IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 6, June 2015.
www.ijiset.com
ISSN 2348 – 7968

Refinement of Prime Generating Algorithms

Neeraj Anant Pande[1]

[1] Department of Mathematics & Statistics, Yeshwant Mahavidyalaya (College), Nanded-431602
Maharashtra, INDIA

Abstract
Primes are the most mysterious figures amongst the positive integers. The fact that their sequence hitherto lacks a known simplistic formula generating successive primes poses a challenge and demands the better methods to generate them. As they are known to be infinite in number from almost antiquity, the search for the most efficient algorithm to generate primes has always been alive. Various algorithms have been devised to generate them which compete in the race of superiority. Here six different algorithms are presented which implement the progressively evolving methods based upon few very fundamental properties of integers. The approach has basically relied on using gradually efficient ranges for search for possible positive divisors of number, ignoring composite divisors for divisibility tests and finally using known non-primality of all even integers greater than 2. It is remarkable that application of every single property produces quite a refined version of earlier prime generating sieve.
Keywords: *Algorithm, Prime Number, Sieve.*

1. Introduction

The very formal beginning of the subject of mathematics is with arithmetic. In arithmetic, primarily positive integers are studied. With respect to the fundamental operations for these positive integers, 0 and 1 play key role for addition as well as subtraction. Amongst them 1 is considered as building block as successive addition (or subtraction) of 1 can construct all integers. When it comes to the advanced operation of multiplication, prime numbers take over the charge as building blocks.

As far as the definition of a prime number is concerned, it is so naïve stating that an integer $p > 1$ is said to be prime if, and only if, the only perfect divisors of p are ± 1 and $\pm p$.

As stated right in the beginning, since we are essentially interested in positive integers, we look at a prime as a positive integer greater than 1 having 1 and itself as the only positive divisors, which are just the trivial or improper divisors.

What makes the primes the building blocks for all positive integers except 1 is the property well known in mathematical community as the Fundamental Theorem of Arithmetic which goes like :

Every integer $n > 1$ can be expressed uniquely as product of powers of distinct primes.

2. Peculiar Properties of Primes

These prime numbers exhibit many properties some of which are characteristic of their very own. A noteworthy few of them are :

- 2 is the very first and the only even prime.
- Primes are infinite in number.
- They are highly irregularly spaced.
- There are very small gaps (of 2) in almost infinitely many consecutive primes.
- There are arbitrarily large gaps between successive primes.
- They lack a systematic and simplistic formula yielding all of them in succession.

The third and the last of these properties have always posed a challenge of devising better and better prime generating sieves. A sieve is a process to generate all primes in the given range 2 to n for any $n > 1$.

3. Various Approaches for Prime Generation

Beginning with very naïve one, systematically various refined algorithms have been developed for generating successive primes. The refinement approaches of these evolving algorithms fall under three major categories, viz., those in which the range of search for possible divisors is reduced, those in which the selective divisors are preferred, those in which the range of the numbers themselves is halved. Of course, a just combination of all of these leads to magnificent rise in the efficiency in terms of requirements of both time and number of checks.

ISSN 2348 – 7968

4. Algorithms by Reducing Divisor Ranges

The most simple and equally inefficient method for determination of primes in a given range up to certain $n > 1$ is of looking for all possible numbers between 2 to $k - 1$ as perfect divisors for every $k \leq n$. On finding even one perfect divisor in this range, the concerned k is not prime, or else it is prime. The first Algorithm 1.1.1 goes this way :

```
Take an integer n larger than 1
For all values of k from 2 to n
   For values of integer d from 2 to k–1
      If d divides k perfectly
         Stop checking, k is not prime
      Else
         Continue checking with next d
   If checks don't stop, k is prime
Take next value of k
```

Here, for every k which is prime, all $k - 2$ testings (from 2 to $k - 1$) are needed to be done. While if some k is not prime testings continue only till first perfect positive divisor is encountered.

