STANFORD ARTIFICAL INTELLIGENCE LABORATORY
MEMO AIM - 187

STAN-CS-73-331

THE COMPUTING TIME OF THE EUCLIDEAN ALGORITHM

BY

GEORGE E. COLLINS

SUPPORTED BY

NATIONAL SCIENCE FOUNDATION
GRANT GJ-30125 X
AND
ADVANCED RESEARCH PROJECTS AGENCY

ARPA ORDER NO. 457

JANUARY 1973

COMPUTER SCIENCE DEPARTMENT

School of Humanities and Sciences

STANFORD UNIVERSITY

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY January, 1973
MEMO AIM-l@

COMPUTER SCIENCE DEPARTMENT
REPORT NO. CS-331

THE COMPUTING TIME OF THE EUCLIDEAN ALGORITHM

by

George E. Collins?

ABSTRACT: The maximum, minimum and average computing times of the classical Euclidean algorithm for the greatest common divisor of two integers are derived, to within codominance, as functions of the lengths of the two inputs and the output.

+On leave from the University of Wisconsin-Madison.

This research is supported by NSF grant GJ-30125X, the Wisconsin Alumni Research Foundation, and (in part) by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183).

1. Introduction

Knuth, [11], Dixon, [6] and [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 factor 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 ail 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], Section 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 $f(x) \leq c \cdot g(x)$ for all x S. We may 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.

Knuth ([lo], pp. 104-108) defines $f(x)=O(g(x))$ in case there is a positive constant c such that $|f(x)| \leq c \cdot |g(x)|$. As long as one is dealing only with non-negative valued functions, this formally coincides with the definition above 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 $S \subseteq S_1 \times \ldots \times S_n$, a Cartesian product, we will denote by f_a the function f restricted to $(\{a\} \cap S_2 \times \ldots \times S_n) \cap S$; that is, $f_a(x_2,\ldots,x_n)=f(a,x_2,\ldots,x_n)$ for $(a,x_2,\ldots,x_n) \in S$. 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 non-negative 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 $f_1 \cdot f_2 \preceq g_1 \cdot g_2$.

(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 $f+g \preceq f \cdot g$.

(f) If $1 \preceq f$, then $f \sim f+c$.

(g) Let $S \subseteq S_1 \times \ldots \times S_n$ and $a \in S_1$. If $f \preceq g$, then $f_a \preceq g_a$.

(h) Let $S=S_1 \cup S_2$. If $f \preceq g$ on S_1 and $f \preceq g$ on S_2, then $f \preceq g$ on S.

Proof. These properties follow immediately from the definition, except for (e). To prove (e), assume $1 \preceq f$ and $1 \preceq g$ so that, for some positive real number c, $cf \geq 2$ and $cg \geq 2$. We then have $(cf-2)(cg-2) \geq 0$, so $c^2fg+4 \geq 2c(f+g) \geq c(f+g)+4$. Hence $c^2fg \geq c(f+g)$, $cfg \geq f+g$ and $f+g \preceq fg$. I

3. Computing Time Functions

Let A be any algorithm and let S be the set of all valid inputs to A (the elements of S may be n-tuples). We associate with A a computing time function t_A defined on S, $t_A(x)$ being the number of basic operations performed by the algorithm A when presented with the input x, a positive integer. 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 $S = \cup_{n=1}^{\infty} S_n$, where each S_n is a non-empty 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 $\mathcal{S} = \{S_1, S_2, S_3, \ldots\}$ of S we define maximum, minimum and average computing time functions, t_A^+, t_A^- and t_A^* on $\mathcal{S}$ as follows, where $|S_n|$ denotes the number of elements of S_n.

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

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

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

As 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 $\beta\geq 2$, as discussed by Knuth in [11], Section 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 B-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, we have

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

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

$$L_\beta(a)\sim \ell n(|a|+2) \text{ on } R, \qquad (6$$

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:

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

I the set of integers,

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

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

We will also need the following theorem.

Theorem 2. (a) $L(\prod_{i=1}^{n} a_i) \preceq \sum_{i=1}^{n} L(a_i)$ for $a_1,\ldots,a_n \in I$. (b) $L(\prod_{i=1}^{n} a_i) \sim \sum_{i=1}^{n} L(a_i)$ for $a_1,\ldots,a_n \in I-\{-1,0,1\}$.

