A probability-based proof that any deterministic algorithm which always computes a prime factor of n cannot be polynomial-time

Bhupinder Singh Anand

Submission of January 1, 2015: Calcutta Statistical Association Bulletin

Abstract

We define the residues $r_i(n)$ for all $n \geq 2$ and all $i \geq 2$ such that $r_i(n) = 0$ if, and only if, i is a divisor of n. We then show that the joint probability $\mathbb{P}(p_i|n \cap p_j|n)$ of two unequal primes $p_i,\ p_j$ dividing any integer n is the product $\mathbb{P}(p_i|n).\mathbb{P}(p_j|n)$. We conclude that the prime divisors of any integer n are mutually independent; and that any deterministic algorithm which always computes a prime factor of n cannot be polynomial-time.[1]

1 The prime divisors of any integer n are mutually independent

We define the residues $r_i(n)$ for all $n \geq 2$ and all $i \geq 2$ as below:

Definition 1 $n + r_i(n) \equiv 0\ (mod\ i)$ *where* $i > r_i(n) \geq 0$.

Since each residue $r_i(n)$ cycles over the i values $(i-1, i-2, \ldots, 0)$, these values are all incongruent and form a complete system of residues[2] *mod* i.

We immediately conclude that:

[1]Keywords: complete system of incongruent residues, integer factorising algorithm, mutually independent prime divisors, polynomial-time.
MSC Nos: 03D15, 11A07, 11A41, 11A51, 11N36, 11Y05, 11Y11, 11Y16, 68Q15

[2][HW60], p.49.

Lemma 1 *$r_i(n) = 0$ if, and only if, i is a divisor of n.* □

By the standard definition of the probability $\mathbb{P}(e)$ of an event e[3], it immediately also follows that:

Lemma 2 *For any $n \geq 2$, $i \geq 2$ and any given integer $i > u \geq 0$, the probability $\mathbb{P}(r_i(n) = u)$ that $r_i(n) = u$ is $1/i$, and the probability $\mathbb{P}(r_i(n) \neq u)$ that $r_i(n) \neq u$ is $1 - 1/i$.* □

We note the standard definition[4]:

Definition 2 *Two events e_i and e_j are mutually independent for $i \neq j$ if, and only if, $\mathbb{P}(e_i \cap e_j) = \mathbb{P}(e_i).\mathbb{P}(e_j)$.*

We then have that:

Lemma 3 *If p_i and p_j are two primes where $i \neq j$ then, for any n, we have:*

$$\mathbb{P}((r_{p_i}(n) = u) \cap (r_{p_j}(n) = v)) \\ = \mathbb{P}(r_{p_i}(n) = u).\mathbb{P}(r_{p_j}(n) = v)$$

where $p_i > u \geq 0$ and $p_j > v \geq 0$.

Proof: The $p_i.p_j$ numbers $v.p_i + u.p_j$, where $p_i > u \geq 0$ and $p_j > v \geq 0$, are all incongruent and form a complete system of residues[5] *mod* $(p_i.p_j)$. Hence:

$$\mathbb{P}((r_{p_i}(n) = u) \cap (r_{p_j}(n) = v)) = 1/p_i.p_j$$

By Lemma 2:

$$\mathbb{P}(r_{p_i}(n) = u).\mathbb{P}(r_{p_j}(n) = v) = (1/p_i)(1/p_j).$$

The lemma follows. □

If $u = 0$ and $v = 0$ in Lemma 3, so that both p_i and p_j are prime divisors of n, we immediately conclude by Definition 2 that:

[3][Ko56], Chapter I, §1, Axiom III, p.2; see also [GS97], Chapter 1, §1.2, Definition 1.2, p.19.

[4][Ko56], Chapter VI, §1, Definition 1, p.57 and §2, p.58; see also [GS97], Chapter 4, §4.1, Theorem 4.1, p.140.

[5][HW60], p.52, Theorem 59.

Corollary 1 $\mathbb{P}((r_{p_i}(n) = 0) \cap (r_{p_j}(n) = 0)) = \mathbb{P}(r_{p_i}(n) = 0).\mathbb{P}(r_{p_j}(n) = 0)$. □

Corollary 2 $\mathbb{P}(p_i|n \ \cap \ p_j|n) = \mathbb{P}(p_i|n).\mathbb{P}(p_j|n)$. □

Theorem 1 *The prime divisors of any integer n are mutually independent.* □

2 Integer Factorising cannot be polynomial-time

We note the standard definition[6]:

Definition 3 *A deterministic algorithm[7] computes a number-theoretical function $f(n)$ in polynomial-time if there exists k such that, for all inputs n, the algorithm computes $f(n)$ in $\leq (log_e\ n)^k +$ k steps.*

It now follows from Theorem 1 that:

Corollary 3 *Any deterministic algorithm that always computes a prime factor of n cannot be polynomial-time.*

Proof: Any computational process to identify a prime divisor of n must necessarily appeal to a logical operation for identifying such a factor.

Since n is a prime if, and only if, it is not divisible by any prime $p \leq \sqrt{n}$, and n may be the square of a prime, it follows from Theorem 1 that we necessarily require at least one logical operation for each prime $p \leq \sqrt{n}$ in order to logically identify a prime divisor of n.

Since the number of such primes is of the order $O(n/log_e\ n)$, any deterministic algorithm that always computes a prime factor of n cannot be polynomial-time—i.e. of order $O((log_e\ n)^c)$ for any c—in the length of the input n. The corollary follows. □

[6]cf. [Cook], p.1; also [Br00], p.1, fn.1: "For a polynomial-time algorithm the expected running time should be a polynomial in the length of the input, i.e. $O((logN)^c)$ for some constant c".

[7]A deterministic algorithm computes a mathematical function which has a unique value for any input in its domain, and the algorithm is a process that produces this particular value as output.

References

[Br00] Richard P. Brent. 2000. *Recent Progress and Prospects for Integer Factorisation Algorithms*. In *Computing and Combinatorics*, Lecture Notes in Computer Science, Volume 1858, 2000, pp.3-22, Springer, New York/Heidelberg.

[Cook] Stephen Cook. 2000. *The P versus NP Problem*. Official description provided for the Clay Mathematical Institute, Cambridge, Massachusetts.

[GS97] Charles M. Grinstead and J. Laurie Snell. 1997. *Introduction to Probability*. Second Revised Edition, 1997, American Mathematical Society, Rhode Island, USA

[HW60] G. H. Hardy and E. M. Wright. 1960. *An Introduction to the Theory of Numbers*. 4th edition. Clarendon Press, Oxford.

[Ko56] A. N. Kolmogorov. 1933. *Foundations of the Theory of Probability*. Second English Edition. Translation edited by Nathan Morrison. 1956. Chelsea Publishing Company, New Yourk.

[An05] Bhupinder Singh Anand. 2005. *Three Theorems on Modular Sieves that suggest the Prime Difference is* $O(\pi(p(n)^{1/2}))$. Private investigation.

Authors postal address: #1003 B Wing, Lady Ratan Tower, Dainik Shivner Marg, Worli, Mumbai - 400 018, Maharashtra, India. *Email*: bhup.anand@gmail.com.