A property of positive integers immediately suggests an improvement in this approach. Any positive integer k cannot have an integral divisor greater than half of it, i.e., $k/2$. So the range of search for positive divisors reduces immediately to half with the following version of Algorithm 1.1.2 :

```
Take an integer n larger than 1
For all values of k from 2 to n
   For values of integer d from 2 to[k / 2]
      If d divides k perfectly
         Stop checking, k is not prime
      Else
         Continue checking with next d
   If checks don't stop, k is prime
Take next value of k
```

For any real number x, the symbol $[x]$ represents the greatest integer less than or equal to x. This approach doesn't reduce the number of steps required for those numbers k which are not prime. Because, as in the previous attempt, here also, for this composite k, the first divisor is going to be detected in the same number of steps at same value. But when it comes to those k which are prime, instead of all those $k - 2$ testings (from 2 to $k - 1$) only $\left[\frac{k-2}{2}\right]$ testings (from 2 to $\left[\frac{k}{2}\right]$) will be done.

Further improvement is achievable by employing one more interesting property of positive integers k that their divisors always occur in pairs, for every divisor that is less than $\sqrt{k}$, there a divisor which is greater than $\sqrt{k}$ and vice versa. This eliminates the necessity to test for divisors greater than $\sqrt{k}$ for primality and a better Algorithm enumerated 1.1.3 arises :

```
Take an integer n larger than 1
For all values of k from 2 to n
   For values of integer d from 2 to [√k]
      If d divides k perfectly
         Stop checking, k is not prime
      Else
         Continue checking with next d
   If checks don't stop, k is prime
Take next value of k
```

Now even for prime k, instead of $\left[\frac{k-2}{2}\right]$ checks much less $\left[\sqrt{k}\right]-1$ checks are enough to ascertain primality. Of course, as in the previous case, this approach doesn't provide any refinement in case of composite k, but does do a significant calculation reduction for prime k; to be very precise, $\left[\frac{k}{2}\right]-\left[\sqrt{k}\right]$ of them.

Runtime comparison of above three algorithms has been exhaustively done in [3] with comparative point of view. Here we present the number of steps required by each of these algorithms for determining all primes in initial ranges of powers of 10.

Table 1: Algorithms Reducing Divisor Ranges

Sr. No.	Range	Steps Required		
		Algorithm 1.1.1	***Algorithm 1.1.2***	***Algorithm 1.1.3***
1.	1-10	15	9	8
2.	1-100	1133	616	236
3.	1-1000	78022	40043	5288
4.	1-10000	5775223	2907640	117527
5.	1-100000	455189150	227995678	2745694

The comparative analysis of performance is depicted in the following figure in terms of the percentage reduction in the number of steps required compared to Algorithm 1.1.1.

The efficiency achieved over Algorithm 1.1.1 by the last one in this series, viz., Algorithm 1.1.3 is enormous, it is seen to require less than 1% of the efforts compared to the prior one for sufficiently higher ranges.