Proof. $L(ab) \leq L(a)+L(b)$ for $a,b \in I$, so $L(\prod_{i=1}^{n} a_i) \leq \sum_{i=1}^{n} L(a_i)$ by induction on n, proving (a). To prove (b), assume first that $2 \leq |a_i| < \beta$ for $1 \leq i \leq n$. Then $L(\prod_{i=1}^{n} a_i) \geq \log_\beta \prod_{i=1}^{n} |a_i| = (\log_\beta 2) \log_2 \prod_{i=1}^{n} |a_i| \geq (\log_\beta 2) \log_2 2^n = (\log_\beta 2) n = (\log_\beta 2) \sum_{i=1}^{n} L(a_i)$, so $\sum_{i=1}^{n} L(a_i) \leq (\log_2 \beta) L(\prod_{i=1}^{n} a_i)$.

Next, assume $L_\beta(a_i) \geq 2$ for $1 \leq i \leq n$, and let $\ell_i = L_\beta(a_i)$. Then $L(\prod_{i=1}^{n} a_i) \geq \log_\beta (\prod_{i=1}^{n} |a_i|) \geq \log_\beta (\prod_{i=1}^{n} \beta^{\ell_i - 1}) = \sum_{i=1}^{n} (\ell_i - 1) \geq \sum_{i=1}^{n} \ell_i/2$, so $\sum_{i=1}^{n} L(a_i) \leq 2L(\prod_{i=1}^{n} a_i)$.

Combining these two cases, we may assume $L(a_i)=1$ for $1 \leq i \leq m$ and $L(a_i) \geq 2$ for $m+1 \leq i \leq n$. Then $\sum_{i=1}^{n} L(a_i) \leq (\log_2 \beta) L(\prod_{i=1}^{m} a_i) + 2L(\prod_{i=m+1}^{n} a_i) \leq 2(\log_2 \beta) \{L(\prod_{i=1}^{m} a_i) + L(\prod_{i=m+1}^{n} a_i)\} \leq 4(\log_2 \beta) L(\prod_{i=1}^{n} a_i)$ since $L(a)+L(b) \leq 2L(ab)$ for $a,b \in I-\{0\}$. ▮

It should be noticed that a simple inductive proof of (b) was not possible because n is regarded as a variable, not as an arbitrary but fixed positive integer. As an immediate corollary of Theorem 2, we have

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

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

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

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

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

Thus, for these algorithms, the natural decomposition of the set $\mathcal{S}=\{(a,b):a,b \in I\}$ consists of the sets $S_{m,n}=\{(a,b):L(a)=m \& L(b)=n\}$. 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

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

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

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

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 $a \geq b$. The output of the algorithm is the positive integer $c = \gcd(a,b)$.

Algorithm E

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

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

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

(4) Return.

This algorithm computes two sequences, $(a_1, a_2, \ldots, a_{\ell+2})$ and $(q_1, q_2, \ldots, q_\ell)$ such that $a_1=a$, $a_2=b$, $a_i = q_i a_{i+1} + a_{i+2}$ with $0 \leq a_{i+2} < a_{i+1}$ for $1 \leq i \leq \ell$, and $a_{\ell+2}=0$. $a_1, \ldots, a_{\ell+1}$ are the successive values assumed by the variable c and $q_1, \ldots, q_\ell$ are the successive values assumed by the variable q. $(a_1, \ldots, a_{\ell+2})$ is called the remainder sequence of (a,b) and $(q_1, \ldots, q_\ell)$ 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 i^{th} execution of step (2) is $\sim L(q_i)L(a_{i+1})$. The computing time for the i^{th} execution of step (3) is certainly dominated by $L(a_{i+1})$ since at most it 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

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

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 &into the sets

$$S_{m,n,k}=\{(a,b):L(a)=m \& L(b)=n \& L(\gcd(a,b))=k\}, \qquad (18)$$

with $m \geq n \geq k \geq 1$. We may verify that each set $S_{m,n,k}$ is non-empty as follows. If m=k, then $(\beta^{m-1},\beta^{m-1}) \epsilon S_{m,n,k}$. If m>k, let $a=\beta^{m-1}+\beta^{k-1}$ and $b=\beta^{n-1}$. Then $c=\gcd(a,b)=\beta^{k-1}$, $L(a)=m$, $L(b)=n$ and $L(c)=k$, so $(a,b) \epsilon S_{m,n,k}$. 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^*.

<u>Theorem 3</u>. $t_E^+(m,n,k) \preceq n(m-k+1)$.

