EUCLIDEAN GCD ALGORITHM IS NOT OPTIMAL

Nader H. Bshouty

Department of Computer Science, Technion-I.I.T, Haifa, Israel
Department of Computer Science, University of Calgary
2500 University Drive N.W.
Calgary, Alberta, Canada T2N 1N4
e-mail: bshouty@cpsc.ucalgary.ca

To my daughter Vivian, 23/11/89.

ABSTRACT:

Using the operations $\{+,-\}$, multiplication and division by constants $\{\times_{\mathbf{Z}}, /_{\mathbf{Z}}\}$, floor operation, $\{\lfloor\ \rfloor\}$ and indirect addressing, we compute $GCD(x,y)$, $x,y \in [0,N]$ and find $a,b \in [0,N]$ such that $ax+by=GCD(x,y)$ with operation complexity

$$O\left(\frac{\log N}{\log\log N}\right)$$

and space complexity $O((\log N)^\epsilon)$ for any constant $0<\epsilon<1$. The numbers that are produced in the algorithms are less than $max(x,y)$. We also prove that using these operations our bound is tight.

In the boolean model we prove that to obtain this upper bound we must use $\Omega((\log N)^\epsilon)$ space for some constant ϵ.

We also study the direct sum complexity of GCD and prove that GCD function does not satisfy the direct sum conjecture and we study the operation complexity of computing GCD and LCM of n numbers and find tight bounds for these problems.

1. INTRODUCTION

Let $\mathbf{Q}$ be the rational field. A random access machine $RAM(F)$ with S space and W word length is a RAM with registers $\{M[i]\}_{i=1,\cdots,S}$ and accumulator A each of which can store an element with binary representation of length W. The computation is directed by finite program P that consists instructions $P_1,\cdots,P_t$ of the following type:($e \in \{1,\cdots,S\}$).

(1) $A \leftarrow 0$, $A \leftarrow M[e]$.

(2) $M[e] \leftarrow A$.

(3) $A \leftarrow A \circ M[e]$ where $\circ \in F$ is a function $\mathbf{Q} \times \mathbf{Q} \rightarrow \mathbf{Q}$.

(4) IF $< \circ A >$ THEN GOTO d_1 ELSE GOTO d_2, where $\circ \in F$ is a function $\mathbf{Q} \rightarrow \{YES, NO\}$.

When $\leftarrow \in F$ then we also allowed indirect addressing:

(5) $A \leftarrow M[M[e]]$, $M[M[e]] \leftarrow A$,

and when $\{C\} \in F$ where $C \subseteq \mathbf{Q}$ then we allowed

(6) $A \leftarrow c$ where $c \in C$.

We say that the program P_f with the operations F *computes* $f : D \rightarrow \mathbf{Q}^s$ over the domain $D \subseteq \mathbf{Q}^r$ if the execution of the program for the input $x = (M[1],\cdots,M[r]) \in D$ stops with output $(M[1],\cdots,M[s]) = f(x)$. The *operation complexity* $Comp_F(P_f)$ *of the program* P_f is the maximal number of steps of the above form that are executed in P_f over all possible inputs $x \in D$. The *operation complexity of the function* f over the domain D is $min_{P_f} Comp_F(P_f)$ over all programs P_f that compute f over the domain D.

The set F can contain the following:

$+,-,\times,/$	Arithmetic operations on $\mathbf{Q}$
$\lfloor\ \rfloor$, mod	Floor and modulu operations
$\wedge,\vee,\sim,\otimes$	Bitwise integer boolean operations
Rot_r, Shi_r	Rotate and shift r times
$log_r, \|\ \|$	$\lfloor \log_r x \rfloor$. $\|\|x\|\|$ is the length of the binary representation of x
$\circ_G$	$\circ$ operation with second operand from G
$\lfloor /G \rfloor$	Integer division by $g \in G$
$\leftarrow$	Indirect addressing
$>$	Comparisons $<,>,=$
$=$	Equation $=$
E	Any YES/NO question
$G[\alpha_1,\cdots,\alpha_v]$	Constants from G depend on $\alpha_1,\cdots,\alpha_v$.

Let $D_N = [0,N]$ be the set of integers $\{0,\cdots,N\}$ and let $GCD : D_N \times D_N \rightarrow \mathbf{Q}$ be the greatest common divisor function. For upper bound for computing the GCD function over the domain $D_N \times D_N$ we mention two algorithms. The first is Euclidean algorithm, [AHU], [K], in $RAM(+,-,\times,/,\lfloor\ \rfloor,=)$ or $RAM(mod,=)$ with space $O(1)$, word length $||N|| = \lfloor logN \rfloor + 1$ and operation complexity

$$O(\log N).$$

The second algorithm with the same upper bound is established by Stein, [S], in the model $RAM(\{1\},+,-,Shi_1\ ,>)$ with space $O(1)$ and word length $||N||$.

Few lower bounds are known for the GCD. Mansour-Shieber-Tiwari , [MST1], proved the lower bound

$$\Omega(\log\log\log N)$$

in the model $RAM(\{1\},+,-,\times,/,\lfloor\ \rfloor,\leftarrow,>)$. (When we do not mention the word length and the space, we assume them to be ∞).

Other lower bounds established in [B2], [B3] and [B5] are: A lower bound $\Omega(\log N)$ in $RAM(\mathbf{Q},+,-,\times,/,\leftarrow,E)$ and

$$\Omega\left(\frac{\log N}{\log\log N}\right)$$

in $RAM(C,+,-,\times,\wedge,\vee,\otimes,\sim,Shi_1,\leftarrow,E)$ where C is any set of constants with $|C| \leq poly(\log N)$.

2. NEW RESULTS

In this paper we prove the following results

Theorem i . *Let $0 < \epsilon < 1$ be a constant. Using $F = \{\{1\},+,-,\times_{\mathbf{Z}},\lfloor/2^i\rfloor,\leftarrow,>\}$, we can compute $GCD(x,y)$, $x,y \in [0,N]$ by*

$$O\left(\frac{\log N}{\log\log N}\right)$$

operations with $||N||$ word length and space $O((\log N)^{\epsilon})$.

Our algorithm is based on Stein algorithm. Stein algorithm first finds the maximal w such that $2^w|GCD(x,y)$ and then uses the following algorithm: (1) $GCD(x,y) = GCD(x/2,y)$ if x is even and y is odd, (2) $GCD(x,y) = GCD(x,y/2)$ if x is odd and y is even. (3) $GCD(x,y) = GCD((x-y)/2,y)$ if x and y are odd and $y < x$. (4) $GCD(x,y) = GCD((y-x)/2,x)$ if x and y are odd and $x \leq y$. The operation complexity of Stein algorithm is $O(\log N)$ since every operation reduce $||x|| + ||y||$ by 1.

We observed that the first and last $\lfloor\log\log N\rfloor$ bits in the binary representation of x and y uniquely determine the first $\lfloor\log\log N\rfloor - O(\log\log\log N)$ operations in Stein algorithm. We save these information and uses them to reduce $||x||$ or $||y||$ by $\lfloor\log\log N\rfloor - O(\log\log\log N)$ with $O(1)$ steps. This give the upper bound.

For a more stronger set of operations we have the same lower bound

Theorem ii . *Using $F = \{\mathbf{Q},+,-,\times_{\mathbf{Q}},/_{\mathbf{Q}},\lfloor\ \rfloor,\leftarrow,>\}$, any program that computes $GCD(x,y)$, $x,y \in [0,N]$ requires*

$$\Omega\left(\frac{\log N}{\log\log N}\right)$$

operations.

In the paper this lower bound is proved for a much more stronger model and for a much more weaker problem. Our lower bound is based on Just-Meyer-Wigderson result. We extend their result and use the extension result for the GCD function.

We also prove

Theorem iii . *Using* $F = \{\mathbf{Q}, +, -, \times_{\mathbf{Q}}, /_{\mathbf{Q}}, \lfloor\ \rfloor, Shi_{\mathbf{Z}}, Rot_{\mathbf{Z}}, \sim, \log_A, ||\ ||, \leftarrow, >\}$ *where* $A = \{2, 3, \cdots\}$ *with word length* $W = poly(\log N)$*, any program that computes* $GCD(x, y)$*,* $x, y \in [0, N]$ *requires*

$$\Omega\left(\frac{\log N}{\log\log N}\right)$$

operations.

It sometimes not convinent to people to use space more than O(number of inputs+ number of outputs), i.e $O(1)$ in the case of GCD. We prove that for some models, to obtain the above new upper bound we must use $(\log N)^\epsilon$ memory for some constant ϵ., i.e

Theorem iv . *Using* $\{C, +, -, \times, \wedge, \vee, \otimes, \sim, Shi_1, \leftarrow, E\}$*,* $|C| < \min(S, poly(\log N))$ *and space* S*, any program that compute* $GCD(x, y)$*,* $x, y \in [0, N]$ *requires*

$$\Omega\left(\frac{\log N}{\min(\log S, \log\log N)}\right)$$

operations.

Therefore

(1) If $S = O(1)$ *then we have* $\Omega(\log N)$ *lower bound for computing the GCD.*

(2) To obtain the $\Omega\left(\frac{\log N}{\log\log N}\right)$ *lower bound we must use at least* $S = (\log N)^\epsilon$ *space for some constant* ϵ.

In many applications we also need an algorithm that for $x, y \in [0, N]$ gives two numbers $a, b \in [0, N]$ such that $ax + by = GCD(x, y)$. We extend Theorem i and prove

Theorem v . *Let* $0 < \epsilon < 1$ *be a constant. Using* $F = \{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor/2^i\rfloor, \leftarrow, >\}$*, we can compute* $a, b \in [0, N]$ *such that* $ax + by = GCD(x, y)$ *by*

$$O\left(\frac{\log N}{\log\log N}\right)$$

operations with $||N||$ *word length and space* $O((\log N)^\epsilon)$.

Obviously, the lower bound for GCD is also a lower bound for computing a and b.

In the case where we need to compute $GCD(x_1, y_1), \cdots, GCD(x_t, y_t)$ it is not true that the operation complexity of this problem is equal to t times the operation complexity of computing one GCD. In this paper we show that we can do better

Theorem vi . *Using* $F = \{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor/2^i\rfloor, \leftarrow, >\}$*, we can compute* $GCD(x_1, y_1), GCD(x_2, y_2)$ $, \cdots,$ $GCD(x_t, y_t)$*,* $t < N$*,* $x_i, y_i \in [0, N]$ *by*

$$O\left(\frac{t\log N}{\log\log N + \log t}\right)$$

operations with space $O((t\log N)^\epsilon)$ *for any constant* $0 < \epsilon < 1$.

We also prove that this bound is tight

Theorem vii . *Using* $F = \{\mathbf{Q}, +, -, \times_{\mathbf{Q}}, /_{\mathbf{Q}}, \lfloor\ \rfloor, \leftarrow, >\}$, *any program that computes* $GCD(x_1, y)$, $GCD(x_2, y)$, $\cdots$, $GCD(x_t, y)$, $t < N$, $x_i, y \in [0, N]$ *requires*

$$O\left(\frac{t \log N}{\log\log N + \log t}\right)$$

operations.