ISSN 2348 – 7968

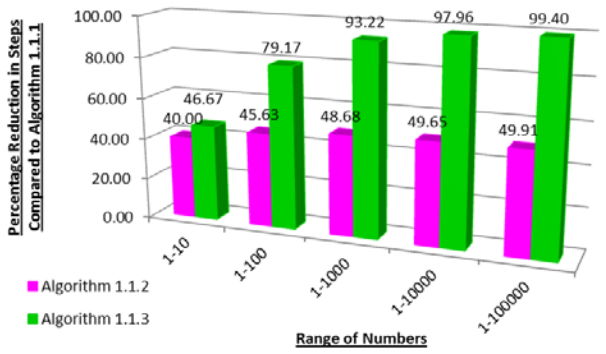

Fig. 1 Percentage Step Reduction with Respect to Algorithm 1.1.1

5. Algorithms by Using Selective Divisors

The improvement obtained in the previous approach was based on reducing the number of divisors used for testing primality. But in the process the different complete ranges of numbers were under consideration, viz., from 2 to $k-1$ or 2 to $[k/2]$ or 2 to $\left\lceil \sqrt{k} \right\rceil$. Another property of divisibility can now be used which states that an integer k which is not divisible by 2 cannot be divisible by any other even integer. This immediately suggests that for any k, once divisibility by 2 is tested and found to have negative outcome, testing divisibility by all higher even integers like 4, 6, 8, … will surely give negative results and consequently is unnecessarily. So, we can take just selective divisors for testing primality, viz., 2 to $\left\lceil \sqrt{k} \right\rceil$ but only odd ones after 2. This leads to Algorithm 2.1.3 proposed in [4]:

```
Take an integer n larger than 1
2 is a prime
For all values of k from 3 to n
    For odd, except 2, d from 2 to ⌈√k⌉
        If d divides k perfectly
            Stop checking, k is not prime
        Else
            Continue checking with next d
    If checks don't stop, k is prime
Take next value of k
```

For prime k, instead of $\left\lceil \sqrt{k} \right\rceil - 1$ checks of previous best Algorithm 1.1.3, only $\left\lceil \frac{\sqrt{k}}{2} \right\rceil + 1$ checks are enough to ascertain primality. And here unlike previous cases, in case of composite k also, there is a little reduction in calculations as even numbers are skipped for divisibility tests for them either.

It was the genius of Eratosthenes centuries ago who could visualize this scope of improvement further and gave an algorithm known today by his name as Sieve of Eratosthenes which says that much in the same way like we can neglect even numbers after 2 for divisibility in primality tests which merely are multiples of 2, we can also neglect all the multiples of all primes except themselves in the primality tests leading to Algorithm 3.1.3 of [5]:

```
Take an integer n larger than 1
2 is a prime
For all values of k from 3 to n
    For prime d from 2 to ⌈√k⌉
        If d divides k perfectly
            Stop checking, k is not prime
        Else
            Continue checking with next d
    If checks don't stop, k is prime
Take next value of k
```

For every positive integer d, there is an arithmetical function denoted by $\pi(d)$ which counts the number of primes less than or equal to d. Making use of this function, we see here that for prime k, only $\pi\left(\sqrt{k}\right)$ checks are enough to ascertain primality which are very much less than earlier number $\left\lceil \frac{\sqrt{k}}{2} \right\rceil + 1$. Of course, even for composite k, this procedure has less number of checks required depending upon actual value of k.

The extents of improvements can be visualized through an exhaustive comparison of number of steps required by two algorithms of this section and the best of the previous section.

Table 2: Improvement of Algorithm by Use of Selective Divisors

Sr. No.	Range	Steps Required		
		Algorithm 1.1.3	***Algorithm 2.1.3***	***Algorithm 3.1.3***
1.	1-10	8	9	8
2.	1-100	236	185	181
3.	1-1000	5288	3349	2801
4.	1-10000	117527	65956	43753
5.	1-100000	2745694	1445440	744436

The comparative analysis of performance is depicted in the following figure in terms of the percentage reduction in the number of steps required compared to Algorithm 1.1.3.

The efficiency achieved over Algorithm 1.1.3 by both of those in this section is seen to be crossing 50% very soon.

ISSN 2348 – 7968

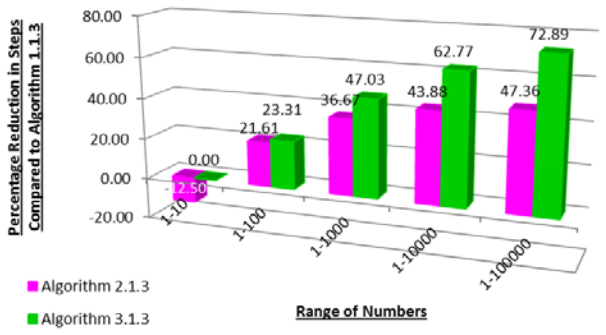

Fig. 2 Percentage Step Reduction with Respect to Algorithm 1.1.3

6. Algorithms by Using Selective Numbers

The improvement obtained in the previous approaches was based on reducing the number of divisors used for testing primality by decreasing the range as well as selective choice of divisors. One more simple and peculiar property of primes (listed as very first in section 2) helps further refine algorithms for prime search.

