Generating Large Prime Numbers Using the Perrin Sequence

Dan Tamir

2007-11-05
Generating Large Prime numbers Using the Perrin Sequence

Dan Tamir
Texas State University

Introduction:

The Perrin sequence is defined by the recursion [2]:

\[ A_n = A_{n-2} + A_{n-3}; \quad \{A_1, A_2, A_3\} = \{0, 2, 3\} \]  

Perrin has shown that “if \( p \) is prime, then \( p \mid A_p \)” (meaning \( p \) divides \( A_p \)). The converse, i.e., “if \( p \mid A_p \), then \( p \) is prime”, was “believed” to be true for decades, until Adams and Shanks (1982) have shown that the composite number \( 521 \times 521 = 271441 \) divides \( A_{271441} \) (hence, \( 271441 \mid A_{271441} \)) [1].

The Perrin test is defined to be the truth value of the proposition: ‘\( P \mid A_p \)’. A pseudo Perrin prime (pPp) is defined to be an integer \( q \) such that \( q \) passes the Perrin test. That is:

\[ q \mid A_q = 'True' \]  

The characteristic equation of the recursion \( A_n = A_{n-2} + A_{n-3} \) is

\[ (z^3 - z - 1) = 0 \]  

The real root of the equation \( r \sim 1.324717957 \) can be used to approximate \( A_n \) as [1]:

\[ A_n \sim r^n \sim (1.324717957)^n \]  

Finding large primes

The Perrin test is not sufficient. Nevertheless, it is a strong primality test [1]. Hence, it can be used as a “pre-processing procedure” for identifying large prime numbers.

Equations 3 and 4 can be used to implement the pPp test efficiently. For example, one can verify that \( 271441 \) is not a prime using multiple precision operations available in Mathematical packages such as Matlab and Mathematica, or using a multiple precision libraries such as GMP (GNU multiple precision library).

Tamir used Matlab to solve equation (3) with a precision of 1 million digits. Let \( r_{1m} \) be the solution obtained by Matlab. Next, he used equation (4) in the form:

\[ A_{271441} = \text{ceiling}((r_{1m})^{271441}) \]
He also used an iterative procedure to calculate $A_{271441}$ using equation (1) and verified equation (5). Finally, he has shown that, as expected, $271441 \mid A_{271441}$.

To generalize and further improve the proposed procedure for generating large primes numbers, consider $n$ where $n$ is a large number with relatively high likelihood to be prime (e.g., $n$ is a large Mersenne number or a Euclidean pseudo prime). In other words, $n$ is constructed with “prime likelihood” and potentially has passed several initial primality tests. Then, before running $n$ through factorization one can check if $n$ is a pPp.

**Critique**

There are two main potential problems in this approach:

1) **Precision** – depending on the precision of the estimate of $r$, the root of the characteristic equation (3), Equation (4) may provide a poor estimate for $A_n$. In this case taking the ceiling or the floor of $r^n$ may not yield the right value for $A_n$. Nevertheless, the “intuitive function” ‘neighborhood$(x)$’ which returns a set of integers around the real number ‘$x$’ can be used to enable identifying the actual value of $A_n$.

This can be accomplished solving the equation: ‘$i = j + k$’ where: $i \in \text{neighborhood}(x)$, $j \in \text{neighborhood}(x-2)$, and $k \in \text{neighborhood}(x-3)$.

2) **Time / space complexity** – the pPp uses $A_n$. For a large $n$, $A_n >> n$. Hence the pPp significantly increases the space requirements for verifying primality. Moreover, the fact that $A_n >> n$, also increases the time complexity of the algorithm.

**Further Research**

Further research to evaluate the severity of the critique problems raised above is due. It is currently being done by Tamir.

**References**