Sometimes we need to compute $GCD(x_1, y_1), \cdots, GCD(x_t, y_t)$ where x_i, y_i are given only after computing $GCD(x_{i-1}, y_{i-1})$. In this case we usually build tables of resonable size M and use these tables to accelerate the computation of the GCD. Since we usually has large enough t, we assume that the tables can be built in small operation complexity and therefore we assume that the tables are given for free. We denote by $T(M)$ the set of all tables of size M. For computing the GCD with tables we have

Theorem viii . *Using* $F = \{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, \leftarrow, >, T(M)\}$, *we can compute* $GCD(x, y)$, $x, y \in [0, N]$ *by*

$$O\left(\frac{\log N}{\log\log N + \log M}\right)$$

operations.

We also prove that this bound is tight

Theorem ix . *Using* $F = \{\mathbf{Q}, +, -, \times_{\mathbf{Q}}, /_{\mathbf{Q}}, \lfloor\ \rfloor, \leftarrow, >, T(M)\}$, *any program that computes* $GCD(x, y)$, $x, y \in [0, N]$ *requires*

$$O\left(\frac{\log N}{\log\log N + \log M}\right)$$

operations.

For computation of GCD and LCM of many numbers we prove

Theorem x . *Using* $\{\{1\}, +, -, \times, /, \lfloor\ \rfloor, \leftarrow, >\}$ *we can compute* $GCD(x_1, x_2, \cdots, x_n)$, $x_1, x_2, \cdots, x_n \in [0, N]$ *by*

$$O\left(\frac{\log N}{\log\log N} + n\right)$$

operations.

Theorem xi . *Using* $\{+, -, \times, /, \lfloor\ \rfloor, \leftarrow, >\}$ *we can compute* $LCM(x_1, \cdots, x_n)$, $x_1, \cdots, x_n \in [0, N]$ *by*

$$O\left(\frac{n \log N}{\log\log N + \log n}\right)$$

operations.

Theorem xii . *Using* $F = \{\mathbf{Q}, +, -, \times_{\mathbf{Q}}, /_{\mathbf{Q}}, \lfloor\ \rfloor, \leftarrow, >\}$, *any program that computes* $LCM(x_1, \cdots, x_n)$, $x_1, \cdots, x_n \in [0, N]$ *requires*

$$\Omega\left(\frac{n \log N}{\log\log N + \log n}\right).$$

In the paper we prove more general results.

The paper is organized as follows: In section 3 we give a few words about the model. In section 4 we prove the upper bound in Theorem i. In section 5 we give an algorithm that computes a, b such that

$ax + by = GCD(x, y)$ and prove Theorem v. In section 6 we prove the lower bounds in Theorems ii, iii and iv. In section 7 we study the complexity of GCD using tables and prove Theorems viii and ix. In section 8 we define the t-Direct sum complexity of problems and study it for the GCD function and prove Theorem vi and vii. In section 9 we study the operation complexity of computing GCD and LCM of n numbers and prove Theorems x, xi and xii.

3. THE MODEL

The reson we define a RAM model with W word length and S space is that if we allow large word length and space then we can factor every composite number n by operation complexity $O(\log n)$, [Sh], multiply two $n \times n$ integer matrices by $O(n^2)$ operations, [BBF], sort n integer elements by $O(n)$ operations, [KR], [PS] , verify if $GCD(x, y) = 1$ by $O(1)$ operations, [B5], compute ϵ-accuracy of the square root of x by $O(1)$ operations [MST2] and solve the membership problem of n integer elements by $O(n)$ operations.

Our model read each input in one step and each operation is of 1 operation complexity, i.e we measure the complexity under the *worst-case complexity with uniform cost criterion* where each RAM instruction requires one unit of time and each register requires one unit of space. For probabilistic and deterministic, sequential and parallel bit complexity of the GCD problem we refer the reader to [AHU], [BGH], [Mo], [Sc0], [Sc1] and [Sc2].

With careful definition of indirect addressing in the polynomial computation model we also prove that Euclidean GCD algorithm is not optimal for polynomial over finite fields, [B7].

Our algorithms in this paper is practical and indeed accelerate the computation of GCD. But it is still an open problem if this upper bound can be established without indirect addressing. Valuable works done in the literature to accelerate and analyse Euclidean GCD algorithm. For details see [BR], [Br], [BT], [C], [D], [La], [Le], [Ma], [MG] and [YK].

4. UPPER BOUND FOR GCD

In this section we prove Theorem i.

Before we start we give some notations. The sets $\mathbf{N}, \mathbf{Z}$ and $\mathbf{Q}$ denote the set of positive integers, integers and rational numbers, respectively. For an integer x we denote by $[x] = x_n|x_{n-1}|\cdots|x_1$ the binary representation of x, i.e, $x = \sum_{i=1}^{n} x_i 2^{i-1}$, $[x]_{i,j}$ is the number $x_j|x_{j-1}|\cdots|x_i$ and $||x|| = n$ is the *length* of x. For two integers x and y we write $x|y$ if there exist an integer z such that $y = zx$ and we write $x^w||y$ if $x^w|y$ and $x^{w+1} \not| y$.

Let x and y be two n-bits numbers. To compute $GCD(x, y)$ we first compute the least $d = 2^w$ such that $2^w||GCD(x, y)$. The following algorithm take (x, y) as an input and output $(2^w, x', y')$ where x' or y' is odd and $GCD(x, y) = 2^w GCD(x', y')$.

Algorithm A

Even_GCD(x, y, d)

(1) If x or y is odd then $\{x' \leftarrow x,\ y' \leftarrow y,\ d \leftarrow 1,$ output (d, x', y'), return $\}$.

(2) $c_0 \leftarrow 2,\ i \leftarrow 0,\ x \leftarrow x/2,\ y \leftarrow y/2,\ d \leftarrow 2$.

(3) while $c_i | x$ and $c_i | y$ do

$\{x \leftarrow x/c_i,\ y \leftarrow y/c_i,\ d \leftarrow d \times c_i,\ i \leftarrow i+1, c_i \leftarrow c_{i-1} \times c_{i-1}\}$.

(4) For $j := i - 1$ to 0 do

$\{$ If $c_j | x$ and $c_j | y$ then $\{\ x \leftarrow x/c_j,\ y \leftarrow y/c_j,\ d \leftarrow d \times c_j\}\}$.

(5) $x' \leftarrow x,\ y' \leftarrow y$.

(6) output(d, x', y'), return.

Lemma 1. *We have*

(i) $GCD(x, y) = d\ GCD(x', y')$.

(ii) $d = 2^w || GCD(x, y)$.

(iii) *All the numbers that are produced in the algorithm are less than* $\max(x, y)$.

(iv) *The algorithm has operation complexity and space complexity* $O(\log\log(\min(x, y)))$.

Proof . If we execute the algorithm for the inputs

$$x = 2^{2^{i_1}+\cdots+2^{i_t}}\bar{x}, \qquad y = 2^{2^{i_1}+\cdots+2^{i_t}}\bar{y},\ \ i_1 > i_2 > \cdots > i_t \geq 0,$$

where $\bar{x}$ or $\bar{y}$ is odd then we have: After executing step (3) we have

$$d = 2 \times 2 \times 2^2 \cdots 2^{2^{i_1-1}} = 2^{2^{i_1}}, \qquad c_j = 2^{2^j},\ \ j = 1, \cdots, i_1$$

and the maximal number that is produced at this step is $2^{2^{i_1}} \leq \min(x, y)$. In step (4) we execute the instruction after the **then** for $j = i_2, \cdots, i_t$ and obtain $d = 2^{2^{i_1}+\cdots+2^{i_t}}$. Now (i)-(iii) are obvious.

Since $2^{2^{i_1}} \leq \min(x, y)$ we have $i_1 \leq \log\log\min(x, y)$ which implies (iv). ○

This follows

Lemma 2. *Using the operations* $RAM(\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, =)$ *we can compute* $d = 2^w || GCD(x, y)$ *by operation complexity and space complexity* $O(\log\log(min(x, y)))$ *and word length* $||\max(x, y)||$.

Proof . The question $c|x?$ is equivalent to the question $x - c\lfloor x/c \rfloor = 0?$. The rest follows from lemma 1. ○

We now assume that x or y is odd.

The following algorithm takes as inputs x_1, x_2 and y_1, y_2 the first and last k bits of $[x]$ and $[y]$, respectively, and computes integer numbers $\alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2$ where

$$X = \frac{\alpha_1 x + \beta_1 y}{\gamma_1}, \quad Y = \frac{\alpha_2 x + \beta_2 y}{\gamma_2}$$

are the numbers that are produced by executing $k - O(\log k)$ steps in Stein algorithm.

Algorithm B

Let $x_1, x_2, y_1, y_2, \alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2, l_1, l_2$ be integers where $\|x_1\|, \|x_2\|, \|y_1\|, \|y_2\| \leq k$.

Part_GCD(x_1, x_2, y_1, y_2)

(1′) $\alpha_1 \leftarrow 1,\ \beta_1 \leftarrow 0,\ \gamma_1 \leftarrow 1,\ l_1 \leftarrow 0\ \alpha_2 \leftarrow 0,\ \beta_2 \leftarrow 1,\ \gamma_2 \leftarrow 1,\ l_2 \leftarrow 0$

(2′) While $|x_2 - y_2| \geq l_1 + l_2 + 2$ and $l_1 + l_2 < k$ do GCD_First_Last.

(3′) If $x_2 \geq l_1 + l_2 + 2$ and $y_2 \geq l_1 + l_2 + 2$ and $l_1 + l_2 \neq k$ then

$\{\gamma \leftarrow \max(\gamma_1, \gamma_2),\ \alpha_1 \leftarrow \frac{\gamma}{\gamma_1}\alpha_1 - \frac{\gamma}{\gamma_2}\alpha_2,\ \beta_1 \leftarrow \frac{\gamma}{\gamma_1}\beta_1 - \frac{\gamma}{\gamma_2}\beta_2,\ \gamma_1 \leftarrow \gamma\}$

(4′) return.

GCD_First_Last

(1) If $2|x_1$ then

$\{x_1 \leftarrow x_1/2,\ x_2 \leftarrow \lfloor x_2/2 \rfloor,\ \gamma_1 \leftarrow 2\gamma_1,\ l_1 \leftarrow l_1 + 1,$ return$\}$

(2) If $2|y_1$ then

$\{y_1 \leftarrow y_1/2,\ y_2 \leftarrow \lfloor y_2/2 \rfloor,\ \gamma_2 \leftarrow 2\gamma_2,\ l_2 \leftarrow l_2 + 1,$ return$\}$

(3) If $x_2 > y_2$ then $\{x_1 \leftarrow \frac{x_1 - y_1}{2},$ If $x_1 < 0$ then $x_1 \leftarrow x_1 + 2^{k-1},\ x_2 \leftarrow \lfloor \frac{x_2 - y_2}{2} \rfloor$

$\gamma \leftarrow \max(\gamma_1, \gamma_2),\ \alpha_1 \leftarrow \frac{\gamma}{\gamma_1}\alpha_1 - \frac{\gamma}{\gamma_2}\alpha_2,\ \beta_1 \leftarrow \frac{\gamma}{\gamma_1}\beta_1 - \frac{\gamma}{\gamma_2}\beta_2,\ \gamma_1 \leftarrow 2\gamma,\ l_1 \leftarrow l_1 + 1,$ return$\}$

(4) If $x_2 \leq y_2$ then $\{y_1 \leftarrow \frac{y_1 - x_1}{2},$ If $y_1 < 0$ then $y_1 \leftarrow y_1 + 2^{k-1},\ y_2 \leftarrow \lfloor \frac{y_2 - x_2}{2} \rfloor$

$\gamma \leftarrow \max(\gamma_1, \gamma_2),\ \alpha_2 \leftarrow \frac{\gamma}{\gamma_2}\alpha_2 - \frac{\gamma}{\gamma_1}\alpha_1,\ \beta_2 \leftarrow \frac{\gamma}{\gamma_2}\beta_2 - \frac{\gamma}{\gamma_1}\beta_1,\ \gamma_2 \leftarrow 2\gamma,\ l_2 \leftarrow l_2 + 1,$ return$\}$

Lemma 3. *Let x and y be two positive integers, where $w \geq \|x\|, \|y\| \geq 2k$. Let*

$$x_{1,1} = [x]_{1,k},\quad x_{2,1} = [x]_{w-k+1,w},$$

$$y_{1,1} = [y]_{1,k},\quad y_{2,1} = [y]_{w-k+1,w}.$$

Let $(x_{j,i}, y_{j,i}, \alpha_{j,i}, \beta_{j,i}, \gamma_{j,i}, l_{j,i})$, $i = 1, \cdots, c$, $j = 1, 2$ be the numbers that are produced from calling the procedure Part_GCD$(x_{1,1}, x_{2,1}, y_{1,1}, y_{2,1})$ and $\alpha_{j,c+1}, \beta_{j,c+1}, \gamma_{j,c+1}$, $j = 1, 2$ are the number produced after executing step (3′). Then

(*i*) $c \leq k$.