<u>Proof</u>. Since $b=a_2>a_3>\ldots>a_{\ell+1}$, we have by (17) that

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

Since $L(a)\sim L(a+1)$ for $a \geq 1$ and since $q_\ell \geq 2$ we obtain, by Theorem 2,

$$\sum_{i=1}^{\ell} L(q_i) \sim L(q_\ell \Pi_{i=1}^{\ell-1}(q_i+1)). \qquad (20)$$

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 $i<\ell$ and hence $\Pi_{i=1}^{\ell-1}(q_i+1) < a_1 a_2/a_\ell a_{\ell+1}$. Combining this with $q_\ell = a_\ell/a_{\ell+1}$ yields

$$q_\ell \Pi_{i=1}^{\ell-1}(q_i+1) \leq ab/c^2. \qquad (21)$$

Since $L(ab/c^2) \leq L(a^2/c^2) \sim L(a/c) \sim L(a)-L(c)+1$, (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.

<u>Theorem 4</u>. $t_E(a,b) \succeq D(a,b)\{D(a,b)+L(\gcd(a,b))\}$.

<u>Proof</u>. Let $(q_1,\ldots,q_\ell)$ and $(a_1,\ldots,a_{\ell+2})$ be the quotient and remainder sequences of (a,b), $c=\gcd(a,b)$ and $k=L(c)$. By (17),

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

Since $a_{\ell+2}=0$, $a_{\ell+1}=c$ and $a_i=q_i a_{i+1}+a_{i+2} \geq a_{i+1}+a_{i+2}$, a simple induction shows that $a_{\ell+2-i} \geq cF_i$, where F_i is the i^{th} 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) $F_{i+1} \geq \phi^i/\sqrt{5}$, where $\phi=(1+\sqrt{5})/2$, and $\phi^2>\sqrt{5}$ so $F_{i+3} > \phi^i$. Hence $\sum_{i=1}^{\ell} L(a_i) \geq \sum_{i=2}^{\ell+1} \log_\beta(cF_i) \geq \ell(\log_\beta c)+\sum_{i=1}^{\ell-2} \log_\beta \phi^i \geq \ell(\log_\beta c)+\binom{\ell-2}{2}(\log_\beta \phi)$. So for $k \geq 2$ and $\ell \geq 4$, $\sum_{i=1}^{\ell} L(a_i) \geq \frac{1}{2}k\ell+(1/16)(\log_\beta \phi)\ell^2 \succeq k\ell+\ell^2$ while for $k=1$ and $\ell \geq 4$, $\sum_{i=1}^{\ell} L(a_i) \geq (1/16)(\log_\beta \phi)\ell^2 \succeq \ell^2 \sim k\ell+\ell^2$. For $\ell \leq 3$, $\sum_{i=1}^{\ell} L(a_i) \geq L(c)=k \sim k\ell+\ell^2$. So by Theorem 1, part (h), $\sum_{i=1}^{\ell} \succeq k\ell+\ell^2$ for all k and ℓ, proving the theorem, since $\ell=D(a,b)$. ∎

<u>Theorem 5</u>. For every positive integer n, there exist positive integers e and f with $e \geq f$, $L(e)=L(f)=n$, $\gcd(e,f)=1$, and $D(e,f)=n$.

<u>Proof</u>. Let $F^{(h)}$ be the generalized Fibonacci sequence defined by $F_0^{(h)}=1$,

$F_1^{(h)}=h$, and $F_{i+2}^{(h)}=F_i^{(h)}+F_{i+1}^{(h)}$ for $i \geq 0$. If $e=F_{n+1}^{(h)}$ and $f=F_n^{(h)}$ then $e \geq f > 0$ and $F_{n+1}^{(h)}, F_n^{(h)}, \ldots, F_1^{(h)}=h, F_o^{(h)}=1, 0$ is the remainder sequence of (e,f) so $\gcd(e,f)=1$ and $D(e,f)=n$. Hence it suffices to show that for every $n \geq 1$ there is an $h \geq 1$ such that $\beta^{n-1} \leq F_n^{(h)} \leq F_{n+1}^{(h)} < \beta$. It can be verified by calculation that for $n \leq 6$ this holds with $h=n$.

