Recursive Prime Numbers

Sabin Tabirca* Kieran Reynolds*
*Computer Science Department, University College Cork, Ireland
email: {s.tabirca, k.reynolds}@cs.ucc.ie,

Many researchers study prime numbers for the curiosities that they possess rather than the position they occupy at the foundations of Number Theory. This study may be in any numbers of areas from applications to multimedia to searching for special or unusual primes. It is truly awe inspiring to see how much time can be expended on prime numbers without a realistic application.

In this article, a sequence of prime numbers, called Recursive Prime Numbers, is identified before a complete search is undertaken to verify that the sequence is finite by finding all existing prime numbers of the specified form. This could be done with considerable effort by hand, but here a simple computer program has been used to speed the calculations. So now the question must be answered; what are recursive prime numbers. The easiest way to answer that is to say that a prime number is recursively prime if the number is prime can be constructed by adding a digit to an already recursive prime number.

1. Recursive Prime Numbers

The idea of Recursive Prime Numbers arose when asked if it were possible to create infinite sequences of prime numbers by adding digits to the end of an existing prime.

Definition 1. *A number is said to be a recursive prime number if it and all of the initial segments of the decimal expansion are prime. We can recursively define those numbers as follows:*

a) 2, 3, 5, 7 are recursive prime numbers

b) if $\overline{a_0a_1...a_n}$ *is a recursive prime number and* $\overline{a_0a_1..a_na_{n+1}}$ *is a prime number then* $\overline{a_0a_1..a_na_{n+1}}$ *is a recursive prime number as well.*

Example 1. 23333 is a recursive prime number since 2, 23, 2333 and 23333 are all prime.

Although, with only a little examination it becomes clear that it is very unlikely that such an infinite sequence could be found, still the concept is one that is quite interesting and demanded some attention. It is not a difficult task to systematically find all prime numbers of this form. Let us consider the following sets of prime numbers

$$L^1(a) = \{a\}, \forall\ a \in \{2,3,5,7\} \tag{1}$$

$$L^{n+1}(a) = \left\{10 \cdot x + y : x \in L^n(a),\ y \in \{1,3,7,9\},\ 10 \cdot x + y \text{ is prime}\right\} \forall\ n \geq 1 \tag{2}$$

$$L(a) = \bigcup_{n \geq 1} L^n(a), \forall\, a \in \{2,3,5,7\}, \tag{3}$$

where $L^n(a)$ represents the set of the recursive prime numbers which start with the digit a and have n digits.

We used Java computation to generate the set $L(a)$ of all the recursive prime numbers that start from the digit a (see Figure 1). The program chooses to use Java's long type as opposed to the reference type, BigInteger. This choice was made to simplify the code in the expectation that there would be no need for the increased size provided for BigInteger. Similarly, a simple trial division primality test has been used in lieu of a more efficient test since the numbers are expected to remain relatively small in all cases (see Figure 1). There are many references available for those interested in primality testing, [Knuth, ***], [Shallit, 1996], [Romero, 1998].

The code creates two queues queueOld and queueNew for the sets $L^n(a)$ and $L^{n+1}(a)$ respectively. Initially, the queue queueOld contains the digit a. The loop for simulates Equation (2) by generating the elements of queueNew from the elements of queueOld and the set of last digits. An element prime of queueOld is removed from the queue, which is concatenated with the last digits {1, 3, 7, 9}. If a prime number is obtained, we insert it in the queue queueNew. When all those numbers are composite, we find that queueNew is empty, therefore the computation finishes.

```
public Vector listRecurs(long a){
      LinkedList queueOld = new LinkedList();
      LinkedList queueNew = new LinkedList();
      Vector primes = new Vector();

      queueOld.addLast(new Long(a));    // digit is added to queue
      long [] lastDigit = {1, 3, 7, 9};

      for(int n=1;! queueOld.isEmpty();n++) {

        // generate queueNew for the set L^{n+1}(a)

        while (! queueOld.isEmpty()){
          // get an element prime from queueOld
          long prime = ((Long)queueOld.removeFirst()).longValue();
          primes.addElement(new Long(temp));

          // generate all the recursive prime numbers from prime
          for (int i = 0; i < lastDigit.length(); i++) {
            long primeNext = prime * 10 + lastDigi[i];
            if(this.testPrimality(primeNext))
                        queueNew.addLast(primeNext);
          }

        }
        while (! queueNew.isEmpty()){
          long nr = ((Long)queueNew.removeFirst()).longValue();
          queueOld.addLast(nr);
        }

      }

      return primes;
}
```

```
public Boolean testPrimality(long num){
   if (this.getLong()==2 || this.getLong()==3)return isPrime;
   if (this.getLong()%2==0 || this.getLong()%3==0)return isComposite;

   for (int i=5; i<(long) ath.floor(Math.sqrt(this.getLong())); i+=4)
      {  if (this.getLong() % i == 0) return isComposite;
         i+= 2;
         if (this.getLong() % i == 0) return isComposite;
      }

      return isPrime;
}
```

Figure 1. Java program to list all recursive prime numbers.