(*ii*) $GCD\left(\frac{\alpha_{1,i}x + \beta_{1,i}y}{\gamma_{1,i}}, \frac{\alpha_{2,i}x + \beta_{2,i}y}{\gamma_{2,i}}\right) = GCD(x, y)$.

(*iii*) $\left\|\frac{\alpha_{1,c+1}x + \beta_{1,c+1}y}{\gamma_{1,c+1}}\right\| + \left\|\frac{\alpha_{2,c+1}x + \beta_{2,c+1}y}{\gamma_{2,c+1}}\right\| \leq w - k + 1 + \lceil \log(k+1) \rceil$.

Proof . It can be easily prove that each time we call *GCD_First_Last*, $l_{1,i} + l_{2,i}$ is increased by 1. By the condition $l_1 + l_2 < k$ in (2'), (*i*) follows. Also, when $l_{2,i}$, ($l_{1,i}$), is increased by 1 then $\|x_{2,i}\|$, ($\|y_{2,i}\|$), is reduced by at least 1 and $l_{1,i} + l_{2,i}$ is the number of times we use the procedure *GCD_First_Last*. Therefore

with the condition in step (2′) we have

$$l_{1,i} + l_{2,i} \leq k, \qquad l_{i,1} + l_{2,i} = i - 1. \tag{1}$$

Consider the following steps

$$(\bar{1})\ x \leftarrow x/2,\ (\bar{2})\ y \leftarrow y/2,\ (\bar{3})\ x \leftarrow (x-y)/2,\ (\bar{4})\ y \leftarrow (y-x)/2.$$

Let $(\pi_1), \cdots, (\pi_c)$ be the steps that are executed for the input $x_{1,1}, x_{2,1}, y_{1,1}, y_{2,1}$ in *GCD_First_Last*. Let $x_i, y_i,\ i = 1, \cdots, c$ be the sequence that is produced from executing $(\bar{\pi}_1), \cdots, (\bar{\pi}_c)$ for $x_1 = x$ and $y_1 = y$. We now prove the following by induction hypothesis:

(A) x_i, y_i are positive integers.

(B) For $i \neq c$, $l = l_{1,i} + l_{2,i}$ we have $[x_{1,i}]_{1,k-l} = [x_i]_{1,k-l}, \qquad [y_{1,i}]_{1,k-l} = [y_i]_{1,k-l}$.

(C) $x_i = \frac{\alpha_{1,i}x + \beta_{1,i}y}{\gamma_{1,i}},\ y_i = \frac{\alpha_{2,i}x + \beta_{2,i}y}{\gamma_{2,i}}$.

(D) $GCD\left(\frac{\alpha_{1,i}x + \beta_{1,i}y}{\gamma_{1,i}}, \frac{\alpha_{2,i}x + \beta_{2,i}y}{\gamma_{2,i}}\right) = GCD(x_i, y_i) = GCD(x, y)$.

(E) $|x_i - 2^{w-k}x_{2,i}| < (i-1)2^{w-k-1} + 2^{w-k}, \qquad |y_i - 2^{w-k}y_{2,i}| < (i-1)2^{w-k-1} + 2^{w-k}$.

For $i = 1$, since $x_{1,1} = [x]_{1,k}$, $y_{1,1} = [y]_{1,k}$, $x_1 = x$, $y_1 = y$, $\alpha_{1,1} = 1$, $\beta_{1,1} = 0$, $\gamma_{1,1} = 1$, $\alpha_{2,1} = 0$, $\beta_{2,1} = 1$ and $\gamma_{2,1} = 1$, (A)-(D).follows. For (E) we have

$$|x_1 - 2^{w-k}x_{2,1}| = |[x]_{1,w-k}| < 2^{w-k}.$$

Assume that (A)-(E) are true for i. Observe that steps (3) and (4) are executed only when $x_{1,i}$ and $y_{1,i}$ are odd. Therefore, $x_{1,i+1}, y_{1,i+1}$ are integers.

Since, by (B) and (1) we have $[x_{1,i}]_1 = [x_i]_1$ and $[y_{1,i}]_1 = [y_i]_1$ and since $x_{1,i+1} = x_{1,i}/2$ or $= (x_{1,i} - y_{1,i})/2$ or $= x_{1,i}$ (and so $y_{1,i+1}$) are integers, we have

$$x_{i+1}, y_{i+1} \text{ are integers.} \tag{2}$$

Now for (E) we have four cases. If $(\pi_{i+1}) = (1)$ then

$$x_{i+1} = x_i/2,\ y_{i+1} = y_i,\ x_{1,i+1} = x_{1,i}/2,\ x_{2,i+1} = \lfloor x_{2,i}/2 \rfloor,\ y_{1,i+1} = y_{1,i},\ y_{2,i+1} = y_{2,i}.$$

Then for some $\lambda \in \{0, 1\}$ we have

$$|x_{i+1} - 2^{w-k}x_{2,i+1}| = |x_i/2 - 2^{w-k}\lfloor x_{2,i}/2 \rfloor| = \left|\frac{x_i - 2^{w-k}x_{2,i}}{2} + \lambda 2^{w-k-1}\right|$$

$$< (i-1)2^{w-k-2} + 2^{w-k} < i2^{w-k-1} + 2^{w-k}.$$

and

$$|y_{i+1} - 2^{w-k}y_{2,i+1}| = |y_i - 2^{w-k}y_{2,i}| < (i-1)2^{w-k-1} + 2^{w-k} < i2^{w-k-1} + 2^{w-k}.$$

The second case $(\pi_{i+1}) = (2)$ is similar to the previous case.

The third case is $(\pi_{i+1}) = (3)$. In this case we have

$$x_{i+1} = \frac{x_i - y_i}{2},\ y_{i+1} = y_i, x_{1,i+1} = \frac{x_{1,i} - y_{1,i} + \delta 2^k}{2},\ x_{2,i+1} = \left\lfloor \frac{x_{2,i} - y_{2,i}}{2} \right\rfloor,\ y_{1,i+1} = y_{1,i},\ x_{2,i+1} = x_{2,i},$$

where $\delta = 1$ if and only if $x_{1,i} - y_{1,i} < 0$ and otherwise $\delta = 0$. Since $x_{2,i} \geq y_{2,i}$ we have, for some $\lambda \in \{0,1\}$

$$|x_{i+1} - 2^{w-k} x_{2,i+1}| = \left| \frac{x_i - y_i}{2} - 2^{w-k} \left\lfloor \frac{x_{2,i} - y_{2,i}}{2} \right\rfloor \right| = \left| \frac{x_i - y_i}{2} - 2^{w-k} \frac{x_{2,i} - y_{2,i} - \lambda}{2} \right|$$

$$= \left| \frac{(x_i - 2^{w-k} x_{2,i}) - (y_i - 2^{w-k} y_{2,i})}{2} + 2^{w-k-1}\lambda \right| < (i-1)2^{w-k-1} + 2^{w-k} + 2^{w-k-1} = i2^{w-k-1} + 2^{w-k}.$$

and as before

$$|y_{i+1} - 2^{w-k} y_{2,i+1}| < i2^{w-k-1} + 2^{w-k}.$$

The case $(\pi_{i+1}) = (4)$ is similar to the last case. This follows

$$|x_{i+1} - 2^{w-k} x_{2,i+1}| < i2^{w-k-1} + 2^{w-k},\ |y_{i+1} - 2^{w-k} y_{2,i+1}| < i2^{w-k-1} + 2^{w-k}. \tag{3}$$

We now prove (B). By (3) we have

$$2^{w-k}(x_{2,i+1} - y_{2,i+1}) - (i2^{w-k} + 2^{w-k+1}) < x_{i+1} - y_{i+1} < 2^{w-k}(x_{2,i+1} - y_{2,i+1}) + (i2^{w-k} + 2^{w-k+1}). \tag{4}$$

Therefore whenever, $|x_{2,i+1} - y_{2,i+1}| \geq l_{1,i+1} + l_{2,i+1} + 2 = i + 2$, see step $(2')$ and (1), we have

$$x_{i+1} > y_{i+1} \quad \text{iff} \quad x_{2,i+1} > y_{2,i+1}.$$

Therefore with (2) we have

$$x_{i+1}, y_{i+1} \text{ are positive integers.}$$

Now by (B) and (1) it can be easily prove that for $l = l_{1,i+1} + l_{1,i+1} = l_{1,i} + l_{2,i} + 1$ we have

$$[x_{i+1}]_{1,k-l} = [x_{1,i+1}]_{1,k-l},\ \ [y_{i+1}]_{1,k-l} = [y_{1,i+1}]_{1,k-l},$$

which implies (B).

For (C) we have the following cases. If $(\pi_{i+1}) = (1)$ then $x_{i+1} = x_i/2$ and

$$x_{i+1} = \frac{\alpha_{1,i}x + \beta_{1,i}y}{2\gamma_{1,i}} = \frac{\alpha_{1,i+1}x + \beta_{1,i+1}y}{\gamma_{1,i+1}}.$$

The case $(\pi_{i+1}) = (2)$ where $y_{i+1} = y_i/2$ is similar to the previous. If $(\pi_{i+1}) = (3)$ (or (4)) then $x_{i+1} = (x_i - y_i)/2$. Since $\gamma_{1,i}$ and $\gamma_{2,i}$ are power of two we have

$$x_{i+1} = \frac{\frac{\alpha_{1,i}x + \beta_{1,i}y}{\gamma_{1,i}} - \frac{\alpha_{2,i}x + \beta_{2,i}y}{\gamma_{2,i}}}{2} = \frac{\alpha_{2,i+1}x + \beta_{2,i+1}y}{\gamma_{2,i+1}}.$$

For (D). Since $GCD(x_i, y_i) = GCD(x, y)$ and x or y is odd, then x_i or y_i is odd. If $x_{i+1} = x_i/2$ or $y_{i+1} = y_i/2$ then since x_{i+1} and y_{i+1} are integers we have $GCD(x_{i+1}, y_{i+1}) = GCD(x_i, y_i) = GCD(x, y)$. If

$x_{i+1} = (x_i - y_i)/2$ or $y_{i+1} = (y_i - x_i)/2$ then since x_{i+1} and y_{i+1} are integers we also have $GCD(x_{i+1}, y_{i+1}) = GCD(x_i, y_i) = GCD(x, y)$. This complete the proof of (A)-(E). (D) follows (ii).

