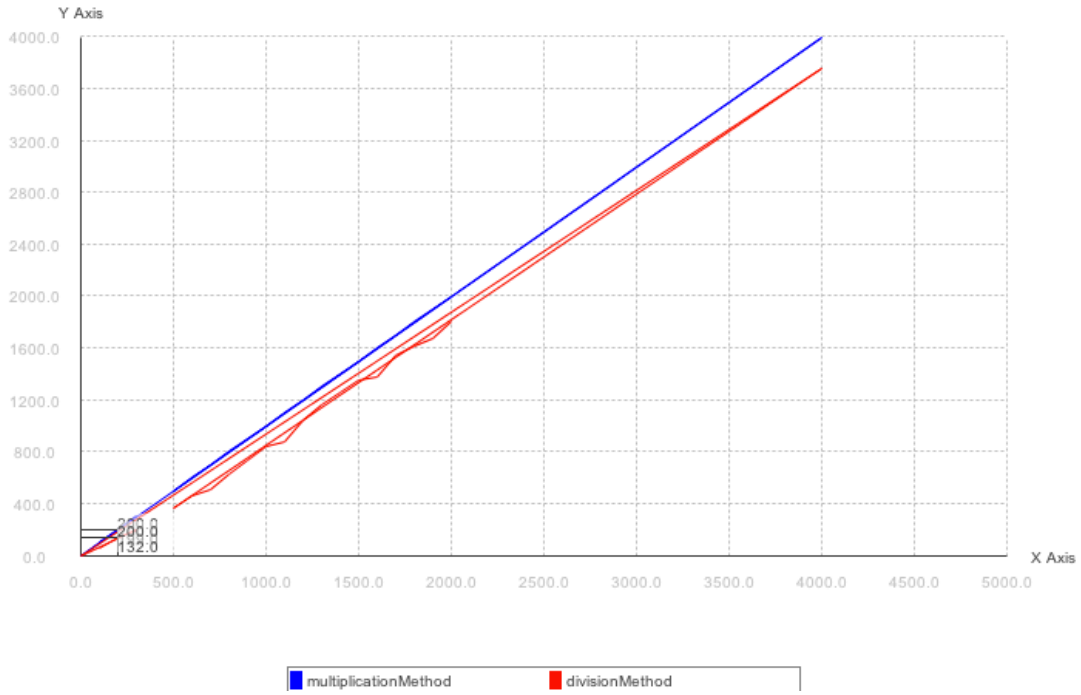
Lucia BERGER
COMP251
A1
260473661

Q1: Your second task will be to try several (well chosen) set of values for w, d, and m. Use your results to decide (and justify with graphs) which method is the best. Discuss the behaviour of the hash functions with variation of the parameters w, d, m, and a.

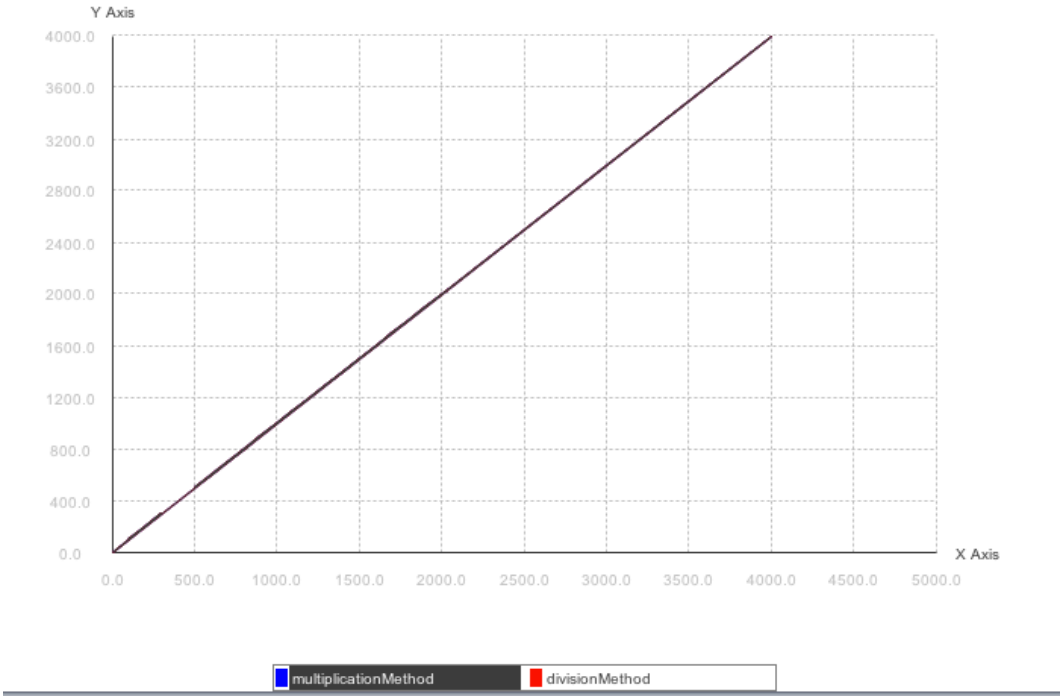
MULT	DIV
99.0	54.0
199.0	120.0
299.0	180.0
0.0	0.0
3999.0	3789.0
499.0	344.0
599.0	431.0
699.0	518.0
799.0	604.0
899.0	741.0
999.0	804.0
1099.0	912.0
1199.0	1009.0
1299.0	1079.0
1399.0	1185.0
1499.0	1277.0
1599.0	1405.0
1699.0	1564.0
1799.0	1644.0
1899.0	1720.0
1999.0	1773.0

My results show that multiplication results had less collisions and overall better results. However, it is worth noting that the differences were not great. In fact at the base value of 32 8 1000, there is little difference as shown on the graph below.



However, lets try another case:

At *d = 42* and *w = 12,* we examine little to no difference between the two methods. This can be attributed to the methods being fairly similar together. When d and w are very different, we'd presume them to be different.



The last assessment we will look at is the values of r and m, which would be the minimal number of collisions as they equate each other. We see this too:

MULT	DIV
99.0	109.0
199.0	109.0
299.0	299.0
0.0	10.0
3999.0	3999.0
499.0	499.0
599.0	599.0
699.0	699.0
799.0	799.0
899.0	899.0
999.0	999.0
1099.0	1199.0
1199.0	1199.0
1299.0	1299.0
1399.0	1399.0
1499.0	1499.0
1599.0	1599.0
1699.0	1699.0
1799.0	1799.0
1899.0	1899.0
1999.0	1999.0

Our final conclusion can be determined that in practice and implementation, there is not much difference between the two methodologies. While in theory, divisionMethod should result in a more efficient graphs with less collisions, we did not find that with our code.