SIAM J. COMPUT.
Vol. 3, No. 1, March 1974

THE COMPUTING TIME OF THE EUCLIDEAN ALGORITHM*

GEORGE E. COLLINS†

Abstract. The minimum, maximum and average computing times of the classical Euclidean algorithm are derived. With positive integer inputs of lengths m and n, and with output (greatest common divisor) of length k, $m \geqq n \geqq k$, the minimum is shown to be codominant with $n(m - n + 1) + k(n - k + 1)$, while both the maximum and the average are shown to be codominant with $n(m - k + 1)$.

Key words. Euclidean algorithm, greatest common divisor, arithmetic algorithms, algorithm analysis

1. Introduction. Knuth [11], Dixon [6], [7] and Heilbronn [8] have recently investigated in considerable depth the average number of divisions performed in the Euclidean algorithm for integers. Although many interesting questions remain unanswered, the relatively elementary result of Dixon in [7] already suffices to completely determine the average computing time of the Euclidean algorithm to within a constant factor, which is in any case dependent on the particular computer used and inessential details of the implementation. Such a determination of the average computing time of the Euclidean algorithm is the main result of the present paper. The maximum and minimum computing times of the Euclidean algorithm for integers will also be derived since, although their determination is quite elementary, they have apparently not previously been published. These computing times are all derived as functions of three variables, namely the lengths of the two inputs and the length of the resulting g.c.d. (greatest common divisor). Previous results on the computing time of the Euclidean algorithm ([2] and [11, § 4.5.2, Exercise 30]) have been limited to upper bounds on the maximum computing time.

2. Dominance and codominance. The relations of dominance and codominance between real-valued functions were introduced in [3], where they were used in the analysis of the computing time of an algorithm for polynomial resultant calculation. The related concepts and notation have subsequently been adopted by several authors, for example, Brown [1], Heindel [9] and Musser [12]. The definitions and some fundamental properties will be repeated here since they will not yet be familiar to many readers.

If f and g are real-valued functions defined on a common domain S, we say that f *is dominated by* g, and write $f \preceq g$, in case there is a positive real number c such that $f(x) \leqq c \cdot g(x)$ for all $x \in S$. We also say that g *dominates* f, and write $g \succeq f$. Dominance is clearly a reflexive and transitive relation. It is important to note that the definition is not restricted to functions of one variable since the elements of S may be n-tuples.

* Received by the editors February 12, 1973, and in revised form September 17, 1973. This work was supported by the National Science Foundation under Grant GJ-30125X, by the Wisconsin Alumni Research Foundation, and in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense under Grant SD-183.

† Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706.

Knuth [10, pp. 104–108] defines $f(x) = O(g(x))$ in case there is a positive constant c such that $|f(x)| \leqq c \cdot |g(x)|$. As long as one is dealing only with non-negative-valued functions, this formally coincides with the above definition of $f \preceq g$. Although Knuth implies that this definition is applicable only when f and g are functions of one variable, he in fact uses it for functions of more than one variable (e.g., [11, p. 388]) in a manner which is consistent with our definition. Thus dominance is apparently a new notation and terminology but not a new concept. Although Knuth discussed at length the logical weaknesses of the O-notation, he chose not to abandon it in favor of the more natural notation of an order relation.

If $f \preceq g$ and $g \preceq f$, then we say that f and g are *codominant*, and write $f \sim g$. Codominance is clearly an equivalence relation. If $f \preceq g$ but not $g \preceq f$, then we say that f *is strictly dominated by* g, and write $f \prec g$. We may also say that g *strictly dominates* f, and write $g \succ f$. Strict dominance is clearly irreflexive and transitive. Whereas the O-notation has no counterparts for the codominance and strict dominance relations, it will become apparent that these are important concepts in algorithm computing time analyses. Furthermore, the O-notation has a somewhat different meaning in asymptotic analysis than the one used by Knuth (see, e.g., [5]).

If f and g are functions defined on S and S_1 is a subset of S, it will often be convenient to write $f \preceq g$ *on* S_1 in case $f_1 \preceq g_1$, where f_1 and g_1 are the functions f and g restricted to S_1. Also, if $S \subseteq S_1 \times \cdots \times S_n$, a Cartesian product, we will denote by f_a the function f restricted to $(\{a\} \cap S_2 \times \cdots \times S_n) \cap S$; that is, $f_a(x_2, \cdots, x_n) = f(a, x_2, \cdots, x_n)$ for $(a, x_2, \cdots, x_n) \in S$. Similarly we may *fix* any other of the n variables of f.

Dominance and codominance have the following fundamental properties, most of which were listed by Musser in [12].

