J. Cryptology (2003) 16: 287–289
DOI: 10.1007/s00145-003-0051-5

Journal of
CRYPTOLOGY

Generating Random Factored Numbers, Easily

Adam Kalai
Department of Mathematics,
Massachusetts Institute of Technology,
77 Massachusetts Avenue,
Cambridge, MA 02139, U.S.A.
akalai@mit.edu

Communicated by Moni Naor

Received August 2000 and revised May 2003
Online publication 5 September 2003

Consider the problem of generating a random "pre-factored" number, that is, a uniformly random number between 1 and n, along with its prime factorization. Of course, one could pick a random number in this range and try to factor it, but there are no known polynomial-time factoring algorithms. In his dissertation, Bach presents an efficient algorithm for this problem [1], [2]. Here, we present a significantly simpler algorithm and analysis for the same problem. Our algorithm is, however, a $\log(n)$ factor less efficient.

Algorithm

Input: Integer $n > 0$.

Output: A uniformly random number $1 \le r \le n$.

1. Generate a sequence $n \ge s_1 \ge s_2 \ge \cdots \ge s_l = 1$ by choosing $s_1 \in \{1, 2, \ldots, n\}$ and $s_{i+1} \in \{1, 2, \ldots, s_i\}$, until reaching 1.
2. Let r be the product of the **prime** s_i's.
3. If $r \le n$, output r with probability r/n.
4. Otherwise, RESTART.

A common class exercise is pick a random number between 1 and n using a coin with $Pr(H) = Pr(T) = 1/2$. Instead suppose we had n coins $c_1, c_2, \ldots, c_n$ where,

$$\text{coin } c_i \text{ has} \quad Pr(H) = \frac{1}{i} \quad \text{and} \quad Pr(T) = 1 - \frac{1}{i}.$$

Hypothetically, one slow way to pick a number between 1 and n is first to flip c_n and choose n if it is H, otherwise flip c_{n-1} and choose $n-1$ if it is H, and so on.

Claim 1. *One way to choose a uniformly random* $1 \le m \le n$ *is to flip coins* $c_n, c_{n-1}, \dots$ *until we get* H *on some coin* c_m.

Proof. By induction. The base case $n = 1$ is trivial. For a general n, we pick n with probability $1/n$ and otherwise, by induction hypothesis, all $1 \le m \le n-1$ are equally likely. □

Claim 2. *The output of our algorithm is uniform in* $\{1, 2, \dots, n\}$.

Proof. Imagine that in step 1 we chose s_1 by flipping coins $c_n, c_{n-1}, \dots$, until we got T on some c_{s_1}, and chose s_2 by flipping $c_{s_1}, c_{s_1-1}, \dots$, and so on. (Of course, in practice we would use some more efficient method.) Every coin will be flipped, and the number of occurrences of a number m in the sequence is the number of H's we saw on coin c_m before T.

Thus, in step 2, we get a particular $r = \prod_{p \le n} p^{\alpha_p}$ with probability

$$\begin{aligned} Pr\left[r = \prod_{p \le n} p^{\alpha_p}\right] &= Pr[\wedge_{p \le n} \text{ we had } \alpha_p \ H\text{'s followed by } T \text{ on coin } c_p] \\ &= \prod_{p \le n} \left(\frac{1}{p}\right)^{\alpha_p} \left(1 - \frac{1}{p}\right) \\ &= \frac{1}{r} M_n, \end{aligned}$$

where $M_n = \prod_{p \le n}(1 - 1/p)$. Next, the probability of generating such a $1 \le r \le n$ *and* outputting it in step 3 is

$$\frac{M_n}{r}\frac{r}{n} = \frac{M_n}{n}.$$

Since this is the same for every $1 \le r \le n$, each time we reach step 3, we either output a uniformly random $1 \le r \le n$ or restart. □

Intuition. The above analysis shows that in fact every number m occurs at least once in the sequence with probability $1/m$, and at least k times with probability $1/m^k$. This matches the intuition that a prime $p \ll n$ divides a random number in $1 \le r \le n$ at least once with probability $\approx p$ and at least k times with probability $\approx 1/p^k$.

Claim 3. *The expected number of primality tests is* $O(\log^2 n)$.

Proof. Since the probability of outputting any particular $1 \le r \le n$ is M_n/n, the probability of outputting any number in step 3 is $n(M_n/n) = M_n$. If we refer to a *round* as an execution of steps 1, 2, and 3, then the probability of reaching round t is $(1 - M_n)^t$. During a round, we test m with probability $1/m$, the probability we get at least one H on c_m. So

$$Pr[m \text{ is tested during round } t] = \frac{(1 - M_n)^t}{m}.$$

Thus the expected total number of primality tests is[1]

$$\sum_{t=0}^{\infty}\sum_{m=1}^{n}\frac{(1-M_n)^t}{m} = H_n \sum_{t=0}^{\infty}(1-M_n)^t = \frac{H_n}{M_n}.$$

Since $H_n \le 1+\ln n$ and $1/M_n \approx 1.78\ln n$ (Mertens' theorem [3]), H_n/M_n is $O(\log^2 n)$. □

Acknowledgments

I thank Manuel Blum, Michael Rabin, Doug Rohde, Yael Tauman, and the referees for helpful comments.

References

[1] E. Bach, *Analytic Methods in the Analysis and Design of Number-Theoretic Algorithms*, MIT Press, Cambridge, MA, 1985.

[2] E. Bach, How to generate factored random numbers, *SIAM Journal on Computing*, vol. 17 (1988), pp. 179–193.

[3] E. Bach and J. Shallit, *Algorithmic Number Theory*, MIT Press, Cambridge, MA, 1996.

[1] It is tempting to take a shortcut and argue that the expected number of total primality tests is H_n/M_n because the expected number of rounds is $1/M_n$ and the expected number of tests per round is H_n, but this assumes independence.