

Alex Diviney

Dr. Paul Bodily

CS 4412

9/5/22

Project One Report

1. Working Examples:



Primality Tester

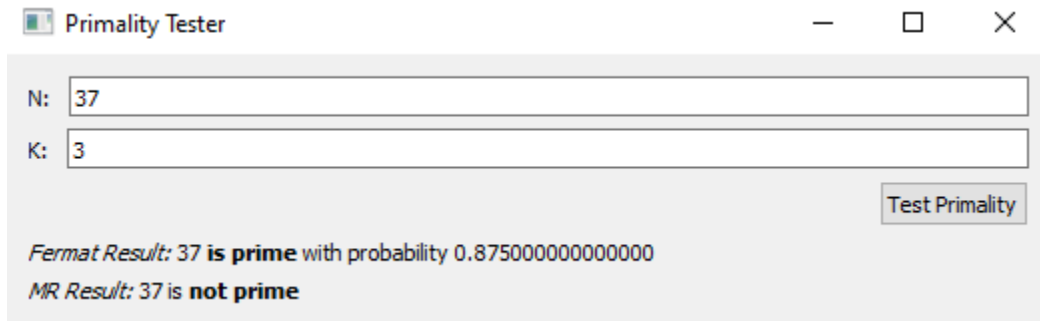
N: 291

K: 4

Test Primality

*Fermat Result: 291 is **not prime***

*MR Result: 291 is **not prime***



Primality Tester

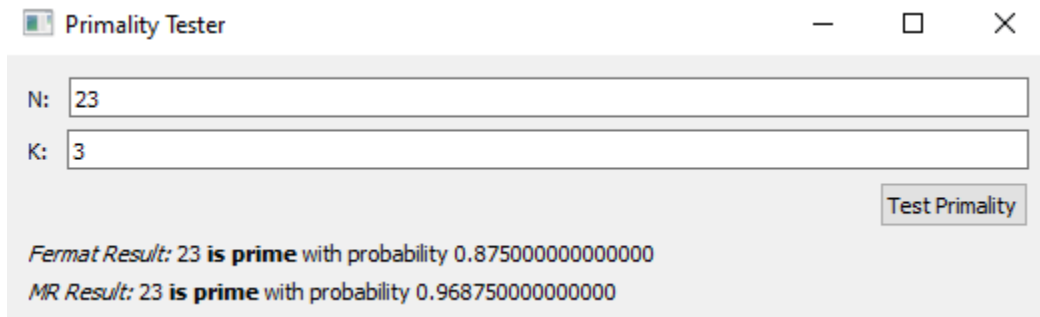
N: 37

K: 3

Test Primality

*Fermat Result: 37 **is prime** with probability 0.8750000000000000*

*MR Result: 37 is **not prime***



Primality Tester

N: 23

K: 3

Test Primality

*Fermat Result: 23 **is prime** with probability 0.8750000000000000*

*MR Result: 23 **is prime** with probability 0.9687500000000000*



Primality Tester



N: 23

K: 7

Test Primality

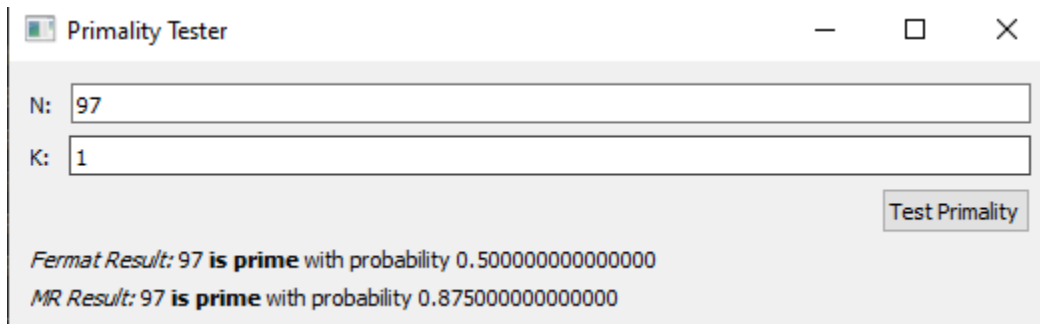
Fermat Result: 23 is prime with probability 0.992187500000000