THEOREM 1. *Let f, f_1, f_2, g, g_1 and g_2 be nonnegative real-valued functions on S, and let c be a positive real number. Then*

(a) $f \sim cf$;
(b) *if* $f_1 \preceq g_1$ *and* $f_2 \preceq g_2$, *then* $f_1 + f_2 \preceq g_1 + g_2$ *and* $f_1 f_2 \preceq g_1 g_2$;
(c) *if* $f_1 \preceq g$ *and* $f_2 \preceq g$, *then* $f_1 + f_2 \preceq g$;
(d) $\max(f, g) \sim f + g$;
(e) *if* $1 \preceq f$ *and* $1 \preceq g$, *then* $f + g \preceq fg$;
(f) *if* $1 \preceq f$, *then* $f \sim f + c$;
(g) *if* $S \subseteq S_1 \times \cdots \times S_n$ *and* $a \in S_1$, *then* $f \preceq g$ *implies* $f_a \preceq g_a$;
(h) *if* $S = S_1 \cup S_2$, *then* $f \preceq g$ *on* S_1 *and* $f \preceq g$ *on* S_2 *implies* $f \preceq g$ *on* S.

Proof. These properties follow immediately from the definition, except for (e). To prove (e), apply (b) to $f \preceq f$ and $1 \preceq g$, obtaining $f \preceq fg$. Similarly $g \preceq fg$, so $f + g \preceq fg$ by (c). ∎

3. Computing time functions. Let A be any algorithm, and let S be the set of all valid inputs to A. In general, S will be denumerable, and its elements may be n-tuples. We associate with A a computing time function t_A defined on S, the positive integer $t_A(x)$ being the number of basic operations performed by the algorithm A when presented with the input x. This assumes that the algorithm

is unambiguously specified in terms of some finite set of basic operations. Changing the set of basic operations (as in reprogramming the algorithm for a different computer) will result in changing the computing time function t_A. Alternatively, we could take the view that this represents a change in the algorithm. However, if B_1 and B_2 are two sets of basic operations such that each operation in B_1 can be performed by a fixed sequence of operations in B_2, and vice versa, then the computing time functions associated with B_1 and B_2 for any algorithm A are codominant, and we will concern ourselves only with the codominance equivalence class of t_A. Thus the choice of basic operations is somewhat arbitrary. We assume a choice which is consistent with any of the existing, or conceivable, random access digital computers but, in order to avoid the triviality of finiteness, with a memory which is indefinitely expandable.

The function t_A is frequently too complex to be of interest for direct study. Instead, we ordinarily decompose S into a disjoint union $S = \bigcup_{n=1}^{\infty} S_n$, where each S_n is a nonempty finite set, S being a denumerable set. The choice of decomposition is made on the basis of some prior knowledge or some conjecture about the general behavior of t_A. Relative to a decomposition $\mathscr{S} = \{S_1, S_2, S_3, \cdots\}$ of S we define maximum, minimum and average computing time functions, t_A^+, t_A^- and t_A^*, on $\mathscr{S}$ as follows, where $|S_n|$ denotes the number of elements of S_n:

$$t_A^+(S_n) = \max_{x \in S_n} t_A(x), \tag{1}$$

$$t_A^-(S_n) = \min_{x \in S_n} t_A(x), \tag{2}$$

$$t_A^*(S_n) = \left\{ \sum_{x \in S_n} t_A(s) \right\} / |S_n|. \tag{3}$$

As an illustration, and in preparation for our analysis of the Euclidean algorithm, let us consider the computing times of the classical algorithms for arithmetic operations, that is, addition, subtraction, multiplication and division, of arbitrarily large integers. We assume that all integers are represented in radix form relative to an integral base $\beta \geqq 2$, as discussed by Knuth in [11, § 4.3]. We know that the computing times of these algorithms depend on the lengths of the inputs.

Following Musser [12] we denote by $L_\beta(a)$ the *β-length* of the integer a, that is, the number of digits in the radix form of a relative to the base β. If $\lceil x \rceil$ is the ceiling function of x, the least integer greater than or equal to x, and $\lfloor x \rfloor$ is the floor function of x, the greatest integer less than or equal to x, we have

$$L_\beta(a) = \lceil \log_\beta (|a| + 1) \rceil = \lfloor \log_\beta |a| \rfloor + 1 \tag{4}$$

for $a \neq 0$, and we define $L_\beta(0) = 1$.

In most contexts the base β is fixed, and we write simply $L(a)$ for the *length* of a. The omission of the subscript is further justified by the observation that, γ being any other base, we have

$$L_\beta \sim L_\gamma, \tag{5}$$

where L_β and L_γ are functions defined on the set I of all integers. In fact, we can use the definition (4) when a is any real number, and we then have

$$L_\beta(a) \sim \ln(|a| + 2) \quad \text{on } R, \tag{6}$$

where ln is the natural logarithm and R is the set of all real numbers, and (6) clearly implies (5). The length function also has the following easily verified fundamental properties (here I is the set of integers):

$$L(a \pm b) \preceq L(a) + L(b) \quad \text{for } a, b \in I, \tag{7}$$

$$L(ab) \sim L(a) + L(b) \quad \text{for } a, b \in I - \{0\}, \tag{8}$$

$$L([a/b]) \sim L(a) - L(b) + 1 \quad \text{for } a, b \in I \quad \text{and} \quad |a| \geqq |b| > 0, \tag{9}$$

where $[x] = \lfloor x \rfloor$ for $x \geqq 0$ and $[x] = \lceil x \rceil$ for $x < 0$.

