See if keeping in the dictionary all of the things that also have the same truth value and number of gates as the minimum circuit will be useful. Do this by looking at what was removed that wasn’t supposed to be. Get its truth values and number of gates. See if that matches the number of gates of the circuit that corresponds to that truth value.

Also it seems like of the things thrown out. We do not mistakenly throw out anything that contains a 0. I hypothesized that anything with a 0 is either useful now or will never be useful.

This may not necessarily be true because this 5 gate circuit ((((0.c).(0.b)).a).((0.c).(0.b))) would probably get thrown out because it isn’t useful when it’s made but is useful in making 00011110 at 8 gates and one other truth value at 7 gates.

Changing the dictionary to mapping the truth value to a list of circuits instead of just the one circuit may be our best bet. These four circuits would have been determined to be unused by that method but are actually used as determined by a starting point at 7 gates:

(((b.a).c).(0.a))

(((b.a).c).(0.b))

(((c.a).b).(0.a))

((((b.a).c).a).((b.a).c))

((((b.a).c).b).((b.a).c))

((((c.a).b).a).((c.a).b))

I am going to work on making a version of the min circuit finder that saves every circuit for a truth value with the min number of gates. So the dictionary would map truth values to a list of circuits rather than just one circuit. This will be saved under GateMinimization2. GateMinimization3 was moved to GateMinimization.

I made a function that checks what was unused in GateMinimization2 and a wrapper function. It seems that going from some start to some end is a lot slower, but it doesn’t lose any truth values. At least it didn’t while going from 4 to 5, 4to 6, and 6 to 7. It did however lose some circuits associated with some truth values while going from 4 to 6. So starting too far away may lead to eventual loss of truth values. This will still be useful, assuming we don’t lose any truth values, in finding out truly how many truth values can be found using 8 gates. \*I am running the wrapper function from 1 to 7 to check the plausibility of this assumption. We won’t have all of the circuits for them but we should have *some* circuit for each truth value. I built a stock file for what the true dictionaries should look like from 1 to 6. I have a program running still that will build a stock for 7 and potentially 8 if I leave it long enough for 8 to finish.

\*189 truth values were found for 6 gates when starting from 1 as opposed to the 192 it was supposed to have. Strangely, by 7gates, it was able to find all 228 still. This means starting too far away will lead to missing truth values, but hopefully 6 is close enough. When the stock file for 7 becomes available, I can try starting from 7 to double check.

While programs run I am going to attempt to find some way of eliminating what’s not needed while keeping what is needed.

!!! 250 truth values were found using 8 or less gates as opposed to the 247 from before. These are the 4 extras it found from the 246. We already knew 01101001 could be done with 8 gates though.

01101001

01101011

01101101

01111001

It just occurred to me that we really only need to save the min circuits for the 80 rearrangement groups, because the circuits for things in the same group are the same but just with inputs rearranged. However I am not sure how this would affect the ability of the program to find circuits especially with is declaring things to be unused.

I will begin working toward recreating this to work with AND. This will make the minimum circuits while using only ANDs and NOTs. A minimum circuit is defined as something that uses the minimum number of AND gates.

Using the file for nor and modifying it, I was able to get the getTruthValue(circuit) and gateCounter(circuit) functional for circuits that only have AND and for circuits with AND and NOR in different combinations. Except if you nor with 0 instead of using ‘ it will count it as one gate. If you use ‘ instead of nor with 0 it will not count it as a gate. Got the minCircuitFinder working. I believe I also got the getFromFile working but this hasn’t been tested. If this is working then the wrapper function that finds the unused circuits should also be working. I will set it up to create the stock files for 0 to 6 overnight. Looking at the number of circuits with 4 gates, I doubt it will finish by tomorrow.