The following theorem establishes the correctness of our computation.

Theorem 1. The contents of queueOld before the n-th iteration of the loop for is $L^n(a)$, therefore the contents of the vector primes is $L(a)$.

Proof. Induction is used for this proof.

Since the queue queueOld initially contains only a, we find that the property holds for n=1. Suppose that before the n-th iteration the contents of queueOld is $L^n(a)$. In the loop for we generate queueNew as follows:

- for any element prime of queueOld= $L^n(a)$ and for any lastDigit[i]
- if prime*10+lastDigit[i] is prime then add it to queueNew

Therefore, the contents of queueNew will be identical as $L^{n+1}(a)$. At the end of this iteration the elements of queueNew are transferred to queueOld therefore before the iteration (n+1)-th the contents of queueOld is $L^{n+1}(a)$.

♦

The computation relieved that the sets $L^n(a)$ are empty for values n>8. Therefore, each digit 2, 3, 5, 7 generates only a finite number of recursive primes. This is detailed in the next section.

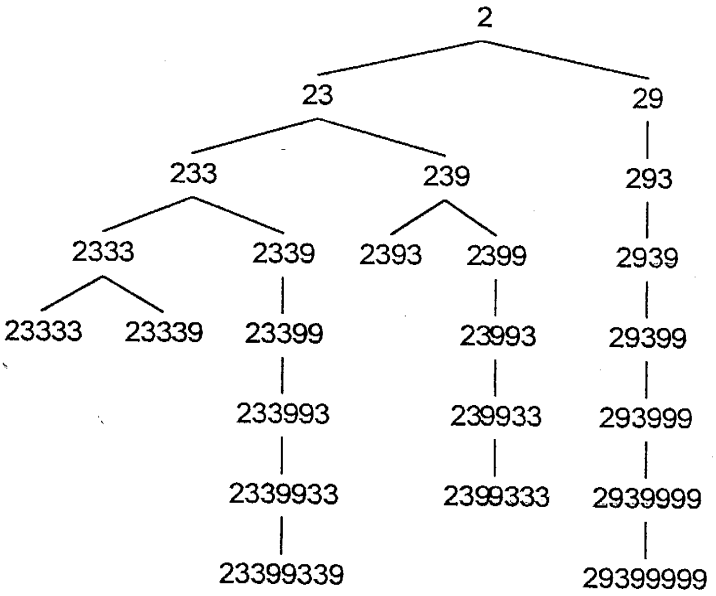

Figure 2. Recursively constructed prime numbers with starting digit 2.

2. Series of Recursive Primes

In order to gain a better understanding of recursive prime numbers it is necessary to view the results for each starting digit separately, beginning with 2. With 2 as a starting digit there are only two possible extensions for each number; 3 or 9. This is due to the fact that concatenating a 1 or 7 at any stage causes the number to become divisible by 3. The results can be visualised as a tree as in Figure 2.

Since at any stage there are only two possible digits to add, this case results in a binary tree. It is interesting to note that the right child of each of the nodes 29, 239 and 233 result in long "slender" branches. It is these branches that result in the longest sequences for this case, two of which are eight digits long. In total there are 24 primes in this tree. As it will be seen later this is the joint largest tree in terms of nodes, and shares the same longest sequences with each of the other trees. It is interesting that this case, despite its limitation of potential digits, is not limited in size at all. In fact, this is the only tree that has two sequences of length 8.

Unlike the previous case, when the number begins with 3 there are 4 potential digits to concatenate to the number at some stages. There are still some limitations. For example 3 and 9 result in composite numbers at the first stage but otherwise are options at subsequent step. Meanwhile, at the first concatenation 1 and 7 result in new primes but following that any sequence can only have one more of these numbers before they become divisible by 3. This tree is not a binary tree, but it very nearly is. Only one node, 31, has three children. As can be seen in the following image.

