Eric Altenburg, Hamzah Nizami, Connie Xu | CS370 | April 14th, 2020

Our algorithm works on 80% of the test cases based on the shell script that we were given. Our algorithm fails the test cases from test case 17 and onward. We attempted to implement a dynamic programming solution with backtracking and the issue that we have come across is that it does not find duplicates that well and it seems to get overwhelmed if the number of resistors is over 5.

This algorithm is best described as a rendition of subset-sum as we are attempting to turn this problem into subset-sum. Based on the formula, we can manipulate the denominator to be the "sum" we are trying to achieve with the given resistor values. We implement the DP subset-sum code given to us in the modules, with a slight modification to allow for duplicates by including the ability to check if a given column evenly divides into a resistor. When tracing simple test cases by hand, this has shown to work, but the algorithm doesn't seem to properly scale in searching for solutions. Under the assumption that we have our table filled out correctly, we check to see if a given target is capable of being created with a given set of resistors, then backtrack as per the python code's (subsetsum_dp.py) specifications. If the vector of floats we obtain through backtracking is within the bounds of the number of resistors allowed, it is added as a potential solution. Then with all potential solutions in one vector, we search for the one with the smallest percentage error and return that. Theoretically, we believe that this would work. We are just having problems implementing it. We believe there is an issue with the modification to allow for duplicates as test cases 17+ return many repeat resistors. We tried remedying this issue by potentially removing these duplicates and replacing them with a combined inverse sum as seen on line 245, though nothing is printed so we assume there is some possible floating-point error preventing the code from running to completion. Due to the time constraints, once encountering this issue, not enough time was remaining to investigate and fix our solution further.

We tried various methods in an attempt to get this to work and we have been working on several different algorithms to try different ways of getting our solution. The solution that we submitted is the one that passes the most test cases but this solution makes less sense than the one that we believe we have a concrete understanding of. The latter of which is an explicit implementation of Dr. B's subset DP code (this implementation does not allow for repeats), and because of this, the algorithm attempts to match the closest combination of resistors within the bounds of the tolerance with no repeating resistors.