We now return to prove (iii). When the algorithm stops we have $|x_{2,c} - y_{2,c}| < l_{1,c} + l_{2,c} + 2 \leq k + 1$ or $l_{1,c} + l_{2,c} = k$. If $l_{1,c} + l_{2,c} = k$ then this means that $\|x_c\| + \|y_c\|$ is k (bits) less that $\|x\| + \|y\|$. Then $x_{c+1} = x_c$, $y_{c+1} = y_c$ and (iii) follows. If $l_{1,c} + l_{2,c} \neq k$ then

$$|x_{2,c} - y_{2,c}| < l_{1,c} + l_{2,c} + 2 \leq k + 1$$

Now two cases can happen. If $x_{2,c} < l_{1,c} + l_{2,c} + 2 \leq k + 1$ then $x_{c+1} = x_c$, $y_{c+1} = y_c$ and then by (4) we have

$$0 < x_c \leq 2^{w-k} x_{2,c} + (c-1)2^{w-k-1} + 2^{w-k} < (3k+3)2^{w-k-1} \leq 2^{w-k-1+\lceil \log(3k+3) \rceil}$$

and therefore

$$\|x_c\| \leq w - k - 1 + \lceil \log(3k+3) \rceil.$$

Also since $\{y_i\}$ are positive we have $\|y_c\| \leq \|y\|$ and this follows (iii).

If $y_{2,c} < l_{1,c} + l_{2,c} + 2 \leq k + 1$ then

$$\|y_c\| \leq w - k - 1 + \lceil \log(3k+3) \rceil, \quad \|x_c\| \leq \|x\|.$$

Now if $|x_{2,c}|, |y_{2,c}| \geq l_{1,c} + l_{2,c} + 2$, then $x_{c+1} = x_c - y_c$, $y_{c+1} = y_c$ and by (4) we have

$$|x_{c+1}| = |x_c - y_c| < 2^{w-k}|x_{2,c} - y_{2,c}| + (c-1)2^{w-k} + 2^{w-k+1} \leq$$

$$(k+1)2^{w-k} + (k-1)2^{w-k} + 2^{w-k+1} \leq (k+1)2^{w-k+1}$$

which implies that

$$\|x_{c+1}\| \leq w - k + 1 + \lceil \log(k+1) \rceil.$$

Since $\{y_i\}$ are positive we have $\|y_{c+1}\| = \|y_c\| \leq \|y\|$ and (iii) follows. ○

We now give upper bounds for the numbers that are produced in algorithm B.

Lemma 4. *We have*

(i) $\gamma_i \triangleq \max(\gamma_{1,i}, \gamma_{2,i}) = \gamma_{2-(\pi_i \bmod 2),i} = 2^{i-1}$.

(ii) $|\alpha_{j,i}|, |\beta_{j,i}| \leq \gamma_{j,i}, \quad j = 1, 2.$

Proof . By $(1')$ in Part_GCD, (i) and (ii) are true for $i = 1$. Assume that (i) and (ii) are true for i. If $(\pi_{i+1}) = (1)$ then $(\pi_i) = (1)$ or $(\pi_i) = (3)$ and therefore $\gamma_i = \gamma_{1,i} = \max(\gamma_{1,i}, \gamma_{2,i}) = 2^{i-1}$. Then $\gamma_{1,i+1} = 2\gamma_{1,i}$, $\gamma_{2,i+1} = \gamma_{2,i}$ and

$$\gamma_{i+1} = \gamma_{1,i+1} = \max(\gamma_{1,i+1}, \gamma_{2,i+1}) = 2^i.$$

The case where $(\pi_{i+1}) = (2)$ is similar to the previous case. If $(\pi_{i+1}) = (3)$ then $\gamma_{1,i+1} = 2\max(\gamma_{1,i}, \gamma_{2,i}) = 2^i$, $\gamma_{2,i+1} = \gamma_{2,i}$ and then $\gamma_i = \gamma_{1,i+1} = 2^i$. The case where $(\pi_{i+1}) = (4)$ is similar to the previous case.

We now prove (ii). If $(\pi_{i+1}) = (1)$ or (2) we have $\alpha_{j,i+1} = \alpha_{j,i} \leq \gamma_{j,i} \leq \gamma_{j,i+1}$ and so for $\beta_{j,i+1}$. If $(\pi_{i+1}) = (3)$ we have

$$|\alpha_{1,i+1}| = \left|\frac{\gamma_i}{\gamma_{1,i}}\alpha_{1,i} - \frac{\gamma_i}{\gamma_{2,i}}\alpha_{2,i}\right| \leq 2\gamma_i = \gamma_{1,i+1}.$$

Similar proof for $\beta_{1,i+1}$ and the case where $(\pi_{i+1}) = (4)$. ○

To proceed we need Stein algorithm that compute the GCD when x or y is odd.

Algorithm C

Stein_GCD(x, y, g)

While $y \neq 0$ do *Stein*(x, y).

$g = x$.

return.

Stein(x, y)

While $2|x$ do $\{x \leftarrow x/2$, return $\}$.

While $2|y$ do $\{y \leftarrow y/2$, return $\}$.

If $x > y$ then $\{x \leftarrow (x - y)/2\}$ else $\{y \leftarrow (y - x)/2\}$.

return.

We now give our GCD algorithm.

Let N be an integer and $x, y \in [0, N]$. Let $n = ||N||$, $0 < \tau < \frac{1}{4}$, $\epsilon = 4\tau$ and $k = k_\tau = \lfloor \tau \log_2 n \rfloor$. In the algorithm we use $\alpha(), \beta(), \gamma()$ that has initial value "nill". (This can be done with indirect addressing by $O(1)$ steps).

Algorithm D

$GCD(x,y)$

(1) $Even\_GCD(x,y,d)$.

(2) $t \leftarrow \max(\|x\|,\|y\|)$

(3) If $t < 2k$ then goto 9.

(4) $x_1 \leftarrow [x]_{1,k}$, $x_2 = [x]_{t-k+1,k}$, $y_1 \leftarrow [y]_{1,k}$, $y_2 = [y]_{t-k+1,k}$.

(5) If $\alpha_1(x_1x_2y_1y_2) = nill$ then

$\{Part\_GCD$, for $j = 1$ to 2 do $\{\alpha_j(x_1x_2y_1y_2) = \alpha_j,\ \beta_j(x_1x_2y_1y_2) = \beta_j,\ \gamma_j(x_1x_2y_1y_2) = \gamma_j\}\}$.

(6) $x \leftarrow \left|\frac{\alpha_1(x_1x_2y_1y_2)x+\beta_1(x_1x_2y_1y_2)y}{\gamma_1(x_1x_2y_1y_2)}\right|$, $\quad y \leftarrow \left|\frac{\alpha_2(x_1x_2y_1y_2)x+\beta_2(x_1x_2y_1y_2)y}{\gamma_2(x_1x_2y_1y_2)}\right|$.

(7) If $x = 0$ or $y = 0$ then { output$(d(x+y))$, stop}.

(8) Goto 3.

(9) Stein_GCD(x,y,g), output(dg).

Lemma 5. *Algorithm D computes the GCD of $x, y \in [0,N]$ in $O(\log N/\log\log N)$ opprations and $O((\log N)^\epsilon)$ space and the numbers that are produced in the algorithm are less than or equal to N.*

Proof . All the steps in the algorithm preserve the GCD of x and y. By lemma 3 steps (3)-(7) preserve the GCD of x and y. Step (9) is Stien algorithm.

By lemma 2 step (1) has operation and space complexity $O(\log\log N)$.

In step (5) the instruction after the **then** is executed for at most 2^{4k} values x_1, x_2, y_1, y_2. By lemma 3 each execution requires $O(k)$ operations. Therefore step (5) requires at most

$$O(k)2^{4k} \le O(\log\log(N)(\log N)^{4\tau}) = o\left(\frac{\log N}{\log\log N}\right) \tag{5}$$

operations. The space needed for step (5) is

$$6 \times 2^{4k} \le O((\log N)^\epsilon),$$

and by lemma 4 the numbers that are produced in step 5 are

$$\alpha_j(x_1x_2y_1y_2), \beta_j(x_1x_2y_1y_2), \gamma_j(x_1x_2y_1y_2) \le 2^{2k} \le (\log N)^{\epsilon/2}, \quad j = 1,2. \tag{6}$$

Therefore by lemma A.2, (6) can be computed by $\|N\|$ word length.

Step (9) is executed only when $\|x\| \le 2k$ and $\|y\| \le 2k$ and therefore it requires operation complexity $O(k) = O(\log\log N)$.

Now by lemma 3 the loop (3)-(8) reduce $\|x\| + \|y\|$ by $k - 2 - \lceil \log k \rceil$. Therefore the number of times we execute this loop is

$$T \le \frac{\|x\| + \|y\|}{k - O(\log k)} = O\left(\frac{\log N}{\log\log N}\right).$$

In this loop each step has operation complexity $O(1)$ except step (5) that totaly requires $o\left(\frac{\log N}{\log\log N}\right)$ operations. This follows the result. ○

We now proof theorem i

Theorem 1. *Let $0 \leq \epsilon \leq 1$ be a constant. Using $F = \{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, >\}$, we can compute $GCD(x,y)$, $x, y \in [0, N]$ by*

$$O\left(\frac{\log N}{\log\log N}\right)$$

operations with $\log N$ word length and space $O((\log N)^\epsilon)$.

Proof . We can assume that $N = \max(x,y)$. For algorithm D we need $n = \|N\|$, $k = k_\tau = \lfloor \tau \log_2 n \rfloor$ and $\tau = \frac{\epsilon}{4}$. By the lemma A.1 in Appendix A, $\|N\|$ can be computed in $O(\log\log N)$ operations and word length $\log N$. Also $\lfloor \log_2 n \rfloor$ can be computed by $O(\log\log n) = O(\log\log\log N)$ operations. Now we can choose $\epsilon = 1/2^t$ and then $\lfloor \tau \log_2 n \rfloor = \lfloor \lfloor \log_2 n \rfloor / 2^{t+2} \rfloor$.

We will also need the constants $w = 2^{\max(\|x\|,\|y\|)}, 2^{\lfloor k/4 \rfloor}, 2^k, 2^{2k}$ and $4k+6$ which can be computed in $O(\log\log N)$ operations with word length $\|N\|$. By lemma 2 step (1) uses F and $\|N\|$ word length.

We change (2) and (3) by

(3′) If $x \leq 2^{2k}$ and $y \leq 2^{2k}$ then goto 9.

We now change step (4) to:

(4′) $x_1 \leftarrow x - 2^k \lfloor x/2^k \rfloor$, $\quad y_1 \leftarrow y - 2^k \lfloor y/2^k \rfloor$,

(4″) $w' = w2^{k-1}$, $\quad x_2 \leftarrow \lfloor x/w \rfloor - 2^k \lfloor x/w' \rfloor$, $\quad y_2 \leftarrow \lfloor y/w \rfloor - 2^k \lfloor y/w' \rfloor$

(4‴) If $x_2 = 0$ and $y_2 = 0$ then $\{w \leftarrow w/2^{\lfloor k/4 \rfloor}$, goto 4″$\}$.