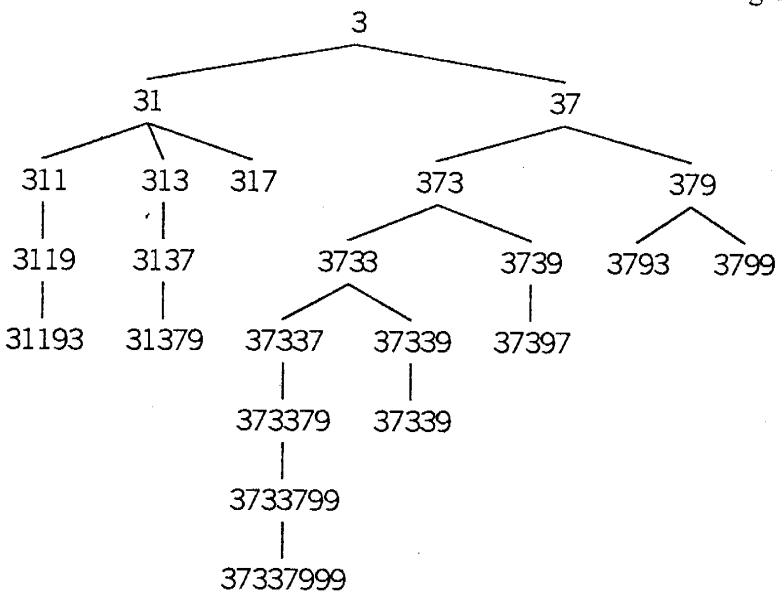

Figure 3. Recursively constructed prime numbers with starting digit 3.

This case results in 23 prime numbers, one less than the previous case and also shares the eight digit longest sequence. One element that is found in this sequence that is absent from the previous example is twin primes. In fact, there are two pairs of primes found in this tree; (311, 313) and (37337, 37339).

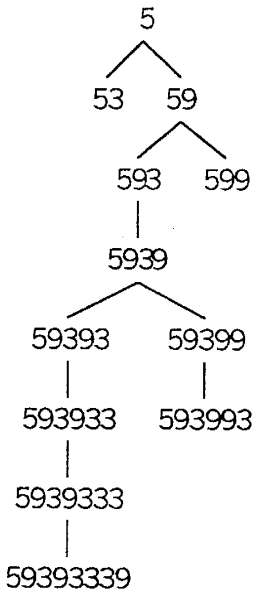

Figure 4. Recursively constructed prime numbers with starting digit 5.

The case with 5 as the first digit produces the most unusual trees. While the other cases result in reasonably broad trees, this case results in a slender tree. Also, the other cases result in 23 or 24 primes, but this case results in just 12 primes. As well as this, it shares the limitation of potential digits with the first case examined, again allowing only 3 and 9 to be concatenated to the number at each stage. Yet surprisingly, the longest sequence found is 8, equal to that of all the previous cases.

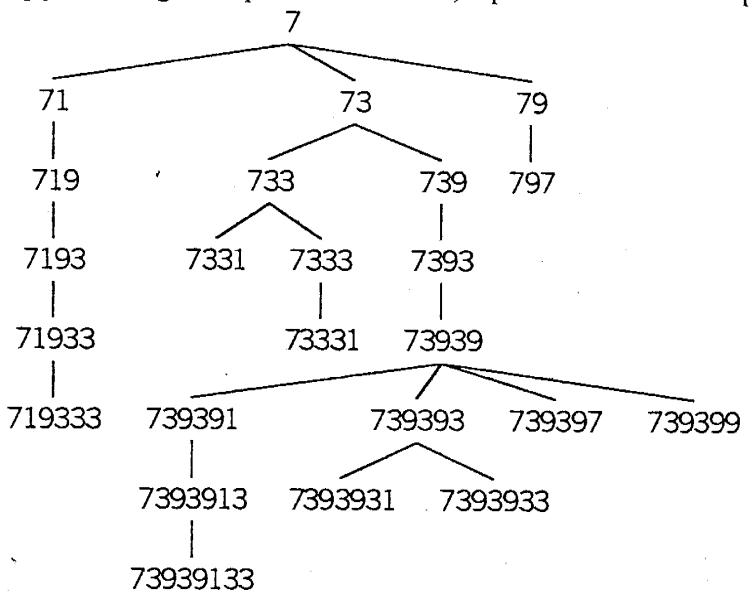

Figure 5. Recursively constructed prime numbers with starting digit 7.

The final case to be examined is the tree rooted at 7. Again there are limitations on the use of 1 and 7 for these numbers. In this case, since the numbers begin with a 7, every sequence can contain just one of either 1 or 7 at any subsequent stage. As with each of the trees seen previously, this also shows some interesting characteristics.

Firstly, it is not a binary tree with two nodes having too many children. The node 7 has three children while interestingly 73939 has four children, one for each possible

digit. This is a unique occurrence in this search. This tree has 24 primes, making it as large as the first example and also has a longest sequence of 8 digits. However, possibly the most interesting feature of this tree is that it contains five pairs of twin primes; (71, 73), (7331, 7333), (739391, 739393), (739397, 739399), and (7393931, 7393933).

Conclusions

This article has proposed a new class of prime numbers called "recursive primes". Using Java computation all the recursive prime numbers have been generated. It has been identified only 83 numbers that are recursive primes. Among them 5 pairs of twin primes have been found.

References.

1. Romero, C. and Kumanduri, R.: Number Theory with Computer Applications, 1998, Prentice Hall, USA.
2. Shallit, J. and Bach, E.: Algorithmic Number Theory: Volume 1 – Efficient Algorithms, 1996, MIT Press, England.