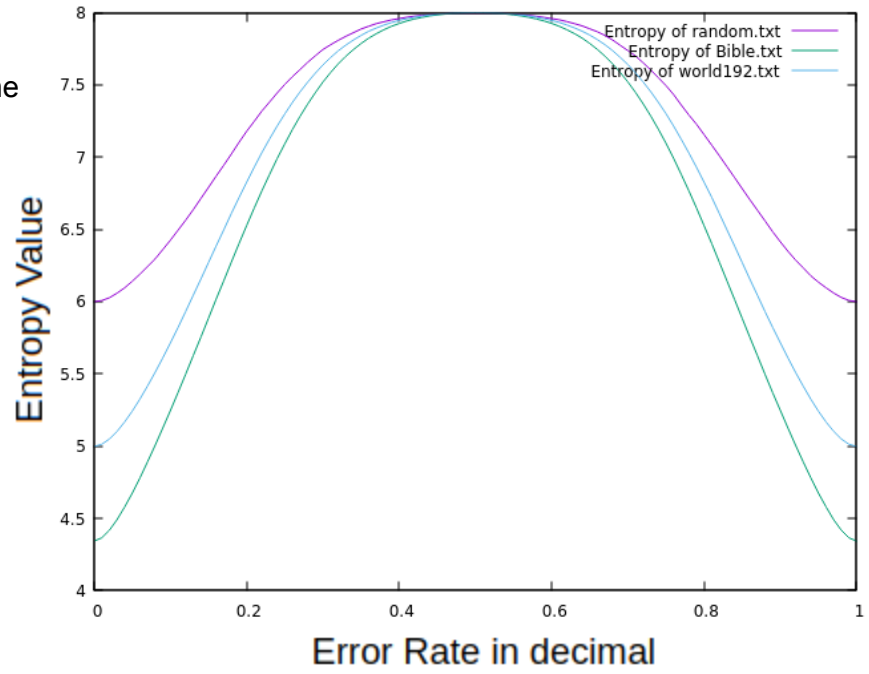
It's clear which deck will have higher entropy based on it's design; The deck on the right. Although, both decks should have the same entropy because all the cards are still in order. But the entropy is different just like how the entropy is different before and after encoding the text files. I look at the entropy on the representation of bits (how the card looks), and not the actual value of it (Value of Card). The reason why the encoded data doesn't look at the value of the bits is because it hasn't decoded it yet.

TLDR: the representation of the bits is what's cared about and not the value. Therefore, entropy values will be different.

Next up, we look at the graph to the left.
The X-Axis represents the error rate injected into the decoding function. While, the X axis represents the entropy value because Error Rate in decimal.

We see that it's in the shape of a bell, but why does the error rate go higher and go lower, mirroring each other in that manner?

##########################
btw, yes this graph took my computer FOREVER to create
##########################



Let's have for example a stream of text "00 0000 0000"
If untouched, that stream of text will have an entropy of 0. There's no chaos, no disorder; it's the exact same. Whereas, if we inject an error rate of 50%, the stream of text will look like "01 0101 0101" .Meaning that 50% of the bits have been flipped, and now the entropy goes as HIGH as it ever will. Then if we inject an error rate of 100%, the stream of text will just be "11 1111 1111." There's absolutely no chaos in this stream of text as it's all the same thing.

If we inject an error rate of 20%, the file will have 20% of all bits flipped 00 0000 0000 ➡ 00 0010 0100	If we inject an error rate of 80%, the file will have 80% of all bits flipped. 00 0000 0000 ➡ 11 1110 1110
This means that 20% are 1's and 80% are 0's	This means that 80% are 0's and 20% are 1's

The two share a correlation, because entropy != % changed. We see that the majority of the bits are one value, and the minority of the bits are another. The only difference between the two is if it's a 1 or a 0.

A real life example:

If we're renovating a house, it starts off very peaceful (no entropy). Once you start taking down walls (entropy up) ,things become chaotic and messy (entropy up) . Once all of the demolition is done, no more chaos could happen (entropy plateau). Then we start cleaning up(entropy down), and painting the walls (entropy down). Until everything is back in place and peaceful like it once was before.

Similarly, to the graph, chaos goes up but it also goes down.

Furthermore, I made another graph to support my point.
The following graph shows alphabet.txt being encoded with an Error rate (X-Axis), and the number of Hamming codes that could and could not be corrected (Y-axis).

At 50% error rate, (The most entropy we'll ever get) We see the graph at it's median point of having the MOST chaos, MOST uncorrected Hamming codes, and MOST corrected Hamming codes. The reason why we're still able to correct errors even at 50% error rate is because of Hamming(8,4). In my implementation, the first four bits constitute the message, and the last four bits constitute the data. More times than not, I will be able to correct the majority of the errors injected into the program.