THEOREM 2. (a) *Let* $S = \{(a_1, \cdots, a_n) : n \geqq 1$ *and* $a_1, \cdots, a_n \in I\}$. *Then* $L(\prod_{i=1}^n a_i) \leqq \sum_{i=1}^n L(a_i)$ *on* S.

(b) *Let* $\bar{S} = \{(a_1, \cdots, a_n) : n \geqq 1$ *and* $a_1, \cdots, a_n \in I - \{-1, 0, 1\}\}$. *Then* $L(\prod_{i=1}^n a_i) \sim \sum_{i=1}^n L(a_i)$ *on* $\bar{S}$.

Proof. $L(ab) \leqq L(a) + L(b)$ for $a, b \in I$, so $L(\prod_{i=1}^n a_i) \leqq \sum_{i=1}^n L(a_i)$ by induction on n, proving (a). To prove (b), assume first that $2 \leqq |a_i| < \beta$ for $1 \leqq i \leqq n$. Then

$$L\left(\prod_{i=1}^n a_i\right) \geqq \log_\beta \prod_{i=1}^n |a_1| = (\log_\beta 2) \log_2 \prod_{i=1}^n |a_i|$$

$$\geqq (\log_\beta 2) \log_2 2^n = (\log_\beta 2) n = (\log_\beta 2) \sum_{i=1}^n L(a_i),$$

so $\sum_{i=1}^n L(a_i) \leqq (\log_2 \beta) L(\prod_{i=1}^n a_i)$. Next, assume $L_\beta(a_i) \geqq 2$ for $1 \leqq i \leqq n$, and let $l_i = L_\beta(a_i)$. Then

$$L\left(\prod_{i=1}^n a_i\right) \geqq \log_\beta \left(\prod_{i=1}^n |a_i|\right) \geqq \log_\beta \left(\prod_{i=1}^n \beta^{l_i - 1}\right)$$

$$= \sum_{i=1}^n (l_i - 1) \geqq \left(\sum_{i=1}^n l_i\right) \Big/ 2,$$

so $\sum_{i=1}^n L(a_i) \leqq 2L(\prod_{i=1}^n a_i)$.

Combining these two cases, we assume $L(a_i) = 1$ for $1 \leqq i \leqq m$ and $L(a_i) \geqq 2$ for $m + 1 \leqq i \leqq n$. Then

$$\sum_{i=1}^n L(a_i) \leqq (\log_2 \beta) L\left(\prod_{i=1}^m a_i\right) + 2L\left(\prod_{i=m+1}^n a_i\right)$$

$$\leqq 2(\log_2 \beta)\left\{L\left(\prod_{i=1}^m a_i\right) + L\left(\prod_{i=m+1}^n a_i\right)\right\} \leqq 4(\log_2 \beta) L\left(\prod_{i=1}^n a_i\right)$$

since $L(a) + L(b) \leqq 2L(ab)$ for $a, b \in I - \{0\}$. ∎

As an immediate corollary of Theorem 2, we have

$$L(a^b) \sim bL(a) \quad \text{for } a, b \in I, \quad |a| \geqq 2 \quad \text{and} \quad b > 0. \tag{10}$$

If A, M and D are the classical algorithms for addition (or subtraction), multiplication and division, respectively, as described in [11, § 4.3], then we clearly have

$$t_A(a,b) \sim L(a) + L(b) \quad \text{for } a,b \in I - \{0\}, \tag{11}$$

$$t_M(a,b) \sim L(a) \cdot L(b) \quad \text{for } a,b \in I - \{0\}, \tag{12}$$

$$t_D(a,b) \sim L(b) \cdot L([a/b]) \quad \text{for } a,b \in I \text{ and } |a| > |b| > 0. \tag{13}$$

Thus, for these algorithms, the natural decomposition of the set $S = \{(a,b): a,b \in I\}$ consists of the sets $S_{m,n} = \{(a,b): L(a) = m \text{ and } L(b) = n\}$. If we write $t^+(m,n)$ in place of $t^+(S_{m,n})$, and similarly for t^- and t^*, then from (11), (12) and (13), and using (9), we have

$$t_A^+(m,n) \sim t_A^-(m,n) \sim t_A^*(m,n) \sim m + n, \tag{14}$$

$$t_M^+(m,n) \sim t_M^-(m,n) \sim t_A^*(m,n) \sim mn, \tag{15}$$

$$t_D^+(m,n) \sim t_D^-(m,n) \sim t_D^*(m,n) \sim n(m - n + 1) \quad \text{for } m \geqq n. \tag{16}$$

Thus for these algorithms the maximum, minimum and average computing times all coincide. This will not be the case for the Euclidean algorithm, to which we now turn.

