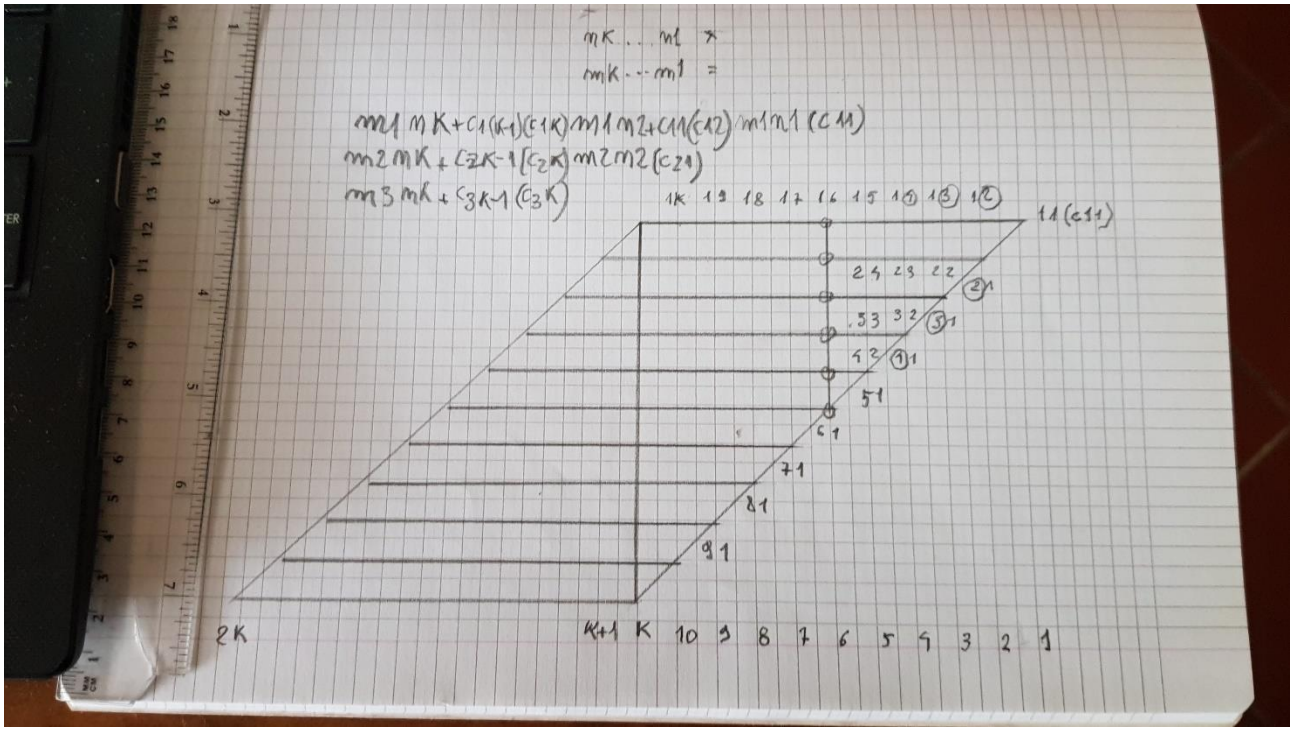
Inverse Long Multiplication Program in the Factorization of Semiprime Numbers