Since $\lfloor k/4 \rfloor = O(\log\log N)$ the algorithm still has the same operation complexity. ○

In the same way we can prove

Corollary 1. *Using $F = \{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, \leftarrow, >\}$ we can compute $GCD(x,y)$ in*

$$O\left(\frac{\log N}{\log\log N}\right)$$

operations where $N = \max(x,y)$. Using $F^* = F \cup \{mod\}$ then the above bound is true for $N = \min(x,y)$.

Proof . The first bound is proved in Theorem 1. The second bound is true since we can compute first $z \leftarrow x \bmod y$ where $x \geq y$ and then compute $GCD(y,z)$. ○

5. EXTENDED GCD ALGORITHM

In this section we solve the extended GCD problem, i.e given $x, y \in [0, N]$. Find integers $a, b \in [0, N]$ such that $ax + by = GCD(x,y)$. Since our algorithm is based on Stien algorithm we first show how to extend Stein algorithm to solve the extended GCD problem.

The input of the algorithm is $x, y \in [0, N]$ and the output is a, b such that $ax + by = GCD(x,y)$.

Algorithm E

Extended_Stein(x,y,a,b)

(1) $\alpha_1 \leftarrow 1,\ \beta_1 \leftarrow 0,\ \gamma_1 \leftarrow 1,\ \alpha_2 \leftarrow 0,\ \beta_2 \leftarrow 1,\ \gamma_2 \leftarrow 1,\ d \leftarrow 1$

(2) While $2|x$ and $2|y$ do $\{x \leftarrow x/2, y \leftarrow y/2, d \leftarrow 2d\}$.

(3) While $2|x$ do $\{x \leftarrow x/2, \gamma_1 \leftarrow 2\gamma_1\}$

(4) While $2|y$ do $\{y \leftarrow y/2, \gamma_2 \leftarrow 2\gamma_2\}$

(5) If $x > y$ then

$$\left\{x \leftarrow \tfrac{x-y}{2},\ \gamma \leftarrow \max(\gamma_1,\gamma_2),\ \alpha_1 \leftarrow \tfrac{\gamma}{\gamma_1}\alpha_1 - \tfrac{\gamma}{\gamma_2}\alpha_2,\ \beta_1 \leftarrow \tfrac{\gamma}{\gamma_1}\beta_1 - \tfrac{\gamma}{\gamma_2}\beta_2,\ \gamma_1 \leftarrow 2\gamma\right\}$$

else

$$\left\{y \leftarrow \tfrac{y-x}{2},\ \gamma \leftarrow \max(\gamma_1,\gamma_2),\ \alpha_2 \leftarrow \tfrac{\gamma}{\gamma_2}\alpha_2 - \tfrac{\gamma}{\gamma_1}\alpha_1,\ \beta_2 \leftarrow \tfrac{\gamma}{\gamma_2}\beta_2 - \tfrac{\gamma}{\gamma_1}\beta_1,\ \gamma_2 \leftarrow 2\gamma\right\}$$

(6) If $y = 0$ then $\{a \leftarrow \alpha_1, b \leftarrow \beta_1, c \leftarrow \gamma_1$, goto 8$\}$

(7) Goto 3.

(8) While $c \neq 1$ do {If $2|a$ and $2|b$

then $\{a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}, c \leftarrow \frac{c}{2}\}$

else $\{a \leftarrow \frac{a+y}{2},\ b \leftarrow \frac{b-x}{2},\ c \leftarrow \frac{c}{2}\}\}$

(9) $a \leftarrow ad,\ b \leftarrow bd$

Lemma 6. *Let x, y be integers not both even. Let $\{x_i, y_i, \alpha_{j,i}, \beta_{j,i}, \gamma_{j,i}\}_{i=1,\cdots,t_1,\ j=1,2}$ be the numbers that are produced in steps (3)-(6) with $x_1 = x,\ y_1 = y,\ \alpha_{j,1} = \alpha_j,\ \beta_{j,1} = \beta_j,\ \gamma_{j,1} = \gamma_j,\ j = 1,2$, and let $\{a_i, b_i, c_i\}_{i=1,\cdots,t_2}$ be the numbers that are produced in step (8). Then*

(*i*) $t_2 < t_1 \leq \|x\| + \|y\| - 1$.

(*ii*) $x_i = \frac{\alpha_{1,i}x + \beta_{1,i}y}{\gamma_{1,i}},\ \ y_i = \frac{\alpha_{2,i}x + \beta_{2,i}y}{\gamma_{2,i}}$.

(*iii*) $GCD(x,y) = GCD(x_i, y_i)$.

(*iv*) $\gamma_i = \max(\gamma_{1,i}, \gamma_{2,i}) = 2^{i-1},\ |\alpha_{j,i}|, |\beta_{j,i}| \leq \gamma_{j,i},\ j = 1,2,\ \gamma_{2,t_1} = 2^{t_1-1} \leq xy$.

(*v*) $a_i x + b_i y = c_i\ GCD(x,y)$.

Proof . By step (3)-(5) we have $\|x_{i+1}\| + \|y_{i+1}\| \leq \|x_i\| + \|y_i\| - 1$. This follows $t_1 \leq \|x\| + \|y\| - 1$. (*ii*)-(*iii*) with the first two inequality in (*iv*) follows exactly as in the proof of lemma 3 and 4. Therefore $\gamma_{2,t_1} = 2^{t_1-1} < xy$. Since $c_1 = \gamma_{1,t_1} \leq \gamma_{2,t_1} = 2^{t_1-1}$ and since $c_{i+1} = c_i/2$ we have

$$t_2 \leq t_1 - 1.$$

This follows (*i*). Since $a_1 = \alpha_{1,t_1}, b_1 = \beta_{1,t_1},\ c_1 = \gamma_{1,t_1} \leq 2^{t_1-1}$ and $y_{t_1} = 0$ we have

$$GCD(x,y) = GCD(x_{t_1}, y_{t_1}) = \frac{\alpha_{1,t_1}x + \beta_{1,t_1}y}{\gamma_{1,t_1}} = \frac{a_1 x + b_1 y}{c_1}.$$

Therefore $a_1 x + b_1 y = c_1 GCD(x,y)$. To prove (*v*) we proceed by induction. Assume that $a_i x + b_i y = c_i GCD(x,y)$. If a_i is even and b_i is even then since c_i is a power of 2 and not 1 we have $\frac{a_i}{2}x + \frac{b_i}{2}y = \frac{c_i}{2}$ which follows $a_{i+1}x + b_{i+1}y = c_{i+1}$.

If not both a_i and b_i are even then three cases can happen. (A) If x is odd and y is even then since c_i is even we have a_i is even and therefore b_i is odd. Then $(a_i+y)x+(b_i-x)y = c_iGCD(x,y)$ and a_i+y, b_i-x are even. Dividing the equation by 2 implies $a_{i+1}x+b_{i+1}y = c_{i+1}GCD(x,y)$. (B) The case where x is even and y is odd is similar to the previous case. (C) If x and y are odd then since not both a_i and b_i are even and since c_i is even we must have a_i and b_i are odd. Then a_i+y, b_i-x are even and as before we have $a_{i+1}x+b_{i+1}y = c_{i+1}GCD(x,y)$. This follows (v). ○

To find better bounds for $\alpha_{i,j}, \beta_{i,j}, a_i, b_i$ we give a more elegant treatment. We first prove that

$$1=\alpha_{1,1}\le\alpha_{1,2}\le\cdots\le\alpha_{1,t_1}, \qquad 0=\beta_{1,1}\ge\beta_{1,2}\ge\cdots\ge\beta_{1,t_1}$$

$$0=\alpha_{2,1}\ge\alpha_{2,2}\ge\cdots\ge\alpha_{2,t_1}, \qquad 1=\beta_{2,1}\le\beta_{2,2}\le\cdots\le\beta_{2,t_1} \tag{7}$$

We give the proof by induction. The above is true for $i=1$. Now steps (3) and (4) do not change $\alpha_{i,j}, \beta_{i,j}$. In the first step of (5) we have:

$$\alpha_{1,i+1} = \frac{\gamma}{\gamma_{1,i}}\alpha_{1,i} - \frac{\gamma}{\gamma_{2,i}}\alpha_{2,i} \ge \alpha_{1,i}$$

because $\gamma/\gamma_{1,i}, \gamma/\gamma_{2,i}$ are 1 and power of 2 and $\alpha_{2,i}\le 0$. The other follows in a similar manner.

Since, the last command that is executed in the algorithm in steps (3)-(5) is the command after the **else** in step (5) we have

$$\alpha_{2,t_1} = \frac{\gamma}{\gamma_{2,t_1-1}}\alpha_{2,t_1-1} - \frac{\gamma}{\gamma_{1,t_1-1}}\alpha_{1,t_1-1} \le -\alpha_{1,t_1-1} = -\alpha_{1,t_1}.$$

Doing this also for β_{2,t_1} we obtain

$$\alpha_{2,t_1}\le -\alpha_{1,t_1}, \qquad \beta_{2,t_1}\ge -\beta_{1,t_1}. \tag{8}$$

By lemma 6 and since $y_{t_1}=0$, $x_{t_1}=GCD(x,y)$ we have

$$\alpha_{1,t_1}x+\beta_{1,t_1}y = c_1GCD(x,y), \qquad \alpha_{2,t_1}x+\beta_{2,t_1}y = 0$$

By steps (3)-(4) we have $\binom{x_{i+1}}{y_{i+1}} = T\binom{x_i}{y_i}$ where

$$T\in\mathcal{A} = \left\{\begin{pmatrix}1/2 & -1/2\\ 0 & 1\end{pmatrix}, \begin{pmatrix}1 & 0\\ -1/2 & 1/2\end{pmatrix}, \begin{pmatrix}1/2 & 0\\ 0 & 1\end{pmatrix}, \begin{pmatrix}1 & 0\\ 0 & 1/2\end{pmatrix}\right\}$$

and therefore for $\gamma = 2^{t_1-1}/c_1$ we have

$$\begin{pmatrix}GCD(x,y)\\ 0\end{pmatrix} = \begin{pmatrix}\gamma\frac{\alpha_{1,t_1}}{2^{t_1-1}} & \gamma\frac{\beta_{1,t_1}}{2^{t_1-1}}\\ \frac{\alpha_{2,t_1}}{2^{t_1-1}} & \frac{\beta_{2,t_1}}{2^{t_1-1}}\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} \triangleq A\begin{pmatrix}x\\ y\end{pmatrix}$$

where A is a multiplication of $A_{t_1-1}A_{t_1-2}\cdots A_1$, $A_1\in\mathcal{A}$. Since $det(A_{t_1-1}\cdots A_1) = \frac{1}{2^{t_1-1}}$ we have $\alpha_{1,t_1}\beta_{2,t_1} - \beta_{1,t_1}\alpha_{2,t_1} = 2^{t_1-1}/\gamma$. Then

$$\begin{pmatrix}x\\ y\end{pmatrix} = \begin{pmatrix}\beta_{2,t_1} & -\gamma\beta_{1,t_1}\\ -\alpha_{2,t_1} & \gamma\alpha_{1,t_1}\end{pmatrix}\begin{pmatrix}GCD(x,y)\\ 0\end{pmatrix}.$$

which implies

$$\alpha_{2,t_1} = -\frac{y}{GCD(x,y)}, \qquad \beta_{2,t_1} = \frac{x}{GCD(x,y)}.$$

and by (8) we have

$$1 \le \alpha_{1,t_1} = a_1 \le \frac{y}{GCD(x,y)}, \qquad 0 \le -\beta_{1,t_1} = -b_1 \le \frac{x}{GCD(x,y)}, \tag{9}$$