4. The maximum and minimum computing times. For simplicity, and without loss of generality, we will consider the following version of the Euclidean algorithm, for which the permissible inputs are the pairs (a,b) of positive integers with $a \geqq b$. The output of the algorithm is the positive integer $c = \text{g.c.d.}(a,b)$.

ALGORITHM E.

Step 1. [Initialize.] $c \leftarrow a$; $d \leftarrow b$.

Step 2. [Divide.] Compute the quotient q and remainder r such that $c = dq + r$ and $0 \leqq r < d$, using Algorithm D (classical algorithm for division).

Step 3. [Test for end.] $c \leftarrow d$; $d \leftarrow r$; if $d \neq 0$, go to Step 2.

Step 4. Return.

This algorithm computes two sequences, $(a_1, a_2, \cdots, a_{l+2})$ and $(q_1, q_2, \cdots, q_l)$, such that $a_1 = a$, $a_2 = b$, $a_i = q_i a_{i+1} + a_{i+2}$ with $0 \leqq a_{i+2} < a_{i+1}$ for $1 \leqq i \leqq l$, and $a_{l+2} = 0$. $a_1, \cdots, a_{l+1}$ are the successive values assumed by the variable c, and $q_1, \cdots, q_l$ are the successive values assumed by the variable q. $(a_1, \cdots, a_{l+2})$ is called the *remainder sequence* of (a,b) and $(q_1, \cdots, q_l)$ is called the *quotient sequence* of (a,b). Steps 2 and 3 are each executed l times; this is the number of divisions performed, which we denote by $D(a,b)$.

By (13), the computing time for the ith execution of Step 2 is $\sim L(q_i)L(a_{i+1})$. The computing time for the ith execution of Step 3 is certainly dominated by $L(a_{i+1})$ since it at most requires copying the digits of a_{i+1} and a_{i+2}. In an implementation of the algorithm in which a large integer is represented by the list of its digits (e.g., [4]), such copying is unnecessary, and the computing time for

each execution of Step 3 is ~ 1. For the same reason, we will assume that the single executions of Steps 1 and 4 have computing times ~ 1. We then have

$$t_E(a,b) \sim \sum_{i=1}^{l} L(q_i) \cdot L(a_{i+1}). \tag{17}$$

If instead we were to assume that copying is required in Steps 1 and 3, (17) would still hold after adding $L(a_1)$ to the right-hand side. But $L(a_1) \sim L(q_1) + L(a_2) \preceq L(q_1)L(a_2)$, so (17) holds in any case.

From (17) we will derive the maximum, minimum and average computing times of Algorithm E, by analyzing the possible distributions of values of the a_i and q_i, obtaining the codominance equivalence classes of these computing times as functions of $L(a)$, $L(b)$ and $L(c)$. Thus we consider the decomposition of S into the sets

$$S_{m,n,k} = \{(a,b) : L(a) = m \text{ and } L(b) = n \text{ and } L(\text{g.c.d.}(a,b)) = k\}, \tag{18}$$

with $m \geqq n \geqq k \geqq 1$. We may verify that each set $S_{m,n,k}$ is nonempty as follows. If $m = k$, then $(\beta^{m-1}, \beta^{m-1}) \in S_{m,n,k}$. If $m > k$, let $a = \beta^{m-1} + \beta^{k-1}$ and $b = \beta^{n-1}$. Then $c = \text{g.c.d.}(a,b) = \beta^{k-1}$, $L(a) = m$, $L(b) = n$ and $L(c) = k$, so $(a,b) \in S_{m,n,k}$. As above, we will write $t_E^+(m,n,k)$ in place of $t_E^+(S_{m,n,k})$, and similarly for t_E^- and t_E^*.

THEOREM 3. $t_E^+(m,n,k) \preceq n(m-k+1)$.

