Miller Rabin Algorithm using Python

Hrishikesh V.
Department of Computer Science
PES University
Bangalore
PES1201700276

***Abstract*—Miller Rabin algorithm is a primality test that determines if a given number is a prime.**

***Index Terms*—Miller-Rabin, Primality test**

I. Implementation

Miller Rabin Primality Test algorithm is used to perform two sets of analyses on a given set of numbers.

The first test checks the accuracy of the algorithm for ”n” integers between 1 and 5 million against different values of k.

The second test determines how the running time of the algorithm is affected by K and the bit size of the input integer.

II. Observations

A. Accuracy test

The input dataset for this test comprised all composite numbers between 1 and 5 million. The reason for picking only composite numbers is that since this algorithm considers primes as probably being prime while composites to be definitely not primes, it is easier to check the accuracy with respect to composites.

The two graphs - one corresponding to Accuracy and the other, the number of incorrectly predicted values have been provided below. As you can see, the accuracy improves with k and consistently stays at 1.0 for values greater than 3.

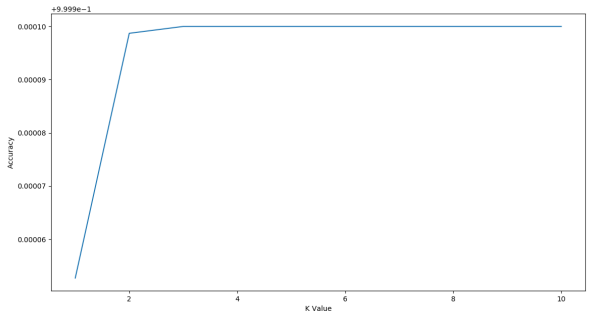

Fig. 1. Accuracy vs. K

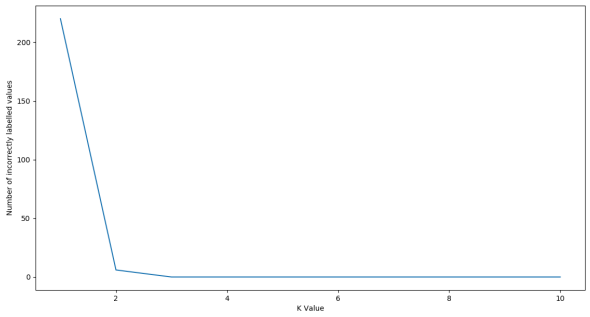

Fig. 2. Number of incorrectly predicted values

B. Time Complexity

The time complexity of the algorithm was first tested by varying the length of the integer passed to the function while keeping k constant at 5. As expected, the time taken by the primality test algorithm increases with the length of the input.

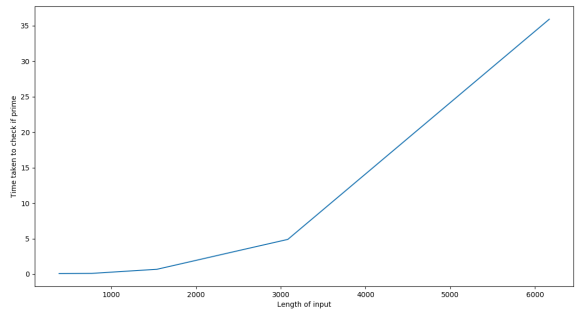

Fig. 3. Constant K with varying input size

When k was varied while keeping the integer length fixed, the results, however were inconclusive. Time taken to run the algorithm did not appear to be correlated with k.

III. Conclusion

By increasing k, the accuracy of the algorithm improved and reached 1.0 almost immediately. Increasing the length of the input resulted in an increase in time taken to execute the algorithm, which was as expected. It is perplexing however to note that for a given integer, changing k has no visible impact on time complexity.

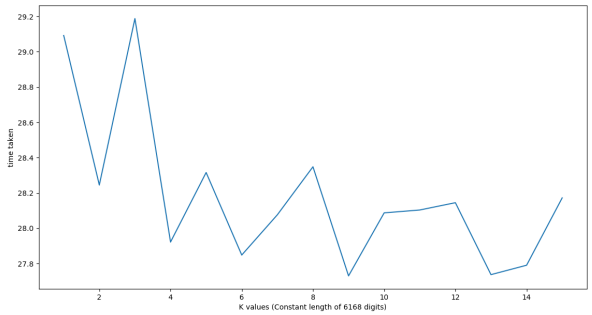

Fig. 4. k vs. Time (in seconds)

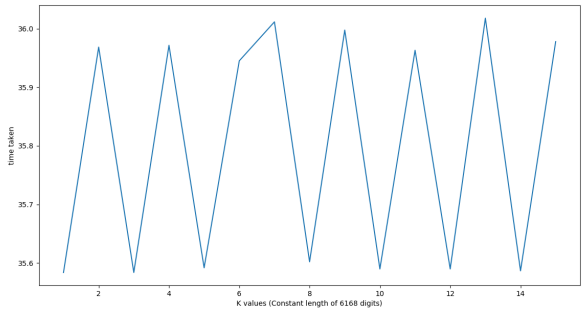

Fig. 5. K vs. Time (in seconds)