by Silvio Gabbianelli

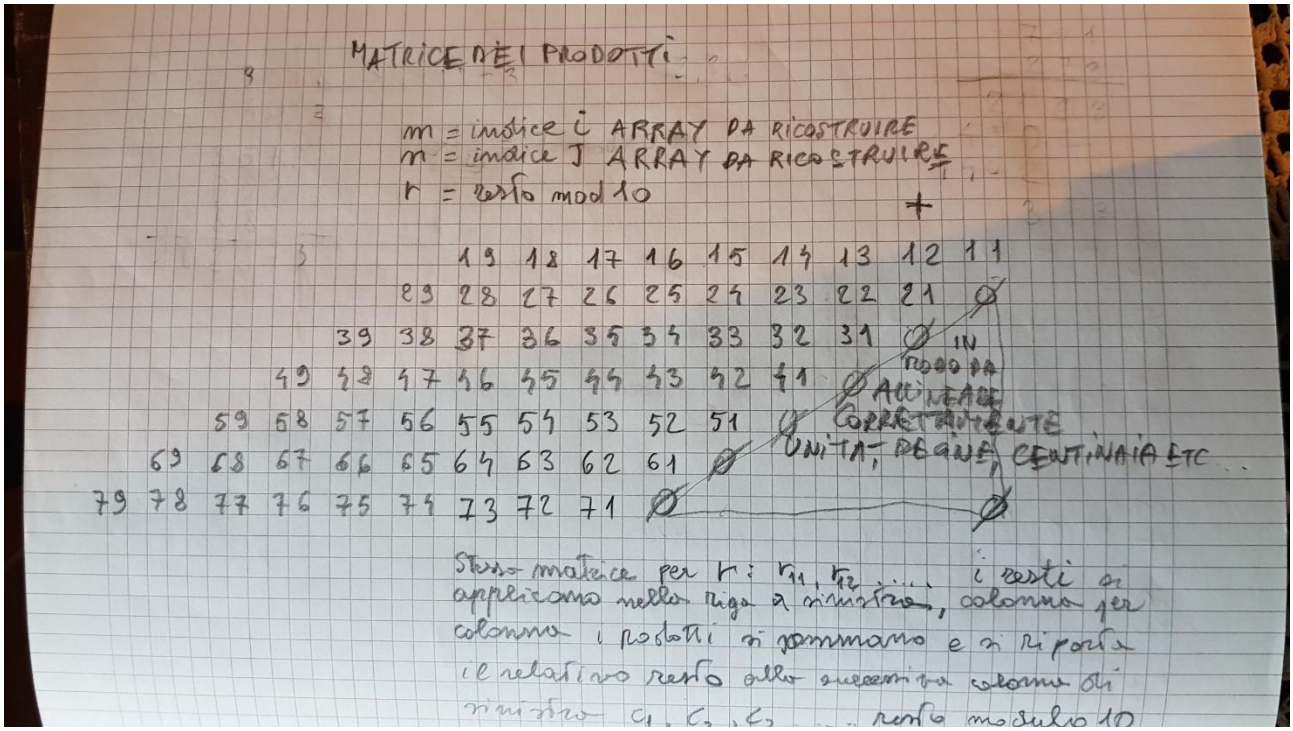
Abstract: Semiprime numbers are natural integers obtained from the multiplication of two prime numbers. In attempting to solve the factorization of a semiprime number, I considered the inverse procedure of the long multiplication method we learned in elementary school. After careful reflection, I found it to be both useful and feasible. The only known element in this procedure is the result (which must be an odd integer semiprime). We need to find the two sets [n1,n2,...nk][n1, n2, ... nk] and [m1,m2,...mk][m1, m2, ... mk] that will represent the two unique multiplicands that produce the known result. These sets are reconstructed through a triangular matrix, where the rows represent units, tens, hundreds, and so on, and the columns grow and then shrink in size. This matrix allows us to identify the necessary multiplicative pairs by ensuring the result satisfies the condition of being an odd integer that doesn't end in 5.

Introduction: Semiprime numbers are integers obtained by multiplying two prime numbers. I tried to solve the factorization of a semiprime number by considering the inverse of the long multiplication method taught in elementary school. This approach seemed useful and feasible after thorough consideration. The only known element in this procedure is the result, which must be an odd semiprime number. We need to find two sets [n1,n2,...nk] and [m1,m2,...mk] that represent the unique multiplicands yielding the known result. These sets are reconstructed using a triangular matrix, where each row represents units, tens, hundreds, etc.



Procedure: The algorithm begins by identifying possible pairs for position1, which when multiplied, result in the rightmost digit of the known result's units. The procedure continues, introducing two new elements per column until reaching column kk, ensuring the sums correspond

to the digits of the known result. If the sets' elements match correctly, their product equals the known result. Otherwise, there is an error in the matrix reconstruction.