Since $F_n^{(h)}=F_{n-1}+hF_n$ for $n \geq 1$ (see [10], Section 1.2.8, Exercise 13) and $F_n=(\phi^n-\hat{\phi}^n)/\sqrt{5}$ where $\hat{\phi}=-\phi^{-1}=\frac{1}{2}(1-\sqrt{5})$ for $n \geq 0$ (see [10], Section 1.2.8, Formula (14)), we have $|F_n-\phi^n\sqrt{5}|=|\hat{\phi}^n/\sqrt{5}|=(|\hat{\phi}/\phi|^n/\sqrt{5})\phi^n$. But $|\hat{\phi}/\phi|^5 < .009$, so $|F_n-\phi^n/\sqrt{5}| < .005\,\phi^n$ for $n \geq 5$ and hence $F_n/F_{n-1} < 1.005\,\phi^n/.995\,\phi^{n-1} < 1.011\phi < 1.64$ for $n \geq 6$.

Assume as induction hypothesis that $\beta^{n-1} \leq F_n^{(h)} < F_{n+1}^{(h)} < \beta^n$ with $h \geq n \geq 6$. Let k be the least positive integer for which $\beta^n \leq F_{n+1}^{(k)}$. Then $k > h$ and $F_{n+1}^{(k)}/F_{n+1}^{(k-1)}=\{F_n+kF_{n+1}\}/\{F_n+(k-1)F_{n+1}\} < k/(k-1) < 7/6$, so $F_{n+1}^{(k)} < (7/6)F_{n+1}^{(k-1)} < (7/6)\beta^n$. Also, $F_{n+2}^{(k)}/F_{n+1}^{(k)}=\{F_{n+1}+kF_{n+2}\}/\{F_n+kF_{n+1}\} \leq \max\{F_{n+1}/F_n, F_{n+2}/F_{n+1}\} < 1.64$, so $F_{n+2}^{(k)} < 1.64F_{n+1}^{(k)} < (7/6)(1.64)\beta^n < 2\beta^n \leq \beta^{n+1}$. Hence $\beta^n \leq F_{n+1}^{(k)} < F_{n+2}^{(k)} < \beta^{n+1}$ and $k \geq h+1 \geq n+1$, completing the induction. ∎

<u>Theorem 6</u>. $t_E^+(m,n,k) \sim n(m-k+1)$.

<u>Proof</u>. By Theorem 3, it suffices to prove that $t_E^+(m,n,k) \succeq n(m-k+1)$. Using Theorem 5, choose e and f with $e \geq f > 0$, $L(e)=L(f)=n-k+1$, $\gcd(e,f)=1$ and $D(e,f)=n-k+1$. Let $\bar{b}=f$ and $\bar{a}=e+qf$ where q is the least non-negative integer such that $e+qf \geq \beta^{m-k}$. If $q=0$ then $\bar{a}=e$, $m=n$ and $L(\bar{a})=m-k+1$. If $q=1$, then $m > n$ so $\bar{a}=e+f \leq 2e < 2\beta^{n-k+1} \leq \beta^{n-k+2} \leq \beta^{m-k+1}$ and $L(\bar{a})=m-k+1$. If $q \geq 2$ then $\bar{a}=e+qf \leq 2e+(q-1)f < 2(e+(q-1)f) < 2\beta^{m-k} \leq \beta^{m-k+1}$ and $L(\bar{a})=m-k+1$. Also, $\gcd(\bar{a},\bar{b})=\gcd(f,e+qf)=\gcd(e,f)=1$ and $D(\bar{a},\bar{b})=D(e,f)=n-k+1$.

Let $c=\beta^{k-1}$, $a=\bar{a}c$ and $b=\bar{b}c$. Then $c=\gcd(a,b)$, $L(c)=k$, $L(a)=m$, $L(b)=n$ and $D(a,b)=n-k+1$. Hence by Theorem 4 $t_E^+(m,n,k) \succeq (n-k+1)\{(n-k+1)+k\} \sim n(n-k+1)$.

Also, by (17), $t_E^+(m,n,k) \succeq L(q_1)L(a_2) \sim (m-n+1)(n)$. So by Theorem 1, part (c), $t_E^+(m,n,k) \succeq n(n-k+1)+n(m-n+1) \sim n(m-k+1)$. ∎

In the next theorem we obtain the minimum computing time of the Euclidean algorithm, which is much easier.

Theorem 7. $t_E^-(m,n,k) \sim n(m-n+1)+k(n-k+1)$.

