The working backwards method seemed promising at first, but I cannot figure a way to parse the returned data or a way to reformat the returned data in a way that can be parsed more easily. The idea is when we work backwards to truthvalues we have, we want to be able to plug in each possible circuit for each truth value and see which combination will make the smallest circuit. I will need to think this over a bit more.

I made a function that reformats the data returned by working backwards (reformat). Hopefully this will make it easier to use.

I made a function the converts that reformatted data into all possible circuits and returns a list of the smallest circuits for each piece of reformatted data (makeMinCircuit). Then I incorporated it into the wrapper function (combine). I will run it and see how long it takes to find 01101001 with maxDepth=3 and how many gates it finds for it. It may take a while; I estimate 35 minutes. I included print statements to show the progress. It is at least able to find a circuit with 10 gates. Remember the number of gates we are hoping for is 8.

It found the 8 gate circuits, but it is not done running. I will let it keep running to see how long it takes. It took maybe 2 minutes to find the 8 gate circuits. Unfortunately the program doesn’t know when we find the minimum so it doesn’t know when to stop.

While this runs I will go through the edX videos, Jing said to look at.

The program took 1972.70700002 seconds or approximately 33 minutes. I added options to stop the program mid-way and will test it out on a couple other truth values. It gives the option to stop midway if we found a set of circuits with 1 more gate than what we went up to going forward because this must be the minimum. It also gives the option to stop approximately every minute if it has been running for more than 5 minutes. Note that it only gives these options if you tell it to. It also does not seem to find all the circuits that would be found through brute force going forward even if you increase the depthLimit. It doesn’t seem to have had any problems yet, but the gateCounter used may need to be switched to my method instead of Bryan’s method. This will prevent the chance of it calling something like “(((a.b).(a.c)). ((a.c).(a.b)))” 5 gates when it is actually 4 gates. I am not sure if this will come up though. I made a modified combined that can quickly and be run for all 256 inputs at once and stopping if it spends more than 200 seconds on an input (hoping that it would have reached the minimum number of gates for that input by then. I will compare the gate counts for each truth value once it has found at least one circuit for each truth value. \*I need to look into making a minCircuitFinder that saves all circuits rather than just the useful ones (minCircuitFinder6). This should be a very simple modification. It seems as though it is able to find all the circuits with this modification (combine3). I have not done an extensive test however.

Back to the RNA seq. I am downloading an Integrated Genome Browser from

<http://bioviz.org/igb/download.html>

to read .wig files.

It doesn’t seem like the application will read the files I want it to read.

For tomorrow:

In GateMinimization2 folder check the “greedyCombine2Circuits.txt” file and the “truthValue and number of gates (greedy).txt” with the “6to9.txt” file and see if each of them have the same number of gates. Check to make sure the truth values in greedyCombine2Circuits.txt file are what they say they are. Check if the last python window ever finishes. This should theoretically give all circuits for all truth values if I did my calculations correctly. Continue working on speeding up the program. Focus more on RNA-Seq.