After the exact procedure was identified, I wrote a Python code to automate the described steps. Given the challenges encountered and my amateur programming skills, I asked my friend Rob Barulich, an expert programmer, to verify the feasibility of the code. After some preparation and personal experimentation, Rob produced a first functional code and is now working to a better refined script to avoid exponential time in finding factors for a large semiprime, news will follow soon. Here an example of the results I obtained with my code with some intermediate printed values:

Result? 162373

Valid combination for column0, z=1, y=3, z_values=[], y_values=[], result=3, row_carries=[0], col_carry=0

Valid combination for column1, z=0, y=7, z_values=[1], y_values=[3], result=7, row_carries=[0, 0], col_carry=0

Valid combination for column2, z=0, y=3, z_values=[1, 0], y_values=[3, 7], result=3, row_carries=[0, 0, 0], col_carry=0

Valid combination for column2, z=1, y=0, z_values=[1, 0], y_values=[3, 7], result=3, row_carries=[0, 0, 0], col_carry=0

Valid combination for column2, z=2, y=7, z_values=[1, 0], y_values=[3, 7], result=3, row_carries=[0, 0, 0], col_carry=1

…….

Valid combination for column0, z=7, y=9, z_values=[], y_values=[], result=3, row_carries=[6], col_carry=0

Valid combination for column1, z=0, y=3, z_values=[7], y_values=[9], result=7, row_carries=[2, 0], col_carry=0

Valid combination for column2, z=0, y=3, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[2, 0, 0], col_carry=0

Valid combination for column2, z=1, y=6, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[4, 0, 0], col_carry=1

Valid combination for column2, z=2, y=9, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[6, 0, 1], col_carry=1

Valid combination for column2, z=3, y=2, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[1, 0, 2], col_carry=1

Valid combination for column2, z=4, y=5, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[3, 0, 3], col_carry=1

Valid combination for column2, z=5, y=8, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[5, 0, 4], col_carry=1

Valid combination for column2, z=6, y=1, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[0, 0, 5], col_carry=1

Valid combination for column2, z=7, y=4, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[2, 0, 6], col_carry=1

Valid combination for column2, z=8, y=7, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[4, 0, 7], col_carry=1

Valid combination for column2, z=9, y=0, z_values=[7, 0], y_values=[9, 3], result=3, row_carries=[0, 0, 8], col_carry=0

Valid combination for column1, z=1, y=6, z_values=[7], y_values=[9], result=7, row_carries=[4, 0], col_carry=1

……...

Valid combination for column1, z=9, y=0, z_values=[7], y_values=[9], result=7, row_carries=[0, 8], col_carry=0

Valid combination for column2, z=0, y=5, z_values=[7, 9], y_values=[9, 0], result=3, row_carries=[3, 0, 0], col_carry=1

Valid combination for column2, z=1, y=8, z_values=[7, 9], y_values=[9, 0], result=3, row_carries=[5, 0, 0], col_carry=2

Valid combination for column2, z=2, y=1, z_values=[7, 9], y_values=[9, 0], result=3, row_carries=[0, 0, 1], col_carry=2

Valid combination for column2, z=3, y=4, z_values=[7, 9], y_values=[9, 0], result=3, row_carries=[2, 0, 2], col_carry=2

Solution found: z=397, y=409

Total time: 4.125816106796265

Results:

- Result: 162373 | Solution: z=397, y=409| Time: 0.0697 seconds
- Result: 180469 | Solution: z=251, y=719 | Time: 0.0174 seconds
- Result: 16802263 | Solution: z=5717, y=2939 | Time: 0.3552 seconds
- Result: 1094290019 | Solution: z=16223, y=67453 | Time: 1.8126 seconds
- Result: 154182384011 | Solution: z=624683, y=246817 | Time: 29.8635 seconds
- Result: 31265007014327 | Solution: z=6680273, y=4680199 | Time: 309.6902 seconds
- Result: 4097709004901117 | Solution: z=90449591, y=45303787| Time: 1914.7875 seconds
- Result: 300511951868794537 | Solution: z=515987113z, y=582402049 | Time: 61236.8730 seconds

As the number of digits increases, the total time becomes impractically long for large semiprime numbers. However, this approach may still be useful in fields where very large numbers are not required.