Here we present an overview and pseudo-code for the program which we used to discover and check our results. The program can be found at <https://github.com/LoyolaResearch/SimpleSandwichMatrix> and is publicly available. We implemented these algorithms in C++, using Microsoft’s Visual Studio™ to afford high performance and fidelity.

Our program uses mostly brute-force approaches to generate all possible iterations of the *BZS C0S* semigroups. However, we developed a few optimizations to simplify and speed up the processing. First, we dynamically stored and generated all possible

Add pseudo-code here

// Main variables…

unsigned long long int count = power(2, leftY \* leftX);

signed char left[count][leftY \* leftX]; // (dynamically allocated)

signed char center[count][leftY \* leftX]; // (dynamically allocated)

signed char middle[count][leftY \* leftX]; // (dynamically allocated)

// generate all possible binary matrices depending on dimensions

for (i = 0; i < count; i++)

for (j = 0; j < leftY \* leftX; j++)

// This program will ask the user for how many rows and columns that they want their multiplied matrices to have, ask if they want it printed in the console or a file, generate all possible matrices of those dimensions that are in the completely simple zero semigroup and follow our requirements along with the corresponding sandwich matrices of the opposite dimensions that fit our requirements, then perform matrix multiplication on all possible combinations.

print first opening question

"How many rows would you like in your outer matrices: "

<user enters number of rows>

print second opening question

"How many columns would you like in your outer matrices: "

<user enters number of columns>

print output destination question

"Would you like to output your sandwich matrices to the console or a file (c/f): "

<user enters c or f>

if “c”

set outputToFile boolean to false

else if “f”

set outputToFile boolean to true

print name of file question

"Please enter a filename in the current directory for the outer and sandwich matrices: "

else

print error message

"Error: invalid input (should be c or f)"

end program

for loop

declare sandwich, left, and right matrices

end loop

nested for loop

initialize temporary matrices to hold possible valid matrices

end loop

nested for loop

if a row of a matrix only has zeros

set isValid to false

end loop

nested for loop

if a column of a matrix only has zeros

set isValid to false

end loop

if isValid is true

use the temporary matrix as a valid sandwich matrix

nested for loop

initialize valid left and right matrices

end loop

start triple-nested for loop

perform matrix multiplication on the valid matrices

end loop

if user chose to output to a file

print summary output to a file named summary.csv

end program