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 $\prod_{i=1}^{\ell}(q_i+1) > \prod_{i=1}^{\ell}(a_i/a_{i+1}) = a/c$. BY (17), $t_E(a,b) \sim \sum_{i=1}^{\ell} L(q_i)L(a_{i+1}) \geq L(c)\sum_{i=1}^{\ell} L(q_i) \sim L(c)\sum_{i=1}^{\ell} L(q_i+1) \geq L(c)L(a/c) \geq L(c)L(b/c) \sim L(c)\{L(b)-L(c)+1\}$. 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 $a \geq b$ and $(a_1, a_2, \ldots, a_{\ell+1}, a_{\ell+2})$ is the remainder sequence of (a,b), then $a \geq F_{\ell+1} \geq \phi^{\ell}/\sqrt{5}$. Since $e > \sqrt{5}$, we have $\ell \ln\phi \geq \ln a + 1$. That is,

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

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

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

for almost all pairs (a,b) with $u \geq a \geq b \geq 1$, 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

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

for almost all pairs (a,b) with $u \geq a \geq b \geq 1$ 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, I shall present an intuitive sketch.

It is a well-known result (see [11], Section 4.5.2, Excercise 10) that the proportion of pairs (a,b) with $u > a \geq b \geq 1$ for which $\gcd(a,b)=1$ approaches $6\pi^{-2}$ as $u \to \infty$. We will first generalize this result to the pairs (a,b) with $u > a \geq b \geq v$ as $u - v \to \infty$. Next we set $u = \beta^{n-k+\frac{1}{2}}$ 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 $u > a \geq b \geq v$ satisfy both $\gcd(a,b)=1$ and $D(a,b) \geq \frac{1}{2}\ln a$. For each pair satisfying these conditions and each c with $\beta^{k-1} \leq c < \beta^{k-\frac{1}{2}}$ we obtain a pair $(\bar{a},\bar{b})=(ac,bc)$ with $\gcd(\bar{a},\bar{b})=c$, $L(\bar{a})=L(\bar{b})=n$ and $L(c)=k$. If $m > n$ than 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 $L(\bar{\bar{b}})=n$, $L(\gcd(\bar{\bar{a}},\bar{\bar{b}}))=k$ and $D(\bar{\bar{a}},\bar{\bar{b}}) \geq \frac{1}{2}\ln \beta^{n-k}$. The pairs $(\bar{\bar{a}},\bar{\bar{b}})$ so obtained constitute at least $.004\beta^{-2}$ of all pairs in $S_{m,n,k}$ and $t_E(\bar{\bar{a}},\bar{\bar{b}}) \succeq n(m-k+1)$ for all $(\bar{\bar{a}},\bar{\bar{b}})$, so $t_E^*(m,n,k) \succeq n(m-k+1)$ for $n-k>h$, say. But it is trivial that $t_E^*(m,n,k) \succeq n(m-k+1)$ for $n-k\leq h$ for any constant h, and so $t_E^*(m,n,k) \sim n(m-k+1)$.

Theorem 8. 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 $u>a,b \geq v$ and $\gcd(a,b)=1$. Then $|q/w^2-6/\pi^2|\leq (2\ln w+4)/w$.

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

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

and ν_k^2 is the number of pairs (a,b) for which $k|\gcd(a,b)$ and $u>a, b\geq v$. By the principle of inclusion and exclusion,

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

where μ is the Mobius function. BY (28),

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

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

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

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

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

together with (31) yields

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

But $\sum_{k=w+1}^{\infty} 1/k^2 < \int_w^{\infty} x^{-2}dx$ and $H_w \leq \ln w+1$, which establishes the theorem after substitution in (33). ▮

Theorem 9. There is a positive integer h such that for n-k%, there are at least $0.02\beta^{2n-2k+1}$ pairs (a,b) for which $\beta^{n-k+*} > a \geq b \geq \beta^{n-k}$, gcd(a,b)=1, and $D(a,b) \geq \frac{1}{2}\ln a$.

Proof. Set $u=\lceil \beta^{n-k+\frac{1}{2}} \rceil$, $v=\beta^{n-k}$, $w=u-v$. Since $6/\pi^2 > 0.6$, $\lim_{w\to\infty} (2\ln w+4)/w=0$, and gcd(a,b)=gcd(b,a), by Theorem 8 there exists h_1 such that there are at least $0.3\, w^2$ pairs (a,b) for which $u > a \geq b \geq v$ and gcd(a,b)=1, for n-k > hl. 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 $u \geq a$, $b \geq 1$. Hence if $h=\max(h_1,h_2)$ and n-k > h there are at most $(1/4)w^2$ pairs (a,b) for which $u > a \geq b \geq v$, gcd(a,b)=1 and $D(a,b) \geq \frac{1}{2}\ln a$. The theorem follows since $w \geq (\sqrt{\beta}-1)\beta^{n-k}$ and $(\sqrt{\beta}-1)^2/\beta \geq (\sqrt{2}-1)^2/2 \geq 0.08$. I

Theorem 10. 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 $a \geq b$, L(a)=m, L(b)=n, L(gcd(a,b))=k and $D(a,b) \geq \frac{1}{2}\ln \beta^{n-k}$.