We now prove that

$$0 \le a_i \le y, \qquad 0 \le -b_i \le x. \tag{10}$$

By induction. Assuming (10) is true for i, we have $0 \le a_{i+1} = \frac{a_i+y}{2} \le y$ or $0 \le a_{i+1} = \frac{a_i}{2} \le y$ and $0 \le -b_{i+1} = \frac{-b_i+x}{2} \le x$ or $0 \le -b_{i+1} = \frac{-b_i}{2} \le x$. We proved

Lemma 7. *We have*

(*i*) $1 = \alpha_{1,1} \le \alpha_{1,2} \le \cdots \le \alpha_{1,t_1} = a_1 \le \frac{y}{GCD(x,y)}, \quad 0 = \alpha_{2,1} \ge \alpha_{2,2} \ge \cdots \ge \alpha_{2,t_1} = -\frac{y}{GCD(x,y)},$

(*ii*) $1 = \beta_{2,1} \le \beta_{2,2} \le \cdots \le \beta_{2,t_1} = b_1 \le \frac{x}{GCD(x,y)}, \quad 0 = \beta_{2,1} \ge \beta_{2,2} \ge \cdots \ge \beta_{2,t_1} = -\frac{x}{GCD(x,y)},$

(*iii*) $0 \le a_i \le y, \quad 0 \ge b_i \ge -x.$

If we execute the algorithm in the case where $y = 2^n$ and $x = 2^n + 2^{n-1} + \cdots + 1$ then it can be easily shown that $\gamma_{1,t_1} = y^2$. This implies that γ_{1,t_1} is not bounded by $O(\max(x,y))$. To make the algorithm of $||N||$ word length, we have to handle $\gamma_{j,i}$. To do this, instead of saving $\gamma_{j,i}$ we save $\gamma_i^{(1)} = \gamma_i/\gamma_{1,i}$, $\gamma_i^{(2)} = \gamma_i/\gamma_{2,i}$ and t_1. The algorithm becomes

Algorithm E′

Extended_Stein(x,y,a,b)

(1) $\alpha_1 \leftarrow 1,\ \beta_1 \leftarrow 0,\ \gamma^{(1)} \leftarrow 1,\ \alpha_2 \leftarrow 0,\ \beta_2 \leftarrow 1,\ \gamma^{(2)} \leftarrow 1,\ d \leftarrow 1,\ t_1 \leftarrow 0,\ t_2 \leftarrow 0$

(2) While $2|x$ and $2|y$ do $\{x \leftarrow x/2, y \leftarrow y/2, d \leftarrow 2d\}$

(3) While $2|x$ do $\{x \leftarrow x/2, \gamma^{(2)} \leftarrow 2\gamma^{(2)},\ t_1 \leftarrow t_1+1,\ h \leftarrow 2\}$

(4) While $2|y$ do $\{y \leftarrow y/2, \gamma^{(1)} \leftarrow 2\gamma^{(1)},\ t_2 \leftarrow t_2+1,\ h \leftarrow 1\}$

(5) If $x > y$ then

$\{x \leftarrow \frac{x-y}{2},\ \alpha_1 \leftarrow \gamma^{(1)}\alpha_1 - \gamma^{(2)}\alpha_2,\ \beta_1 \leftarrow \gamma^{(1)}\beta_1 - \gamma^{(2)}\beta_2,\ t_1 \leftarrow \max(t_1,t_2)+1,\ \gamma^{(2)} \leftarrow 2\gamma^{(2)}$

If $h \neq 2$ then $\{\gamma^{(1)} \leftarrow 1,\ \gamma^{(2)} \leftarrow 2,\ h \leftarrow 2\}\}$

else

$\{y \leftarrow \frac{y-x}{2},\ \alpha_2 \leftarrow \gamma^{(2)}\alpha_2 - \gamma^{(1)}\alpha_1,\ \beta_2 \leftarrow \gamma^{(2)}\beta_2 - \gamma^{(1)}\beta_1,\ t_2 \leftarrow \max(t_1,t_2)+1,\ \gamma^{(1)} \leftarrow 2\gamma^{(1)}$

If $h \neq 1$ then $\{\gamma^{(2)} \leftarrow 1,\ \gamma^{(1)} \leftarrow 2,\ h \leftarrow 1\}\}$

(6) If $y = 0$ then $\{a \leftarrow \alpha_1, b \leftarrow \beta_1,\ t \leftarrow t_1$, goto 8$\}$

(7) Goto 3.

(8) For $i := 1$ to t do {If $2|a$ and $2|b$

then $\{a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}\}$

else $\{a \leftarrow \frac{a+y}{2},\ b \leftarrow \frac{b-x}{2}\}\}$

(9) $a \leftarrow ad,\ b \leftarrow bd$

We now have

Lemma 8. *Let* $\{x_i,y_i,\alpha_{j,i},\beta_{j,i},\gamma_{j,i}\}_{i=1,\cdots,t,\ j=1,2}$, $\{a_i,b_i,c_i\}_{i=1,\cdots,t'}$ *as in lemma 6. Let* $\{x'_i,y'_i,\alpha'_{j,i},\beta'_{j,i},\gamma'_{j,i},h_i,t_{1,i},t_{2,i}\}_{i=1,\cdots,T}$, $\{a'_i,b'_i,t_i\}_{i=1,\cdots,T'}$ *be the numbers that produced in steps (3)-(9) of Algorithm E′. Then*

(*i*) $x'_i = x_i,\ y'_i = y_i,\ \alpha'_{1,i} = \alpha_{1,i},\ \beta'_{1,i} = \beta_{1,i},\ t = T,\ t' = T'$.

(*ii*) $\gamma_{1,i} = 2^{t_{1,i}},\ \gamma_{2,i} = 2^{t_{2,i}},\ \gamma_i^{(1)} = \max(\gamma_{1,i},\gamma_{2,i})/\gamma_{1,i},\ \gamma_i^{(2)} = \max(\gamma_{1,i},\gamma_{2,i})/\gamma_{2,i}$

(*iii*) $a'_i = a_i,\ b'_i = b_i,\ c'_i = 2^{t_i}$.

(*iv*) $\gamma_i^{(1)} \leq y\ ,\gamma_i^{(2)} \leq x$.

The proof of lemma 8 is straightforward and will be omitted.

This lemma with lemma 7 and lemma A.2 in Appendix A follows

Theorem 2. *Using* $F = \{\{1\},+,-,Shi_{\{1,-1\}},>\}$ *we can compute* $GCD(x,y)$, $x,y \in [0,N]$ *and* $a,b \in [0,N]$ *such that* $ax+by = GCD(x,y)$ *by*

$$O(\log(N))$$

operations with $||\max(x,y)||$ *word length and space* $O(1)$.

To prove Theorem v we need two algorithms. The first algorithm take as an input $x,y \in [0,N]$ and output α,β,t such that $\alpha x+\beta y = 2^t GCD(x,y)$ and the second take α,β,t as an input and output a,b such

that $ax+by = GCD(x,y)$. Both algorithms must has operation complexity $O\left(\frac{\log N}{\log\log N}\right)$. The first algorithm can be easily obtain from algorithms B,C and D by saving the sequences $\alpha_{i,j}(x_1x_2y_1y_2), \beta_{i,j}(x_1x_2y_1y_2)$ and $t_j = \log\gamma_{i,j}(x_1x_2y_1y_2)$ with the changes made in Algorithm E'. We need only the following

Lemma 9. *Let $\{\alpha_j,\beta_j,t_j\}_{j=1,2}$ be integers and $\gamma^{(j)} = \max(2^{t_1},2^{t_2})/2^{t_j}$. Denote by*

$$L[\alpha_1,\beta_1,t_1,\gamma^{(1)},\alpha_2,\beta_2,t_2,\gamma^{(2)}](x,y) = \left(\frac{\alpha_1x+\beta_1y}{2^{t_1}}, \frac{\alpha_2x+\beta_2y}{2^{t_2}}\right)$$

Then

$$L[\alpha_1,\beta_1,t_1,\gamma^{(1)},\alpha_2,\beta_2,t_2,\gamma^{(2)}] \circ L[\alpha_3,\beta_3,t_3,\gamma^{(3)},\alpha_4,\beta_4,t_4,\gamma^{(4)}] =$$

$$L[\gamma^{(3)}\alpha_1\alpha_3+\gamma^{(4)}\beta_1\alpha_4,\ \gamma^{(3)}\alpha_1\beta_3+\gamma^{(4)}\beta_1\beta_4,\ t_1+\max(t_3,t_4),\ \gamma^{(1)},$$

$$\gamma^{(3)}\alpha_2\alpha_3+\gamma^{(4)}\beta_3\alpha_4,\ \gamma^{(3)}\alpha_2\beta_3+\gamma^{(4)}\beta_2\beta_4,\ t_2+\max(t_3,t_4),\ \gamma^{(2)}]$$

Proof . Straightforward. ○

This lemma follows

Lemma 10. *Let $x,y \in [0,N]$. Using $F = \{\{1\},+,-,\times_{\mathbf{Z}},\lfloor/2^i\rfloor,\leftarrow,>\}$ we can compute $\alpha,\beta \in [0,N]$ and $t \in [0,2\log N]$ such that*

$$\alpha x+\beta y = 2^tGCD(x,y)$$

with operation complexity $O\left(\frac{\log N}{\log\log N}\right)$, word length $\|N\|$ and space $O((\log N)^\epsilon)$ for any constant $0<\epsilon<1$.

We now give the second algorithm that execute step (5) in Algorithm E' by $O\left(\frac{\log N}{\log\log N}\right)$ operations.

We first use the following algorithm

Algorithm F

Let x_1,y_1,a_1,b_1 be integers such that $\|x_1\|,\|y_1\|,\|a_1\|,\|b_1\| \le k$.

Normalize_First$(x_1,y_1,a_1,b_1,\delta,\psi)$

(1) $\delta \leftarrow 0,\ \psi \leftarrow 1$

(2) For $i := 1$ to k do

{ If $2|a_1$ and $2|b_1$ then $\{a_1 \leftarrow \frac{a_1}{2},\ b_1 \leftarrow \frac{b_1}{2},\ \psi \leftarrow 2\psi\}$

else $\{a_1 \leftarrow \frac{a_1+y_1}{2},\ b_1 \leftarrow \frac{b_1+x_1}{2},\ \delta \leftarrow \delta+\psi,\ \psi \leftarrow 2\psi\}\}$

Now we have the following

Lemma 11. *Let $ax-by = 2^tGCD(x,y)$, $a,b,x,y \ge 0$ and*

$$a_1 = [a]_{1,k},\quad b_1 = [b]_{1,k},\quad x_1 = [x]_{1,k},\quad y_1 = [y]_{1,k} \qquad (*)$$

and let $\{a_{1,i},b_{1,i},x_{1,i},y_{1,i},\delta_i,\psi_i\}_{i=1\cdots,k}$ be the numbers produces in algorithm F. Then

$$\frac{a+\delta_iy}{\psi_i}, \frac{b+\delta_ix}{\psi_i} \le \max(x,y)$$

are integers and

$$a_i x - b_i y \equiv \left(\frac{a+\delta_i y}{\psi_i}\right) x - \left(\frac{b+\delta_i x}{\psi_i}\right) y = 2^{t-i} GCD(x,y) \qquad (**)$$

Proof . From (*) it follows that the commands that are executed for a_1, b_1, x_1, y_1 are the same commands that will be executed if we replace a_1, b_1, x_1, y_1 by a, b, x, y. Now assuming (**) is true for i it can be easily prove by lemma 6 that is also true for $i+1$. ○

Now we give the algorithm that take as an input a, b, t such that $ax - by = 2^t GCD(x,y)$ and output $A, B \in [0, N]$ such that $Ax + By = GCD(x,y)$.

