THE POLLARD $p-1$ FACTORIZATION ALGORITHM

FRANCO CAFIERO

The Pollard $p-1$ factorization algorithm is a specialized method for finding prime factors p of an integer n such that $p-1$ has only "small" prime factors. We analyze the limitations of this algorithm for certain types of integers and offer informally a rough estimate of the smoothness bound B required to factor a given integer n.

Definition 1. *Let $n, B \in \mathbb{Z}$. We say that n is B-smooth if all of the prime factors of n are less than or equal to B. That is, for $n = p_1 p_2 ... p_r$ (for some $r \in \mathbb{Z}$) with p_i prime for $1 \leq i \leq r$, $p_i \leq B$ for all i.*

For a given integer n, the Pollard $p-1$ algorithm finds a prime factor p of n such that $p-1$ is B-smooth for a given B. A formal explanation of the algorithm can be found in Garrett's "Making, Braking Codes." For our purposes it is sufficient to explain that the algorithm requires a list of primes less than or equal to B. This is because the algorithm cycles through these primes until it finds a prime p_j such that $p-1$ is p_j-smooth.

Garrett also states that one of the algorithm's failure modes occurs when there are no prime factors of n that are B-smooth for a certain B. It is immediately evident from this fact that one can construct an integer that will resist $p-1$ factorization unless the smoothness bound B selected is inconveniently large.

Definition 2. *Let p be a prime number. If there exists a prime q such that $p = 2q+1$, then we say that p is a strong prime.*

Using the same notation as above, we call such primes q "strong" because they resist the Pollard $p-1$ factorization unless the smoothness bound B is set to p, where p has approximately the same number of digits as q. We state this more concisely in the following theorem:

Theorem 1. *Fix a smoothness bound B. If n is the product of two strong primes q and r with $q < r$, and if $q > 2B+1$, then n resists Pollard $p-1$ factorization.*

Proof. Define B, n, q and r as above. By definition, $q-1 = 2q'$ and $r-1 = 2r'$ for some primes q and r ($q < r$). We know that the algorithm fails when n has no B-smooth prime factors. So $q-1 = 2q'$; by assumption, $q > 2B+1$, so $q-1 = 2q' > 2B$ and then $q' > B$. Hence $q-1$ is not B-smooth, and neither is r since $q < r$. So n has no B-smooth factors and thus resists the Pollard $p-1$ factorization algorithm. □

It may seem that requiring $p-1$ to be the product of two and another larger prime is arbitrary; one could just as easily find strong primes q such that $q-1 = 2 \cdot 3 \cdot q'$ for some q' prime, or include more small prime factors. However, consider the task of constructing a 100-digit strong prime p with a fixed first digit. If we go about this by calculating, say, $2 \cdot 3 \cdot 5 \cdot p' + 1$ for 99- and 100-digit primes until the expression is prime, the resulting p' will be smaller than one calculated by the expression $2p'$

over the same primes; in the latter expression, p' is multiplied by fewer and smaller factors than in the former, so the p' in the latter can take larger values. After all, the purpose of strong primes in our case is to make Pollard $p-1$ factorization as slow and inefficient as possible by making the necessary smoothness bound B as large as possible.

Let n be the product of two strong primes p and q. For this n to factor by the $p-1$ algorithm, the smoothness bound B should be set very near $\sqrt{n}$. We know n has a prime factor less than or equal to $\sqrt{n}$, viz. p, and since $p-1=2p'$, p' is very close to p. So in the worst case when p is approximately $\sqrt{n}$, p' is approximately $\sqrt{n}$. Even if p is relatively far from $\sqrt{n}$, the large B should not impact runtime too adversely since our implementation of the algorithm cycles acendingly through the list of primes; so if p' is small, the algorithm will get to it (at which point it yields a factor and stops) long before it gets to B and stop.

In the case when n is the product of strong primes, the requirement of having B set to approximately $\sqrt{n}$ makes the algorithm as inefficient as trial division. We found no way to circumvent this blockade, but can offer the following information: we know that for such an n, we have $n = 4p'q' + 2p' + 2q' + 1$, which implies $n-1 = 4p'q' + 2p' + 2q'$. Since n is known, we can derive the following congruence:

$$4p'q' + 2p' + 2q' \equiv 0 \pmod{n-1}$$

If this congruence has a solution (p_o, q_o) with both values prime, then the factorization follows immediately. The problem, then, becomes one of finding an efficient algorithm (if indeed one exists) for solving this congruence. We are guaranteed only the existence of such a solution (because of the existence of n), but not a general method for solving the congruence.

For the times when n is the product of two primes, neither of which is strong, it is difficult to pinpoint an appropriate and reasonably small smoothness bound b. This problem is directly related to the question of how large are the prime factors of a composite integer, which is a non-trivial query. However, we found that $\sqrt[3]{n}$ works fairly well as a smoothness bound for a given integer n.

The final issue that was investigated in this portion of the research concerns the distribution of strong primes. In order to perform test calculations on n composed of strong primes, it was necessary to generate as many strong primes as possible. We were able to generate all strong primes less than 100,000,000 by taking a list of all primes less than 50,000,000, multiplying each by 2, adding 1 to the product and finally testing for primality. The question of whether or not there exists an infinitude of strong primes follows naturally from this calculation. Although we were unable to demonstrate that there are infinitely many strong primes, we strongly suspect that this is the case (we hope to prove this conjecture soon). Before proceeding, let us introduce a variation on the arithmetic function $\pi(n)$:

Definition 3. *Let n be a positive integer. Then $\pi_s(n) := |\{a \in \mathbb{Z} : a \text{ is a strong prime, } a \leq n\}|$.*

We have observed that for $n < 15000$, $\pi_s(n) \approx \frac{n}{(\ln n)^2}$ with an average percent difference of 5.6% .Unfortunately, this handsome approximation falls apart entirely after 15000. However, the surprising accuracy of this approximation led us to consider other powers of $\ln n$, most notably 2.1. Though quite inaccurate for $n < 1,000,000$, $\frac{n}{(\ln n)^{2.1}}$ is very accurate for $1,000,000 \leq n \leq 2,500,000$ (with percent error around 6%), and extremely accurate for $2,500,000 \leq n \leq 100,000,000$;

after 9,400,000, the percent error is consistently at or below 5%, and falls as low as approximately 4.07% near 100,000,000. As n approaches 100,000,000 the percent error decreases very consistently. (Note that, as stated before, we only have available all strong primes less than 100,000,000, so this is as far as we can go currently)

As we have just seen, $\pi_s(n) \approx \frac{n}{(\ln n)^{2.1}}$ for $n > 2,500,000$ (within our available data, of course). We know also that the set of strong primes is a proper subset of the set of all prime numbers, so $\pi_s(n)$ is bounded from above by $\frac{n}{\ln n}$. We then conjecture that

$$\frac{n}{(\ln n)^{2.1}} \leq \pi_s(n) \leq \frac{n}{\ln n}$$

and, more boldly, that

$$\lim_{n\to\infty} \frac{\pi_s(n)}{\frac{n}{(\ln n)^{2.1}}} = 1$$

If this conjecture is not the case, we suspect that it may work for some power a of $\ln n$, $2 < a < 3$.