Proof. Choose an h for which Theorem 9 holds. For every pair (a,b) satisfying Theorem 9 and every integer satisfying $\beta^{k-1} \leq c < \beta^{k-\frac{1}{2}}$ we obtain a pair (ac,bc) with $ac \geq bc$, L(ac)=L(bc)=n, L(gcd(ac,bc))=L(c)=k, and $D(ac,bc)=D(a,b) \geq \frac{1}{2}\ln a \geq \frac{1}{2}\ln \beta^{n-k}$. The mapping f((a,b),c)=(ac,bc) thus defined is one-one so there are at least $(0.02\beta^{2n-2k+1})(\sqrt{\beta}-1)\beta^{k-1} \geq 0.008\beta^{2n-k+1}$ pairs (a,b) with $a \geq b$, L(a)=L(b)=n, L(gcd(a,b))=k and $D(a,b) \geq \frac{1}{2}\ln \beta^{n-k}$. 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 $\lfloor(\beta^m-\beta^{m-1})/a\rfloor \geq (\beta^m-\beta^{m-1})/\beta^n \geq (1-\beta^{-1})\beta^{m-n-1} \geq \frac{1}{2}\beta^{m-n}$ pairs (aq+b,a) with L(aq+b)=m. Since gcd(aq+b,a)=gcd(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 $aq+b \geq a$, $L(aq+b)=m$, $L(a)=n$, $L(gcd(aq+b),a))=k$ and $D(aq+b,a) \geq \frac{1}{4}\ell n\, \beta^{n-k}$. ∎

<u>Theorem 11</u>. $t_E^*(m,n,k) \sim n(m-k+1)$.

<u>Proof</u>. Let $c_1=\min(1,\frac{1}{2}\ell n\, \beta)$. By Theorems 4 and 10, there exist h and $c_2 > 0$ such that $t_E(a,b) \geq c_2 D(a,b)\{D(a,b)+L(gcd(a,b))\} \geq c_2 c_1(n-k)\{c_1(n-k)+k\} \geq c_1^2 c_2 n(n-k)$ 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, $t_E^*(m,n,k) \geq 0.004\, c_1^2 c_2 \beta^{-2} n(n-k) \sim n(n-k)$ for $n-k>h$. By Theorem 7, $t_E^*(m,n,k) \succeq n(m-n+1) \geq n \succeq n(n-k)$ for $n-k \leq h$. Hence by Theorem 1, Part (h), $t_E^*(m,n,k) \succeq n(n-k)$. By Theorem 7, $t_E^*(m,n,k) \succeq n(m-n+1)$ so by Theorem 1, Part (c), $t_E^*(m,n,k) \succeq n(n-k)+n(m-n+1)=n(m-k+1)$. Hence by Theorem 6, $t_E^*(m,n,k) \sim n(m-k+1)$.

REFERENCES

1. W. S. Brown, On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors, Journal. A.C.M., Vol. 18, No. 4 (Oct. 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, pp. 197-231, IBM Corp., Cambridge, Mass., 1969.

3. G. E. Collins, The Calculation of Multivariate Polynomial Resultants, Jour. A.C.M., Vol. 18, No. 4 (Oct. 1971), pp. 515-532.

4. G. E. Collins, The SAC-l Integer Arithmetic System-Version III, Univ. of Wisconsin Computer Science Department Technical Report No. 156, July, 1972, 63 pp.

5. N. G. DeBruijn, Asymptotic Methods in Analysis, North-Holland Publishing co., Amsterdam, 1961.

6. J. D. Dixon, The Number of Steps in the Euclidean Algorithm, Jour. Number Theory, Vol. 2, No. 4 (Nov. 1970), pp. 414-422.

7. J. D. Dixon, A Simple Estimate for the Number of Steps in the Euclidean Algorithm, Am. Math. Monthly, Vol. 78, No. 4 (April 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, Jour. A.C.M., Vol. 18, No. 4 (Oct. 1971), pp. 533-548.

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

11. D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, Addison-Wesley, Reading, Mass., 1969.

12. D. R. Musser, Algorithms for Polynomial Factorization, Univ. of Wisconsin Ph.D. Thesis (Computer Sciences Dept. Tech. Report No. 134), Sept. 1971, 174 pages.