Proof. Since $b = a_2 > a_3 > \cdots > a_{l+1}$, we have by (17) that

$$t_E(a,b) \preceq L(b) \sum_{i=1}^{l} L(q_i). \tag{19}$$

Since $L(q_i) \leqq L(q_i + 1)$ and $q_l \geqq 2$ we obtain, by Theorem 2,

$$\sum_{i=1}^{l} L(q_i) \preceq L\left(q_l \prod_{i=1}^{l-1} (q_i + 1)\right). \tag{20}$$

Since $a_i = q_i a_{i+1} + a_{i+2} > q_i a_{i+2} + a_{i+2}$, we have $q_i + 1 < a_i/a_{i+2}$ for $i < l$, and hence $\prod_{i=1}^{l-1}(q_i + 1) < a_1 a_2 / a_l a_{l+1}$. Combining this with $q_l = a_l/a_{l+1}$ yields

$$q_l \prod_{i=1}^{l-1} (q_i + 1) \leqq ab/c^2. \tag{21}$$

Since $L(ab/c^2) \leqq L(a^2/c^2) \sim L(a/c) \sim L(a) - L(c) + 1$, (19), (20) and (21) yield

$$t_E(a,b) \preceq L(b)\{L(a) - L(c) + 1\}, \tag{22}$$

from which Theorem 3 is immediate. ∎

We now proceed to prove that $t_E^+(m,n,k) \sim n(m-k+1)$, for which purpose we need the following two theorems.

THEOREM 4. $t_E(a,b) \succeq D(a,b)\{D(a,b) + L(\text{g.c.d.}(a,b))\}$.

Proof. Let $(q_1, \cdots, q_l)$ and $(a_1, \cdots, a_{l+2})$ be the quotient and remainder sequences of (a,b), $c = \text{g.c.d.}(a,b)$ and $k = L(c)$. By (17),

$$t_E(a,b) \succeq \sum_{i=1}^{l} L(a_i). \tag{23}$$

Since $a_{l+2} = 0$, $a_{l+1} = c$ and $a_i = q_i a_{i+1} + a_{i+2} \geqq a_{i+1} + a_{i+2}$, a simple induction shows that $a_{l+2-i} \geqq cF_i$, where F_i is the ith term of the Fibonacci sequence defined by $F_0 = 0$, $F_1 = 1$ and $F_{i+2} = F_i + F_{i+1}$. But [10, p. 82] $F_{i+1} \geqq \phi^i/\sqrt{5}$, where $\phi = (1 + \sqrt{5})/2$, and $\phi^2 > \sqrt{5}$ so $F_{i+3} > \phi^i$. Hence

$$\sum_{i=1}^{l} L(a_i) \geqq \sum_{i=2}^{l+1} \log_\beta (cF_i) \geqq l(\log_\beta c) + \sum_{i=1}^{l-2} \log_\beta \phi^i \geqq l(\log_\beta c) + \binom{l-2}{2}(\log_\beta \phi).$$

So for $k \geqq 2$ and $l \geqq 4$,

$$\sum_{i=1}^{l} L(a_i) \geqq \frac{1}{2}kl + \frac{1}{16}(\log_\beta \phi)l^2 \succeq kl + l^2,$$

while for $k = 1$ and $l \geqq 4$,

$$\sum_{i=1}^{l} L(a_i) \geqq \frac{1}{16}(\log_\beta \phi)l^2 \succeq l^2 \sim kl + l^2.$$

For $l \leqq 3$, $\sum_{i=1}^{l} L(a_i) \geqq L(c) = k \sim kl + l^2$. So by Theorem 1, part (h), $\sum_{i=1}^{l} L(a_i) \geqq kl + l^2$ for all k and l, proving the theorem, since $l = D(a, b)$. ∎

By an application of Theorem 4, together with an elementary construction utilizing the generalized Fibonacci sequences $F^{(h)}$ defined by $F_0^{(h)} = 1$, $F_1^{(h)} = h$ and $F_{i+2}^{(h)} = F_i^{(h)} + F_{i+1}^{(h)}$, one can obtain a proof that $t_E^+(m, m, k) \succeq n(m - k + 1)$. However, we will abstain from this construction, obtaining the result instead as a corollary of our analysis, in § 5, of the average computing time. Hence we proceed next to derive the minimum computing time of the Euclidean algorithm.

THEOREM 5. $t_E^-(m, n, k) \sim n(m - n + 1) + k(n - k + 1)$.

Proof. By (17), $t_E^-(m, n, k) \succeq L(q_1)L(a_2) \sim n(m - n + 1)$. Since $q_i = \lfloor a_i/a_{i+1} \rfloor$, we have $q_{i+1} > a_i/a_{i+1}$ and so $\prod_{i=1}^{l}(q_i + 1) > \prod_{i=1}^{l}(a_i/a_{i+1}) = a/c$. By (17),

$$t_E(a, b) \sim \sum_{i=1}^{l} L(q_i)L(a_{i+1}) \geqq L(c) \sum_{i=1}^{l} L(q_i) \sim L(c) \sum_{i=1}^{l} L(q_i + 1)$$

$$\geqq L(c)L(a/c) \geqq L(c)L(b/c) \sim L(c)\{L(b) - L(c) + 1\}.$$

Hence $t_E^-(m, n, k) \succeq k(n - k + 1)$ and by Theorem 1, part (c), $t_E^-(m, n, k) \succeq n(m - n + 1) + k(n - k + 1)$.

