

0. COPYING

Copyright 2009, Ivan Zaigralin.

This document is licensed under Attribution-Share Alike 3.0 license,
full text at <http://creativecommons.org/licenses/by-sa/3.0/>.

1. SUMMARY

This section is intended for the general audience.

The Prime Crawler (or PC) is a pseudo-random number generator. It generates an infinite string consisting of zeroes and ones. The internal state of the generator may be represented by what we call a *pad*, which may be depicted as follows:

$$(1) \quad \begin{array}{r} \underline{001} \\ \underline{01100} \\ \underline{1001100} \end{array}$$

The pad consists of several so-called *lines*, which are fixed finite binary sequences (later treated as infinite periodic sequences). Notice that one number in each line is underlined, indicating the current position in that line. To generate a pseudo-random digit, PC counts the number of ones at underlined positions. If the number of ones is even, it generates 0, and if the number of ones is odd, it generates 1. In the state depicted above, it will generate 1. Readers familiar with the basic programming will recognize this a **xoring** of the underlined digits. After generating a digit, PC shifts underlined positions to the right. If the end of a line is reached, it shifts the underlined position to the beginning of that line. Here are a few consecutive steps for the depicted pad:

	State	Output
(2)	$\underline{001}$	
	$\underline{01100}$	1
	$\underline{1001100}$	
	$\underline{001}$	
	$\underline{01100}$	1
	$\underline{1001100}$	
	$\underline{001}$	
	$\underline{01100}$	0
	$\underline{1001100}$	
	$\underline{001}$	
	$\underline{01100}$	1
	$\underline{1001100}$	

So the sequence generated by the depicted pad starts with 1101 and continues (potentially) forever. To exclude truly trivial cases, we demand that each line must contain at least one 0 and at least one 1. Additionally, we require that the line lengths are pairwise distinct, relatively prime numbers. In the example given, the lengths are 3, 5, 7. It comes out that just on these premises, the period of the generated sequence is the product of line lengths (in this specific case, the period is $3 \cdot 5 \cdot 7 = 105$), the period being the shortest repeating substring. For any PC, the sequence starts repeating itself as soon as the position markers find themselves in the initial state (1).

While using a pad (rather than a formula) in order to generate a pseudo-random sequence may seem inelegant, one has to recognize that even very small pads will result in very large periods. For example, for a pad on the first 100 primes, the period has 220 decimal places. It is easy to see that if n is the number of lines, the period grows faster than a^n , and from the Prime Number Theorem [1] one can see that it grows faster than $n!$.

A large period is, of course, irrelevant if the resulting sequence lacks “randomness” or “variability”. It would be very fortunate if we could show that for some integer k , all possible binary strings of length k are approximately equidistributed, i.e. each of the 2^k different strings occurs about the same number of times within one period of the pseudo-random sequence. It would be even better if we could show that k could be made arbitrarily large by increasing n , the number of lines in the pad, and that we can get to high enough values of k without slowing down the computation too much or running out of the computer memory.

Another nice property we want to secure is being unpredictable. Ideally, one should not be able to predict the next digit with accuracy above 0.5, no matter how many outputs in a row one observes.

As it stands, we can show that binary digits are approximately equidistributed, and we have a good intuition about other very short strings. With a slight modification to the algorithm, we should be able to say the same about strings of length comparable to that of the longest line in a pad.

2. NOTATION

$\mathbb{N} = \{0, 1, 2, \dots\}$.

Unless otherwise stated, *sequences* are indexed by \mathbb{N} .

A *string* is a finite sequence. A *substring* of a sequence is an improper subsequence which is also an interval with respect to the order topology. A *character* is synonymous with a *member of a sequence*.

3. PRIME CRAWLER

A *pad* consists of n lines L_i , each line a binary sequence with the period p_i , $i = 0, 1, 2, \dots, n-1$, with $p_i \neq 1$ pairwise distinct and relatively prime. Define $L_i(j)$ to be the j -th character in L_i , where $i = 0, 1, \dots, n-1$ and $j \in \mathbb{N} = \{0, 1, 2, \dots\}$. To generate a pseudo-random binary sequence $\langle r_j \rangle$, let

$$r_j = \bigoplus_{i=0}^{n-1} L_i(j),$$

where \oplus is **xor**, or the *exclusive or*, i.e. $a \oplus b = 0$ if $a = b$, $a \oplus b = 1$ otherwise.

