Alex Childers

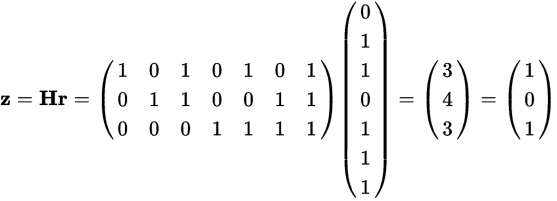
05/12/2019

If I were to rate my work on this assignment, I’d give it an 8 out of 10. I made an effort to incorporate some bottom-up programming to my workflow, as well as explicitly outlining my algorithm before I attempted to code it. I think that these, in addition to (finally!) incorporating abstraction with the use of separate .h and .cpp files, made the program a lot better.

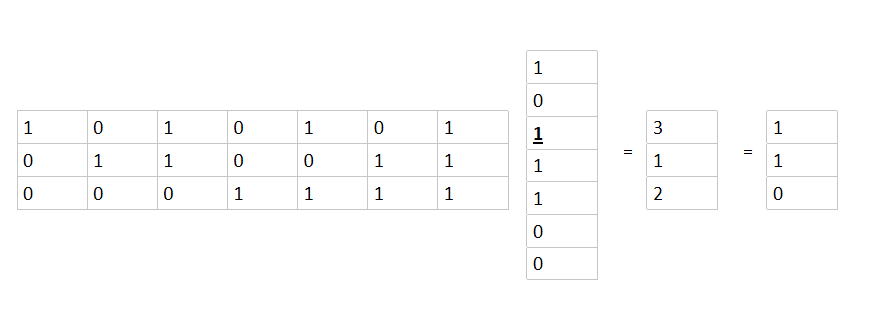
While I think this is one of the most difficult assignments we’ve had this semester, I also thought it was more fun. At the beginning, I had no clue where to start—we weren’t given as much starter code as in previous PAs, so I was left to do research on my own. As such, one of the difficulties I had was just figuring out how to convert chars to binary values, as well as inputting and outputting these values to file. However, I also think that, because this assignment was more difficult, I was forced to put a lot more planning into this than I would do normally under a time crunch. (I’m guilty; I started this assignment pretty late.) And I actually think this planning actually made the process a lot faster. I found myself oscillating between top-down and bottom-up design as needed—the primary encodeFile and decodeFile functions were planned in a top-down manner, but *their* helper functions tended to be written bottom-up, if that makes any sense. I think that, too, was a result of carefully structuring out my pseudocode before writing C++.

Since this assignment was more difficult, with less pre-written code, I do think I felt a lot more invested in finishing the assignment, and I know I’ve learned the material better than I did on previous assignments, where the algorithms were coded out for me. Struggling with how to convert to/from binary and writing all the helper functions made it so worth the effort when I finally implemented file encoding. Frankly, I was euphoric when I finally figured out how to encode a file. I really don’t think I’ve ever been that excited about finishing a segment of code before.

I did struggle with debugging the provided matrix multiplication function, and with converting the parity vector to a number. I don’t know if I was misusing the multiplication function, but I had to change the conditions of some of the for loops to avoid out-of-bounds index errors. It took me a while to figure that out, and eventually I just looked up an outside solution to matrix multiplication, which clarified the problem and helped me understand what needed to be fixed. If I recall correctly, we tested that function on two 2x2 matrices in class, which may have obscured any i/j/k related bugs. Additionally, it took me quite a while to figure out how to translate the 3x1 parity vector Z. In the PA #6 instructions, the following example for an error is given:



with the caption, “Rotating Z yields {1, 0 ,1}. Treating this as a binary sequence yields the number 5.” I erroneously thought that I needed to rotate Z counterclockwise 90 degrees to obtain the binary sequence. So, in trying to correct an error in 0x5C, I got the following:



Instead of rotating it clockwise to get 3, I rotated it left to 6. I really should have looked at my notes for another example of rotating the parity vector, but I figured it out eventually. I don’t know if my program decodes garbled text correctly, since I was unable to decode the newline character in the given sample\_garbled.coded.txt, but I figure my final product is close enough.