If $n = k$, let $a = \beta^{m-1}$ and $b = \beta^{n-1}$ so that $c = \beta^{n-1}$ and $D(a, b) = 1$. By (17), this shows that $t_E^-(m, n, k) \preceq n(m - n + 1) \preceq n(m - n + 1) + k(n - k + 1)$.

If $n > k$, let $a = \beta^{m-1} + \beta^{k-1}$ and $b = \beta^{n-1}$, so that $c = \beta^{k-1}$, $L(a) = m$ and $D(a, b) = 2$. Then by (17), $t_E^-(m, n, k) \preceq n(m - n + 1) + k(n - k + 1)$ for $n > k$. Application of Theorem 1, part (h), concludes the proof. ∎

5. The average computing time. As observed in the proof of Theorem 4, if $a \geqq b$ and $(a_1, a_2, \cdots, a_{l+1}, a_{l+2})$ is the remainder sequence of (a, b), then $a \geqq F_{l+1} \geqq \phi^l/\sqrt{5}$. Since $e > \sqrt{5}$, we have $l \ln \phi \geqq (\ln a) + 1$. That is,

$$D(a, b) \leqq (\ln \phi)^{-1}((\ln a) + 1), \tag{24}$$

with $(\ln \phi)^{-1} = 2.078 \cdots$. Dixon established in [6] that for every $\varepsilon > 0$

$$|D(a, b) - \tau \ln a| < (\ln a)^{1/2+\varepsilon} \tag{25}$$

for almost all pairs (a, b) with $u \geqq a \geqq b \geqq 1$, as $u \to \infty$, where

$$\tau = 12\pi^{-2} \ln 2, \tag{26}$$

and we have $\tau = 0.84276 \cdots$. By more elementary means, Dixon proved in [7] the weaker result that

$$D(a, b) \geqq \tfrac{1}{2} \ln a \tag{27}$$

for almost all pairs (a, b) with $u \geqq a \geqq b \geqq 1$ as $u \to \infty$. In the following, we will show how Dixon's weaker result can be used to prove that the average computing time of the Euclidean algorithm is codominant with its maximum computing time of $n(m - k + 1)$. Before proceeding to the detailed proof, however, we shall present an intuitive sketch.

It is a well-known result (see [11, § 4.5.2, Exercise 10]) that the proportion of pairs (a, b) with $u > a \geqq b \geqq 1$ for which g.c.d.$(a, b) = 1$ approaches $6\pi^{-2}$ as $u \to \infty$. We will first generalize this result to the pairs (a, b) with $u > a \geqq b \geqq v$ as $u - v \to \infty$. Next we set $u = \beta^{n-k+1/2}$ and $v = \beta^{n-k}$ and conclude, combining this result with Dixon's, that, for $n - k$ large, at least half of the pairs (a, b) for which $u > a \geqq b \geqq v$ satisfy both g.c.d.$(a, b) = 1$ and $D(a, b) \geqq \frac{1}{2} \ln a$. For each pair satisfying these conditions and each c with $\beta^{k-1} \leqq c < \beta^{k-1/2}$, we obtain a pair $(\bar{a}, \bar{b}) = (ac, bc)$ with g.c.d.$(\bar{a}, \bar{b}) = c$, $L(\bar{a}) = L(\bar{b}) = n$ and $L(c) = k$. If $m > n$, then from each pair $(\bar{a}, \bar{b})$ we obtain at least $\frac{1}{2}\beta^{m-n}$ pairs $(\bar{\bar{a}}, \bar{\bar{b}})$ of the form $(\bar{a}q + \bar{b}, \bar{b})$ for which $L(\bar{\bar{a}}) = m$ and these also satisfy $L(\bar{\bar{b}}) = n$, $L(\text{g.c.d.}(\bar{\bar{a}}, \bar{\bar{b}})) = k$ and $D(\bar{\bar{a}}, \bar{\bar{b}}) \geqq \frac{1}{2} \ln \beta^{n-k}$. The pairs $(\bar{\bar{a}}, \bar{\bar{b}})$ so obtained constitute at least $0.004\beta^{-2}$ of all pairs in $S_{m,n,k}$ and $t_E(\bar{\bar{a}}, \bar{\bar{b}}) \geq n(m - k + 1)$ for all $(\bar{\bar{a}}, \bar{\bar{b}})$, so $t_E^*(m, n, k) \geq n(m - k + 1)$ for $n - k > h$, say. But it is trivial that $t_E^*(m, n, k) \geq n(m - k + 1)$ for $n - k \leqq h$ for any constant h, and so $t_E^*(m, n, k) \sim n(m - k + 1)$.