Let $0 < \epsilon < 1$, $\lambda = \frac{\epsilon}{4}$, $k = \lfloor \lambda \log\log N \rfloor$, $s_1 = \lfloor t/k \rfloor$ and $s_2 = t \bmod k$.

Algorithm G

Normalize(a, b, t, A, B)

(1) For $i := 1$ to s_1 do

$\{ x_1 \leftarrow [x]_{1,k},\ y_1 \leftarrow [y]_{1,k},\ a_1 \leftarrow [a]_{1,k},\ b_1 \leftarrow [b]_{1,k}$

If $\delta(x_1 y_1 a_1 b_1) = nill$ then $\{$*Normalize_First*$(x_1, y_1, a_1, b_1, \delta, \psi)$

$\delta(x_1 y_1 a_1 b_1) \leftarrow \delta,\ \ \psi(x_1 y_1 a_1 b_1) \leftarrow \psi\}$

$a \leftarrow \frac{a+\delta(x_1 y_1 a_1 b_1) y}{\psi(x_1 y_1 a_1 b_1)},\ \ b \leftarrow \frac{b+\delta(x_1 y_1 a_1 b_1) x}{\psi(x_1 y_1 a_1 b_1)}$ $\}$

(2) For $i := 1$ to s_2 do {If $2|a$ and $2|b$

then $\{a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}\}$

else $\{a \leftarrow \frac{a+y}{2},\ b \leftarrow \frac{b+x}{2}\}\}$

(3) $A \leftarrow a,\ B \leftarrow b$

The space needed for this algorithm is $2 \times 2^{4k} \leq (\log N)^\epsilon$. Since $s_1 = O\left(\frac{\log N}{\log\log N}\right)$, $s_2 = O(\log\log N)$ the operation complexity of the algorithm is $O\left(\frac{\log N}{\log\log N}\right)$. The constants $\psi(x_1 y_1 a_1 b_1)$, $\delta(x_1 y_1 a_1 b_1)$ are bounded by $2^k = O((\log N)^{\epsilon/4})$. Therefore by lemma A.2 the algorithm has word length $||N||$. This follows

Theorem 3. *Let $0 < \epsilon < 1$ be a constant. Using $F = \{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, \leftarrow, >\}$, we can compute $a, b \in [0, N]$ such that $ax + by = GCD(x,y)$ by*

$$O\left(\frac{\log N}{\log\log N}\right)$$

operations with $||N||$ word length and space $O((\log N)^\epsilon)$.

6. LOWER BOUNDS

In this section we prove lower bounds for GCD for several RAM models.

Consider the following operations:

$+,-,\times,/$	Arithmetic operations on $\mathbf{Q}$
$\lfloor\ \rfloor$, *mod*	Floor and modulu operations
$\wedge,\vee,\sim,\otimes$	Bitwise integer boolean operations
Rot_r, Shi_r	Rotate and shift r times
$log_r, \|\ \|$	$\lfloor \log_r x \rfloor$. $\|\|x\|\|$ is the length of the binary representation of x
$\circ_G$	$\circ$ operation with second operand from G
$\lfloor /G \rfloor$	Integer division by $g \in G$
$\leftarrow$	Indirect addressing
$>$	Comparisons $<,>,=$
$=$	Equation $=$
E	Any YES/NO question
$G[\alpha_1,\cdots,\alpha_v]$	Constants from G depends on $\alpha_1,\cdots,\alpha_v$.

$+,-,\times,/$ are the arithmetic operations on $\mathbf{Q}$. $\lfloor x \rfloor$ is the greatest integer that is not greater than x and $x \bmod y = x - y\lfloor x/y \rfloor$. For two integers x and y the boolean operations $x \wedge y$, $x \vee y$, $x \otimes y$ are bitwise and, or, xor , respectively, of the binary representation of x and y. The complement $\sim(x)$ is $1-x_n|1-x_{n+1}|\cdots|1-x_1$ where $x_n|x_{n-1}|\cdots|x_1$ is the binary representation of x. $Shi_r(x)$ is $\lfloor x/2^r \rfloor$, i.e shift r bits to the right if $r > 0$ or $-r$ bits to the left if $r < 0$. When the RAM is of W word length and $A = x_w|x_{w-1}|\cdots|x_1$

$$Shi_r(A) = \begin{cases} 0|\cdots|0|x_W|x_{W-1}|\cdots|x_{r+2}|x_{r+1} & r \geq 0 \\ x_{W-r}|x_{W-r-1}|\cdots|x_1|0|\cdots|0 & r < 0 \end{cases}$$

and

$$Rot_r(A) = \begin{cases} x_r|x_{r-1}|\cdots|x_0|x_W|x_{W-1}|\cdots|x_{r+1} & r \geq 0 \\ x_{W-r}|x_{W-r-1}|\cdots|x_1|x_W|x_{W-1}|\cdots|x_{W-r+1} & r < 0 \end{cases}.$$

The operation $\circ_G$ is $x \circ g$ where $g \in G$. A $RAM(F)$ with $= \in F$ can answer question $A = 0?$. If $> \in F$ then the model can answer comparison questions $A > 0?$, $A = 0?$, $\cdots$ and when $E \in F$ then the RAM has unlimited power of answering YES/NO questions. When $G[\alpha_1,\cdots,\alpha_v] \in F$ then any information we need for the elements of G and the elements $\alpha_1,\cdots,\alpha_v$ is known and can be used in the computation. For example computing $GCD(x,y)$ when $\mathbf{Q}[y] \in F$ means computing $GCD(x,y)$ when we can use the elements of $\mathbf{Q}$ and also any information on y. (All primes that divides y, any function $f(y)$, etc.)

Let $L \subseteq \mathbf{Q}^n$. We say that the program P *recognize* L if P compute the function $\chi_L : \mathbf{Q}^n \to \{0,1\}$ where $\chi_L(x) = 1$ if and only if $x \in L$. In [B3] we proved the following results for infinity sets L

Lemma 12. *Let $|L| = \infty$. For any $RAM(F)$ with finite number of operation $|F| \leq \infty$ and finite number of constants $|C| \leq \infty$ such that $Comp_F(\chi_L)$ is finite, there exist finite subset $L' \subset L$ such that*

$$Comp_F(\chi_L) = Comp_F(\chi_{L'}).$$

The connection between computing a function f and recognizing a set L is given in the following

Lemma 13. *For any set $A \subset D$ where D is the image of a function f we have*

$$Comp_F(f) \geq Comp_F(\chi_{f^{-1}(A)}) - Comp_F(\chi_A).$$

Proof . A program that compute $f(x)$ can be extended by $Comp_F(\chi_A)$ operations steps to verify if $f(x) \in A$. This follows the result. ○

Definition 1. Let t be an integer and let $F[[H]]$ be all the functions that can be computed by H operations in the model $RAM(F)$ without indirect addressing.

Lemma 14. *If $RAM(F[[O(H)]], \leftarrow)$ recognize a language L in t steps then $RAM(F)$ can recognize L in $O(t(\log t + H))$ steps.*

Proof . Indirect addressing in RAM can be organized as a binary search in a set of upto t values currently stored by the RAM. Therefore each access to the memory can be organized as $O(\log t)$ operations without indirect addressing and each operation in $F[[O(H)]]$ can be performed as $O(H)$ operation in F. Therefore, each step in $RAM(F[[O(H)]], \leftarrow)$ can be simulated by $O(t(\log t + H))$ steps in $RAM(F)$. ○

For a program P we denote by $T(P)$ the computation tree correspond to P. We refer the reader who are not familiar with computation trees to [B], [M2] and [Y]. Any computation tree that recognize $L \subseteq \mathbf{Q}^n$ can be extended with a one operation that makes the functions in the leaves constants 0 and 1. This can be done by just add to each leaf v of $T(P)$ the question vertex: IS $f_v = 0$? where f_v is the function that is computed in v and add two new leaves for v a left leaf and a right leaf with constants function 0 and 1, respectively. We call a leaf with constant 1 an *accepted* leaf. All over the paper we assume that the computation trees that recognize a set L has constants 0 and 1 in their leaves.

Lemma 15. *Let $L \subseteq \mathbf{Z}$, $|L| = N$. If L contains no arithmetic progression of length $k+1$ then each program P in*

$$RAM(G = \mathbf{Q}[N], +, -, \times_G, /_G, \lfloor\ \rfloor, F[[O(\log\log N + \log k)]], \leftarrow)$$

that recognize L has operation complexity

$$\Omega\left(\frac{\log N}{\log\log N + \log k}\right).$$

This lower bound is also true even if we do not count the operations $\{+, -, \times_G, /_G\}$.

Allover the paper F denotes the set of operations of the RAM without indirect addressing.

Proof . Let P be a program that recognize L with operation complexity t. By lemma 14 there exist a program P' in $RAM(G = \mathbf{Q}[N], +, -, \times_G, /_G, \lfloor\ \rfloor)$ that recognize L with operation complexity $O(t(\log t + \log\log N + \log k))$. Let $T(P)$ be a computation tree that recognize L in $O(t(\log t + \log\log N + \log k))$ operations. Let v be an accepted leaf in $T(P)$ and $C(v)$ be the set of all inputs that arrive at v. By the results of Just-Meyer-Wigderson in [JMW] we have

$$|C(v)| \leq k^{t+1} t^{O(t)}. \tag{11}$$

As T has at most $2^{O(t(\log t + \log\log N + \log k))}$ accepted leafs, L has at most

$$2^{O(t(\log t + \log\log N + \log k))} k^{t+1} t^{O(t)} = k^{t+1} t^{O(t)} (k \log N)^{O(t)} \tag{12}$$

elements, which implies that $N = |L| \leq k^{t+1}t^{O(t)}(k \log N)^{O(t)}$. Solving this inequality proves the theorem.