MR Result: 23 is prime with probability 0.998046875000000

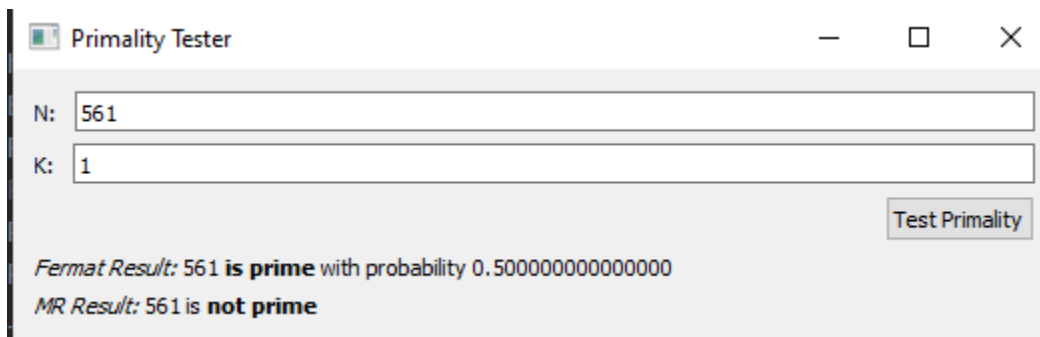
Discussion experimentation.

There were several bugs that I almost didn't catch. The Miller-Rabin method was confusing and I did not read some edge case text properly at first. I had the number 293 causing problems for a while, but now it should all be working properly.

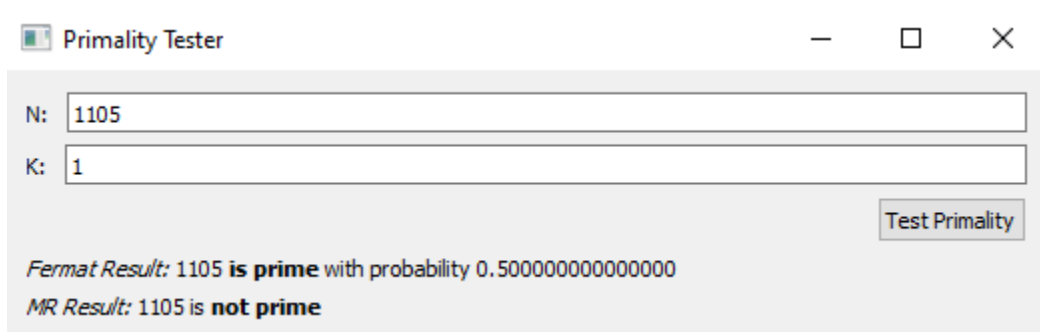
The following images demonstrate requirements passing. I chose 1105 for my unique Carmichael number.



A screenshot of a software window titled "Primality Tester". It contains two input fields: "N:" with the value "97" and "K:" with the value "1". To the right of these fields is a button labeled "Test Primality". Below the input fields, the text displays the results of the primality tests: "Fermat Result: 97 **is prime** with probability 0.5000000000000000" and "MR Result: 97 **is prime** with probability 0.8750000000000000".



A screenshot of a software window titled "Primality Tester". It contains two input fields: "N:" with the value "561" and "K:" with the value "1". To the right of these fields is a button labeled "Test Primality". Below the input fields, the text displays the results of the primality tests: "Fermat Result: 561 **is prime** with probability 0.5000000000000000" and "MR Result: 561 **is not prime**".



A screenshot of a software window titled "Primality Tester". It contains two input fields: "N:" with the value "1105" and "K:" with the value "1". To the right of these fields is a button labeled "Test Primality". Below the input fields, the text displays the results of the primality tests: "Fermat Result: 1105 **is prime** with probability 0.5000000000000000" and "MR Result: 1105 **is not prime**".

Complexity:

The complexity and space of the probability calculations was $O(1)$ for both functions.

The complexity of the modular exponentiation function was $O(\log(N))$, and the space requirements was $O(n)$ where n was the number of operations that the function completed. Normally space requirements for this equation can be $O(1)$ but I decided to store intermediate values instead of run cleanup operations that removed nodes from the end of my list.

Note that I decided to use an iterative approach instead of a recursive approach because I don't like recursion and thought it would be interesting.

The time complexity of the Fermat function was $O(k) * \text{the complexity of the modular exponential function}$, where k was the number of trials. The space complexity was $O(k)$ as I stored an array of random numbers instead of generating as needed.

The complexity of the Miller-Rabin function was $O(k) * O(\log(N))$ where k was the number of trials and $O(\log(N))$ represents that there was a repeated square root operation happening where N was the size of the prime input, ie. The bigger the input the proportionally more time the function took.

Probability equations:

I documented both of my probability equations. In short, the Miller-Rabin method captured $\frac{3}{4}$ of the numbers that Fermat wouldn't. In short, the Miller-Rabin method was 4x more likely to solve a given number, and so its probability of failure was 25% of what the Fermat's formula was.

The base equation for each was $(1/2^k)$ where k is the number of tests.