THEOREM 6. *Let u and v be positive integers with $u > v$, let $w = u - v$, and let q be the number of pairs of integers (a, b) such that $u > a, b \geqq v$ and* g.c.d.$(a, b) = 1$. *Then* $|q/w^2 - 6/\pi^2| \leqq 2((\ln u) + 1)/w + u/w^2 + 1/u$.

Proof. Let v_k be the number of integers a such that $k|a$ and $u > a \geqq v$. Then

$$|v_k - w/k| < 1, \tag{28}$$

and v_k^2 is the number of pairs (a, b) for which $k|$g.c.d.(a, b) and $u > a, b \geqq v$. By the principle of inclusion and exclusion,

$$q = \sum_{k=1}^{u} \mu(k) v_k^2, \tag{29}$$

where μ is the Möbius function. By (28),

$$|v_k^2 - w^2/k^2| < 2w/k + 1. \tag{30}$$

Multiplying (30) by $\mu(k)/w^2$ and summing, we have, by (29),

$$|q/w^2 - \sum_{k=1}^{u} \mu(k)/k^2| < 2H_u/w + u/w^2, \tag{31}$$

where H_u is the harmonic sum $\sum_{k=1}^{u} 1/k$. Using

$$\sum_{k=1}^{\infty} \mu(k)/k^2 = \pi^2/6 \tag{32}$$

together with (31) yields

$$|q/w^2 - \pi^2/6| < 2H_u/w + u/w^2 + \sum_{k=u+1}^{\infty} 1/k^2. \tag{33}$$

But $\sum_{k=u+1}^{\infty} 1/k^2 < \int_u^{\infty} x^{-2}\,dx = 1/u$, and $H_u \leqq (\ln u) + 1$, which establishes the theorem after substitution in (33). ∎

THEOREM 7. *There is a positive integer h such that for $n - k > h$, there are at least $0.02\beta^{2n-2k+1}$ pairs (a, b) for which $\beta^{n-k+1/2} > a \geqq b \geqq \beta^{n-k}$, g.c.d.$(a, b) = 1$ and $D(a, b) \geqq \frac{1}{2} \ln a$.*

Proof. Set $u = \lceil \beta^{n-k+1/2} \rceil$, $v = \beta^{n-k}$, $w = u - v$. Since $\lim_{n-k\to\infty} (u/w) = (1 - \beta^{-1/2})^{-1} \leqq (1 - 1/\sqrt{2})^{-1} < 4$, it follows from Theorem 6 that

$$\lim_{n-k\to\infty} (q/w^2) = 6/\pi^2.$$

Since $6/\pi^2 > 0.6$ and g.c.d.(a, b) = g.c.d.(b, a) there exists an h_1 such that for $n - k > h_1$, there are at least $0.3w^2$ pairs (a, b) for which $u > a \geqq b \geqq v$ and g.c.d.$(a, b) = 1$. By Dixon's theorem there is an h_2 such that if $n - k > h_2$, then $D(a, b) < \frac{1}{2} \ln a$ for at most 0.05 pairs (a, b) with $u \geqq a$, $b \geqq 1$. Hence if $h = \max(h_1, h_2)$ and $n - k > h$, there are at most $\frac{1}{4}w^2$ pairs (a, b) for which $u > a \geqq b \geqq v$, g.c.d.$(a, b) = 1$ and $D(a, b) \geqq \frac{1}{2} \ln a$. The theorem follows since $w \geqq (\sqrt{\beta} - 1)\beta^{n-k}$ and $(\sqrt{\beta} - 1)^2/\beta \geqq (\sqrt{2} - 1)^2/2 \geqq 0.08$. ∎

THEOREM 8. *There is a positive integer h such that for $n - k > h$, there are at least $0.004\beta^{m+n-k}$ pairs (a, b) such that $a \geqq b$, $L(a) = m$, $L(b) = n$, L(g.c.d.$(a, b)) = k$ and $D(a, b) \geqq \frac{1}{2} \ln \beta^{n-k}$.*

Proof. Choose an h for which Theorem 7 holds. For every pair (a, b) satisfying Theorem 7 and every integer satisfying $\beta^{k-1} \leqq c < \beta^{k-1/2}$, we obtain a pair (ac, bc) with $ac \geqq bc$, $L(ac) = L(bc) = n$, L(g.c.d.$(ac, bc)) = L(c) = k$, and $D(ac, bc) = D(a, b) \geqq \frac{1}{2} \ln a \geqq \frac{1}{2} \ln \beta^{n-k}$. The mapping $f((a, b), c) = (ac, bc)$ thus defined is one-to-one so there are at least

$$(0.02\beta^{2n-2k+1})(\sqrt{\beta} - 1)\beta^{k-1} \geqq 0.008\beta^{2n-k+1}$$

pairs (a, b) with $a \geqq b$, $L(a) = L(b) = n$, L(g.c.d.$(a, b)) = k$ and $D(a, b) \geqq \frac{1}{2} \ln \beta^{n-k}$. If $m = n$, this completes the proof; so assume $m > n$. For each pair (a, b) with $L(a) = L(b) = n$, there are at least

