Primality Tester Report

Emily Elzinga

September 5 2022

Figures 1 and 2 show the GUI application.

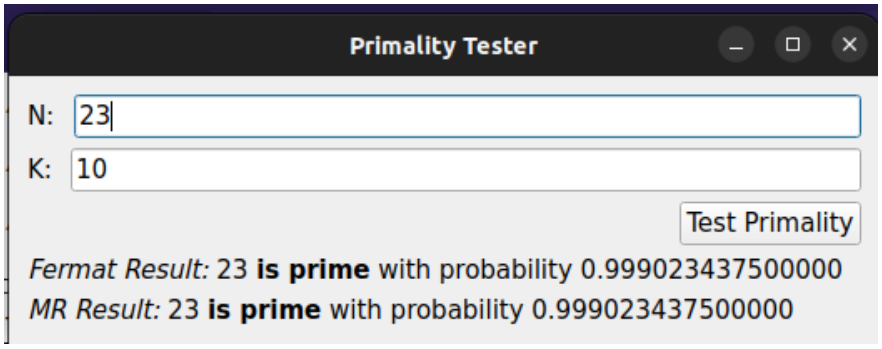

Figure 1 – Primality tester with 21 as an input

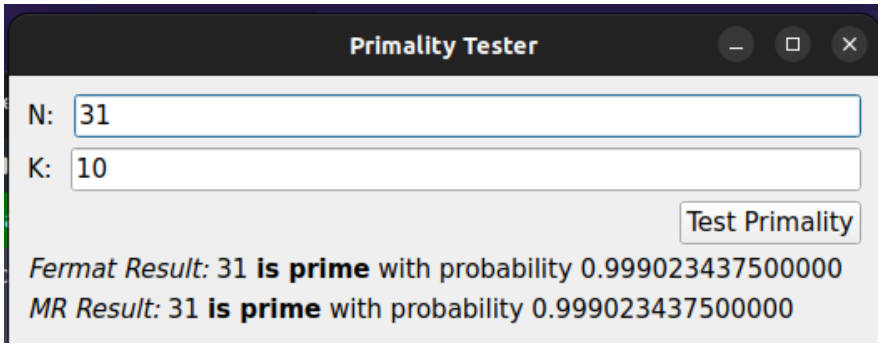

Figure 2 – Primality tester with 31 as an input

Your code must include the following:

- (20 points) A correct implementation of modular exponentiation. **Check**
- (20 points) A correct implementation of the Fermat primality tester. **Mostly check, although 561 comes out as not prime for some base values.**
- (10 points) A correct implementation of the Miller-Rabin algorithm to eliminate Carmichael numbers. **Check**

- (10 points) A brief discussion of some experimentation you did to indentify inputs for which the two algorithms disagree and why they do so. **For some composite numbers and for low K values, Fermat returns as prime. These cases only occur when the randomly generated base for each test is relatively prime with N. These special cases of N are called Carmichael numbers, and they can fool the Fermat test if K is relatively low. Figure 3 shows a disagreement between Miller Rabin and Fermat. 1105 is indeed composite, and is one of the Carmichael numbers.**

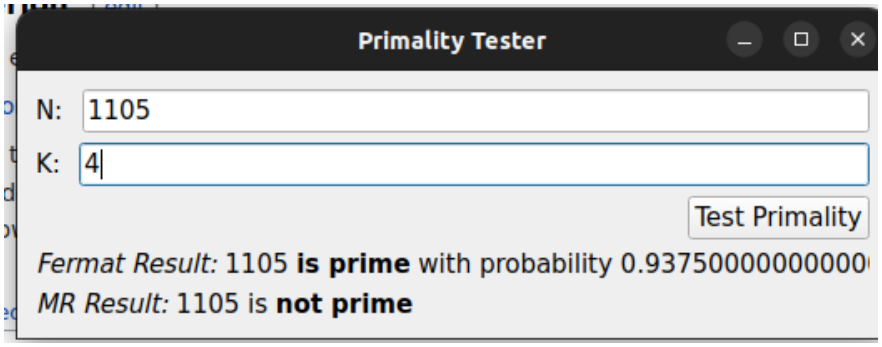

Figure 3 – Primality tester with 1105 as an input

[10 points] The time and space complexity of your algorithm by showing and summing up the complexity of each subsection of your code. **The time complexity for my implementation was** $O(n^5)$.

[10 points] Discuss the two equations you used to compute the probabilities p of correctness for the two algorithms (Fermat and Miller-Rabin). **I was not aware that they were different to eachother. To my understanding, both Fermat and Miller-Rabin return true or false. For a given number of trials k, the probability will be** $1 - \frac{1}{2^k}$**.**