The known fact that 2 is the only even prime and all numbers after 2 are non-primes removes the necessity of testing them for primality. Applying this selective technique for numbers themselves, the bettered version of classical sieve of Eratosthenes takes form of Algorithm 3.2.3 of [6] as follows :

```
Take an integer n larger than 1
2 is a prime
For all odd values of k from 3 to n
    For prime d from 2 to ⌈√k⌉
        If d divides k perfectly
            Stop checking, k is not prime
        Else
            Continue checking with next d
    If checks don't stop, k is prime
Take next value of k
```

Half of the numbers are omitted from primality tests and number of checks reduces equally.

Table 3: Superior Algorithms of Three Approaches

Sr. No.	Range	Steps Required		
		Algorithm 1.1.3	*Algorithm 3.1.3*	*Algorithm 3.2.3*
1.	1-10	8	8	1
2.	1-100	236	181	84
3.	1-1000	5288	2801	1804
4.	1-10000	117527	43753	33756
5.	1-100000	2745694	744436	644439

We conclude by plotting the graphs of number of step requirements by superior algorithms of each section.

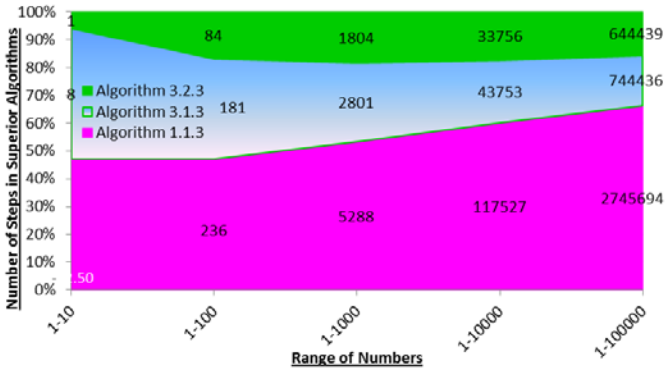

Fig. 3 Step Requirements of Superior Algorithms of Three Approaches

It is no surprise that last Algorithm 3.2.3 surpasses all others as it combines the techniques of all them in addition to its own one.

Acknowledgments

The author has used in implementing the algorithms in the work their products and is thankful to the Java (7 Update 25) Programming Language Development Team and the NetBeans IDE 7.3.1 Development Team.

The author expresses gratitude to the University Grants Commission (U.G.C.), New Delhi of the Government of India for funding this research work under a Research Project (F.No. 47-748/13(WRO)).

References

[1] Donald E. Knuth, "The Art of Computer Programming, Volume 1: Fundamental Algorithms", Addison-Wesley, Reading, MA, 1968.

[2] Evan Niven, Herbert S. Zuckerman, Huge L. Montgomery, "An Introduction to the Theory of Numbers", John Wiley & Sons Inc., U.K., 2008.

[3] N.A.Pande, "Evolution of Algorithms: A Case Study of Three Prime Generating Sieves", Journal of Science and Arts, Romania, No.3(24), pp. 267 – 276, Year 13, 2013.

[4] N.A.Pande, "Algorithms of Three Prime Generating Sieves Improvised by Skipping Even Divisors (Except 2)", American International Journal of Research in Formal, Applied and Natural Sciences, Issue 4, Volume 1, pp. 22 – 27, 2013.

[5] N.A.Pande, "Prime Generating Algorithms by Skipping Composite Divisors", International Journal of Computer Science & Engineering Technology, Vol. 5, No. 09, pp. 935 – 940, 2014.

[6] N.A.Pande, "Improved Prime Generating Algorithms by Skipping Composite Divisors and Even Numbers (Other than 2)", Communicated, Journal of Science and Arts, Romania, 2015.

[7] Herbert Schildt, "Java : The Complete Reference", 7th Edition, Tata McGraw - Hill Education, 2006.