$$\lfloor (\beta^m - \beta^{m-1})/a \rfloor \geqq (\beta^m - \beta^{m-1})/\beta^n \geqq (1 - \beta^{-1})\beta^{m-n-1} \geqq \tfrac{1}{2}\beta^{m-n}$$

pairs $(aq + b, a)$ with $L(aq + b) = m$. Since g.c.d.$(aq + b, a)$ = g.c.d.(a, b) and $D(aq + b, a) = D(a, b) + 1$, we obtain at least $(0.008\beta^{2n-k})(\frac{1}{2}\beta^{m-n}) = 0.004\beta^{m+n-k}$ pairs $(aq + b, a)$ for which $aq + b \geqq a$, $L(aq + b) = m$, $L(a) = n$, L(g.c.d.$(qa + b, a)) = k$ and $D(aq + b, a) \geqq \frac{1}{2} \ln \beta^{n-k}$. ∎

THEOREM 9. $t_E^*(m, n, k) \sim t_E^+(m, n, k) \sim n(m - k + 1)$.

Proof. Let $c_1 = \min(1, \frac{1}{2}\ln\beta)$. By Theorems 4 and 8, there exist h and $c_2 > 0$ such that

$$t_E(a, b) \geqq c_2 D(a, b)\{D(a, b) + L(\text{g.c.d.}(a, b))\} \geqq c_2 c_1 (n - k)\{c_1(n - k) + k\}$$
$$\geqq c_1^2 c_2 n(n - k)$$

for $n - k > h$ and for at least $0.004\beta^{m+n-k}$ elements of $S_{m,n,k}$. Every element of $S_{m,n,k}$ is of the form (ac, bc) with $a < \beta^{m-k+1}$, $b < \beta^{n-k+1}$ and $c < \beta^k$, so $S_{m,n,k}$ has at most $\beta^{m+n-k+2}$ elements. Hence, $t_E^*(m, n, k) \geqq 0.004 c_1^2 c_2 \beta^{-2} n(n - k) \sim n(n - k)$ for $n - k > h$. By Theorem 5, $t_E^*(m, n, k) \succeq n(m - n + 1) \geqq n \succeq n(n - k)$ for $n - k \leqq h$. Hence by Theorem 1, part (h), $t_E^*(m, n, k) \succeq n(n - k)$. By Theorem 5, $t_E^*(m, n, k) \succeq n(m - n + 1)$ so by Theorem 1, part (c),

$$t_E^*(m, n, k) \succeq n(n - k) + n(m - n + 1) = n(m - k + 1). \tag{34}$$

The conclusion of the theorem is now immediate from Theorem 3, (34) and the obvious inequality $t^*(m, n, k) \leqq t_E^*(m, n, k)$.

REFERENCES

[1] W. S. BROWN, *On Euclid's algorithm and the computation of polynomial greatest common divisors*, J. Assoc. Comput. Mach., 18 (1971), pp. 478–504.

[2] G. E. COLLINS, *Computing time analyses for some arithmetic and algebraic algorithms*, Proc. 1968 Summer Institute on Symbolic Mathematical Computation, IBM Corp., Cambridge, Mass., 1969, pp. 197–231.

[3] ———, *The calculation of multivariate polynomial resultants*, J. Assoc. Comput. Mach., 18 (1971), pp. 515–532.

[4] ———, *The SAC-1 integer arithmetic system—Version III*, Tech. Rep. 156, Computer Sciences Dept., Univ. of Wisconsin, Madison, 1973.

[5] N. G. DEBRUIJN, *Asymptotic Methods in Analysis*, North-Holland, Amsterdam, 1961.

[6] J. D. DIXON, *The number of steps in the Euclidean algorithm*, J. Number Theory, 2 (1970), pp. 414–422.

[7] ———, *A simple estimate for the number of steps in the Euclidean algorithm*, Amer. Math. Monthly, 78 (1971), pp. 374–376.

[8] H. HEILBRONN, *On the average length of a class of continued fractions*, Abhandlungen aus Zahlentheorie und Analysis, VEB Deutscher Verlag, Berlin, 1968.

[9] L. E. HEINDEL, *Integer arithmetic algorithms for polynomial real zero determination*, J. Assoc. Comput. Mach., 18 (1971), pp. 533–548.

[10] D. E. KNUTH, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Addison-Wesley, Reading, Mass., 1968.

[11] ———, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969.

[12] D. R. MUSSER, *Algorithms for Polynomial Factorization*, Ph.D. thesis, Tech. Rep. 134, Computer Sciences Dept., Univ. of Wisconsin, Madison, 1971.