3.1. Period. A constant line is not interesting, as a line of zeroes has no effect on r_j , a line of ones negates each r_j . We will assume that no line is constant (actually, it was already implicit in how we wanted periods to be different from 1). On that assumption, the period of $\langle r_j \rangle$, given some pad, is guaranteed to be

$$(3) \quad \prod_{i=0}^{n-1} p_i,$$

as will be shown shortly. In general, any such sequence $\langle r_j \rangle$ with period (3) will be called *the pad product of lines* L_i , $i = 0, 1, \dots, n-1$ and will be written simply as

a product $L_0L_1\dots L_{n-1}$. It should be clear that the pad product is commutative and associative, as is \oplus .

The period is clearly bounded from above by (3); does it have to be that large? Yes. Fix L_0, L_1 with periods p_0, p_1 respectively. It will suffice to show that the period of L_0L_1 is p_0p_1 for p_0 and p_1 relatively prime and the induction will take care of the rest. Consider the initial segment of L_0 which is a concatenation of p_0 strings, each of length p_1 . These strings cannot all be the same, as it would force L_0 to be constant. Hence the initial segment of L_0L_1 of length p_0p_1 cannot be a concatenation of p_0 identical strings. Similarly, the same initial segment cannot be a concatenation of p_1 identical strings. This establishes (3).

3.2. Distribution of Digits. We will say that the proportion of zeroes in a line is the proportion of zeroes in a periodic substring in that line.

Let a_0, a_1, \dots be proportions of zeroes in the respective lines. On our assumptions, $a_i \neq 0$ or 1. Also, $a_i = 1/2$ makes digits perfectly equidistributed in any pad product involving the respective line, so we will omit this case as trivial.

Let $b_i = a_i - 1/2$, so that $b_i \in (-1/2, 1/2)$, $b_i \neq 0$. Let c_i be the proportion of zeroes in the pad product of lines L_0, \dots, L_i . Let $d_i = c_i - 1/2$. In particular, $c_0 = a_0$ and $d_0 = b_0$.

In general, given two lines with proportions of zeroes a_0 and a_1 , the proportion of zeroes in their pad product is

$$a_0a_1 + (1 - a_0)(1 - a_1) = 2a_0a_1 + 1 - a_0 - a_1.$$

In our case, the i -th pad product can be obtained from the $(i - 1)$ -st pad product and the i -th line, so for $i > 0$ we have

$$(4) \quad c_i = 2c_{i-1}a_i + 1 - c_{i-1} - a_i,$$

$$(5) \quad d_i + 1/2 = 2(d_{i-1} + 1/2)(b_i + 1/2) + 1 - (d_{i-1} + 1/2) - (b_i + 1/2)$$

$$(6) \quad d_i = 2d_{i-1}b_i = 2^i \prod_{j=0}^{i-1} b_j.$$

Since $b_i \in (-1/2, 1/2)$ for all $i \in \mathbb{N}$, $|d_i|$ is strictly decreasing for every pad. Note that $d_i \neq 0$ for all $i \in \mathbb{N}$.

3.2.1. Worst Case Scenario. If digits are to be approximately equidistributed, c_i has to approach $1/2$ and d_i has to approach 0. The “worst case” clearly happens when each $|b_i|$ is as large as possible, i.e. when each line contains but one 0 or but one 1. The maximum period length among first i lines of the pad grows at least as fast as primes, and possibly faster. It is easy to come up with an example where period lengths grow too fast for d_i to converge to zero, so let us suppose that period lengths are primes above 2.¹

If we agree to write ρ_i for i -th prime, then the worst case could be conceived as

$$b_i = \frac{1}{2} - \frac{1}{\rho_{i+2}},$$

and we need to inspect what happens to d_i as $i \rightarrow \infty$. (We wrote ρ_{i+2} because i counts from zero while the first prime useful to us is $\rho_2 = 3$.) Well, in agreement with (6), d_i

¹ Which is the *best case* scenario with respect to the period growth, but it is also the most practical one if we actually *want* to have as many lines as possible, given a fixed amount of computer memory.

converges to zero as $i \rightarrow \infty$ iff the following product converges to zero:

$$(7) \quad \prod_{i=1}^{\infty} 2 \left(\frac{1}{2} - \frac{1}{\rho_{i+1}} \right) = \prod_{i=1}^{\infty} \left(1 - \frac{2}{\rho_{i+1}} \right).$$

Thanks to Peter Barendse for pointing out how this is related to the Euler Product [2], and to my office mate Myoungil Kim for the slick algebra (9) ahead. What we have in (7) is the multiplicative inverse of

$$(8) \quad \prod_{i=1}^{\infty} \frac{1}{1 - 2\rho_{i+1}^{-1}},$$

with the latter diverging to infinity just in case if the former approaches zero. Which it does, and we prove it by showing that a slower-growing product diverges:

$$(9) \quad \prod_{i=1}^{\infty} \frac{1}{1 - \rho_i^{-1}} = \left(1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} \right) \left(1 + \frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^3} \right) \left(1 + \frac{1}{5} + \frac{1}{5^2} + \frac{1}{5^3} \right) \dots$$

$$(10) \quad = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$$

$$(11) \quad = \infty.$$

We may conclude that in the case when line periods grow as slow as primes, the proportion of zeroes in a pad product approaches 0.5 as the number of lines tends to ∞ .

3.2.2. *A More Reasonable Case.* Suppose that we have a somewhat more “typical” pad, one in which the proportion of zeroes in each line is at least $1/4$, and hence $b_i \leq 1/4$. This assumption is true of most lines in most pads. Then

$$(12) \quad |d_i| = \left| 2^i \prod_{j=0}^i b_j \right|$$

$$(13) \quad \leq 2^i \prod_{j=0}^i \frac{1}{4}$$

$$(14) \quad = \frac{1}{2^{i+2}},$$

i.e. in a “typical” pad, where zeroes and ones are approximately equidistributed, the proportion of zeroes in the pad product approaches $\frac{1}{2}$ very fast. In practice, one can easily compute an upper bound on $|d_i|$ in a given pad by counting lines with proportion of zeroes at least $1/4$.

4. WHAT NOW?

There are at least a couple of ways in which this work may be improved. One obvious direction for our inquiry is finding other, stronger properties of PC. Similarly to how we managed the digit distribution, we may analyze the distribution of longer strings. We think that the analysis would have to take a different shape. The thing about digits (and short strings in general) is that we may expect them to be well-distributed in the pad, and therefore in the pad product. But as the strings get longer, this is no longer true.

Another (and may be more fruitful) way to go is by improving the algorithm. For example, after producing one full period for a given pad, we may switch to a different

pad of the same size, and go on in that fashion until we cycle through all possible pads. Or, better yet, we may switch the pad after every p outputs, p and p_i being relatively prime for each line's period p_i . Now, if n is the fixed number of lines, then there are $2^{p_0+p_1+p_2+\dots+p_{n-1}}$ different pads, which places the upper bound for the period at

$$(15) \quad 2^{p_0+p_1+p_2+\dots+p_{n-1}} \prod_{i=0}^{n-1} p_i,$$

which is a nice improvement over (3). Moreover, in this context, it should become a lot easier to show that strings up to a certain length are (approximately) equidistributed. However, we should not abandon the option of proving nice things about the PC on a static pad, as the same facts may lead to yet nicer properties once the pad starts changing.

Finally, nothing at all has been said about predictability. Even if we succeed in proving that strings of length k are equidistributed in the pad product, what about strings much longer than k ? At present, we can only hope and pray that with k large enough (say, 10^3) the problem is intractable.

P.S. If we seed the PRNG by specifying a pad, the predictability problem may be related to solving systems of linear equations over a finite field: namely, over \mathbb{Z}_2 .

REFERENCES

- [1] Everybody. Prime number theorem. http://en.wikipedia.org/wiki/Prime_number_theorem.
- [2] Everybody. Riemann zeta function. http://en.wikipedia.org/wiki/Riemann_zeta_function.