In [JMW] it is shown that (11) depends only on the number of operation $\lfloor\ \rfloor$ and (12) is depend only on the operations $\{\leftarrow, >\}$. This follows that the bound is also true even if we do not count the operations $\{+, -, \times_G, /_G\}$. ○

For the proof of our first main theorem we need the following lemma from the elementary number theory

Lemma 16. *Let p_k denote the $k-th$ prime number. There exist constants c, c' such that*

$$c'k \log k \leq p_k \leq ck \log k.$$

We now prove Theorem ii

Theorem 4. *Let*

$$L = \{(x, y) |\ x, y \leq N \quad \textit{are relatively primes}\ \}.$$

Then any program in

$$RAM(G = \mathbf{Q}[y, N], +, -, \times_G, /_G, \lfloor\ \rfloor, F[[O(\log \log N)]], \leftarrow, >)$$

that recognize L has operation complexity

$$\Omega\left(\frac{\log N}{\log \log N}\right).$$

Proof . Let $P = P_1, \cdots, P_t$ be a minimal program that recognize L. Let $p_1, p_2, \cdots$ be the prime numbers $2, 3, 5, \cdots$, respectively, and let

$$M = p_1 \cdots p_k, \quad M \leq N, \quad Mp_{k+1} > N. \tag{13}$$

By adding the condition $x, y \leq M$ in the begging of P we make the program recognize

$$L' = \{(x, y) |\ x, y \leq M \text{ are relatively primes }\}.$$

We first estimate k, p_k and M asymptoticaly. We have $k! \leq p_1 \cdots p_k = M \leq N$ which implies that for some constant c_1

$$k \leq c_1 \frac{\log N}{\log \log N} \tag{14}$$

and therefore by lemma 16 there exist a constant c_2 such that

$$p_k \leq c_2 \log N. \tag{15}$$

Now we find lower bound for k. There exist a constant c' such that

$$(c'(k+1)\log(k+1))^{k+1} \geq p_{k+1}^{k+1} \geq p_1 \cdots p_{k+1} > N$$

which implies that there exist a constant c_3 such that

$$k > c_3 \frac{\log N}{\log \log N}. \tag{16}$$

To estimate M we have by (13)

$$M = p_1 \cdots p_k \geq k! > \left(c_4 \frac{\log N}{\log \log N} \right)^{c_4 \frac{\log N}{\log \log N}},$$

for some constant c_4. Also we need

$$\phi(M) = (p_1 - 1) \cdots (p_k - 1) \geq (k-1)! \geq \left(c_5 \frac{\log N}{\log \log N} \right)^{c_5 \frac{\log N}{\log \log N}}.$$

We now substitute in the algorithm P, $y = M$ and obtain an algorithm that recognize

$$\hat{L} = \{x \leq M \mid GCD(x, M) = 1\}$$

by

$$RAM(G = \mathbf{N}[N], +, -, \times_G, /_G, \lfloor\ \rfloor, F[[O(\log \log(N))]], \leftarrow, >).$$

Here we use lemma 15. Let $A = \{a + \lambda b \mid \lambda = 0, \cdots, l,\ a, b \in \mathbf{Z}\} \subset \hat{L}$ be an arithmetic progression of length $l+1$ in $\hat{L}$. Then we have $GCD(a + \lambda b, M) = 1$, $\lambda = 0, \cdots, l$. If $p_i | b$ for every i then $b = M$ and $a \leq 0$ which implies $|A| \leq 1$.If there exist $p_{i_0} \not| b$ then since the equation $a + \lambda b \equiv 0 \ mod\ p_{i_0}$ has a solution $\lambda_0 = (-a)b^{-1} \ mod\ p_{i_0} < p_{i_0}$ we have $p_{i_0} | GCD(a + \lambda_0 b, M)$. Therefore by (15)

$$l + 1 < p_{i_0} \leq p_k < c_2 \log N.$$

Then by lemma 15 and since $|\hat{L}| = \phi(M)$ we have that the program P has operation complexity

$$\Omega\left(\frac{\log |\hat{L}|}{\log (c_2 \log N) + \log \log |\hat{L}|} \right) = \Omega\left(\frac{\log N}{\log \log N} \right). \quad \bigcirc$$

Notice that by lemma 13 with $A = \{1\}$ this lower bound is also true for computing GCD.

The next theorem shows that if we give a resonable bound on the word length then even if we assume that log and $||\ ||$ (and many other function growth slowly with x) can be computed by 1 operation complexity we still have the same lower bound.

Theorem 5. *Using* $\{G = \mathbf{Q}[N, W, y], +, -, \times_G, /_G, \lfloor\ \rfloor, Shi_G, Rot_G, \sim, \log_{\mathbf{N}}, ||\ ||, \leftarrow, >\}$ *with word length* $W = poly(\log N)$, *any program that recognize*

$$L = \{(x, y) | x, y \in [0, N],\ \ x \text{ and } y \text{ are relatively primes } \}$$

requires

$$\Omega\left(\frac{\log N}{\log \log N} \right)$$

operations.

Proof . As in the proof of Theorem 4 we make a reduction to $\hat{L}$. We first simulate $Shi_G, Rot_G, \sim$ by $O(1)$ operation complexity from $RAM(\mathbf{Q}[N,W],+,-,\times_G,/_G,\lfloor\ \rfloor,F[[O(\log\log N]],\leftarrow,>)$. We have

$$Shi_r(A) = \begin{cases} \lfloor \frac{A}{2^r} \rfloor & r \geq 0 \\ 2^{-r}\left(A - 2^{W+r}\lfloor \frac{A}{2^{W+r}} \rfloor\right) & r \leq 0 \end{cases}$$

$$Rot_r(A) = \begin{cases} \lfloor \frac{A}{2^r} \rfloor + 2^{W-r}\left(A - 2^r \lfloor \frac{A}{2^r} \rfloor\right) & r \geq 0 \\ \lfloor \frac{A}{2^{W+r}} \rfloor + 2^{-r}\left(A - 2^{W+r}\lfloor \frac{A}{2^{W+r}} \rfloor\right) & r < 0 \end{cases}$$

and $\sim(A) = 2^W - 1 - A$.

We now prove $\log_{\mathbf{N}}, \|\ \| \in F[[O(\log\log N)]]$. We have: If $W < (\log N)^\delta$ then

$$\lfloor \log_a x \rfloor = \begin{cases} 0 & 1 \leq x < a \\ 1 & a \leq x < a^2 \\ \vdots & \vdots \\ \lfloor (\log_a 2)(\log N)^\delta \rfloor & 2^{\lfloor (\log_a 2)(\log N)^\delta \rfloor} \leq x < 2^{(\log N)^\delta} \end{cases}$$

Now, to find the domain where x is belong to, we need a program with operation complexity (binary search)

$$O\left(\log\left(\lfloor (\log_a 2)(\log N)^\delta \rfloor\right)\right) = O(\log\log N).$$

Therefore, $\log_{\mathbf{N}} \in F[[O(\log\log N]]$. Now since $\|x\| = \lfloor \log_2 x \rfloor + 1$ we have also $\|\ \| \in F[[O(\log\log N]]$. By theorem 4 the result follows. ○

Before we proceed we give some preliminary results from [B5]

Definition 2. Let $\{n_1,\cdots,n_k,m_1,\cdots,m_j\} \subset \mathbf{Q}$ be set of rational numbers and F be a set of operations. We denote by

$$\Delta_{F,S}(\{n_1,\cdots,n_k\},\{m_1,\cdots,m_j\})$$

the number of operations in F needed to compute the constants $m_1,\cdots,m_j$ using the constants $n_1,\cdots,n_k$ and space S.

Our main result in [B5] is

Lemma 17. *Let $f : D \to \mathbf{Q}^j$ be a function where $D \subseteq \mathbf{Q}^k$. Then the operation complexity of f using the constants C and space S is*

$$Comp_{F\cup\{\leftarrow,E\}}(f) \geq \max_{(n_1,\cdots,n_k)\in D} \Delta_{F,S}(\{n_1,\cdots,n_k\} \cup C, f(n_1,\cdots,n_k))$$

Lemma 18. *Let C and F be a set of constants and a set of operations, respectively. Let T be a set of constants. If*

$$|F| + |C| \leq \min(S, poly(\log|T|))$$

then there exist $t \in T$ such that

$$\Delta_{F,S}(C,t) \geq \Omega\left(\frac{\log|T|}{\min(\log S, \log\log|T|)}\right).$$

Proof . We can assume that the constants in C are constant functions in F. This assumption does not change the order of the operation complexity. We now count the number of programs of length l. The $i-th$ step in the program can be $M[i_1] \leftarrow M[i_2] \circ M[i_3]$ where $\circ \in F$. We have $i_1, i_2, i_3 \in \{1, \cdots, S\}$ if $l > S$, or i_1, i_2, i_3 is chosen from l locations in the memory if $S < l$. Therefore the number of programs of length less than or equal to l are at most $|F|^l(\min(S,l))^{3l}$. This number must be greater that $|T|$, i.e

$$|F|^l(\min(S,l))^{3l} \geq |T|$$

which implies that there exist $t \in T$ such that

$$\Delta_{F,S}(C,t) \geq l \geq \Omega\left(\frac{\log|T|}{\min(\log S, \log\log|T|)}\right). \quad \bigcirc$$

Theorem 6. *Using* $\{C, +, -, \times, \wedge, \vee, \otimes, \sim, Shi_1, \leftarrow, E\}$, $|C| < \min(S, poly(\log N))$ *and space* S, *any program that computes* $GCD(x,y)$, $x, y \in [0, N]$ *requires*

$$\Omega\left(\frac{\log N}{\min(\log S, \log\log N)}\right)$$

operations.

Therefore

(*1*) *If* $S = O(1)$ *then we have* $\Omega(\log N)$ *lower bound for computing GCD.*

(*2*) *To obtain the* $\Omega\left(\frac{\log N}{\log\log N}\right)$ *lower bound we must use at least* $(\log N)^\epsilon$ *space for some constant* ϵ.

Proof . Let

$$T = \{t|\ \ t \leq N^{\frac{1}{4}}, \qquad t\ odd\}, \qquad n = \left\lfloor \frac{1}{3}\log N \right\rfloor.$$

By lemma 18 there exist $A \in T$ such that

$$\Delta_{F,S}(\{2^{n-1}-1, 2^{n-2}\} \cup C \bmod 2^{n-1}, \{A\}) \geq \Omega\left(\frac{\log N}{\min(\log S, \log\log N)}\right), \tag{17}$$

where $F = \{+, -, \times, \wedge, \vee, \otimes, \sim, Shi_1\}$. Here $C \bmod 2^{n-1} = \{c \bmod 2^{n-1} | c \in C\}$.

Let $1 \leq b < A$ be an integer such that

$$b2^n \equiv 1(\bmod A).$$

Then $b2^n - 1 < AN^{1/3} \leq N$. Let

$$a_1 = b2^n - 1,\ a_2 = A2^n.$$

We have $a_2 < AN^{1/3} \leq N$ and

$$GCD(a_1, a_2) = GCD(b2^n - 1, 2^n A) = GCD(b2^n - 1, A) = A.$$

By lemma 17 the operation complexity of computing $GCD(a,b)$, $a, b \in [0, N]$ is greater than

$$\Delta_{F,S}(\{b2^n - 1, A2^n\} \cup C, \{A\}).$$

Let P be a minimal program that computes A from the constants $\{b2^n-1, A2^n\}\cup C$ using the operations in F. We now define a new program P' that computes each number in P modulo 2^{n-1}.

Given $D' = D \bmod 2^{n-1}$ and $B' = B \bmod 2^{n-1}$ then ($L :== L'$ means L in P is converted to L' in P')

(1) $(D+B) \bmod 2^{n-1} :== (D'+B') \wedge (2^{n-1}-1)$.

(2) $(D-B) \bmod 2^{n-1} :==$ if $D' \geq B'$ then we put $(D'-B') \wedge (2^{n-1}-1)$ otherwise we put $(B'-D') \wedge (2^{n-1}-1)$.

(3) $(D \times B) \bmod 2^{n-1} :== (D' \times B') \wedge (2^{n-1}-1)$.

(4) $(D \wedge B) \bmod 2^{n-1} :== (D' \wedge B')$.

(5) $(D \vee B) \bmod 2^{n-1} :== (D' \vee B')$.

(6) $(D \otimes B) \bmod 2^{n-1} :== (D' \otimes B')$.

(7) $\sim (D) \bmod 2^{n-1} :== \sim (D')$.

(8) $(Shi_1\ D) \bmod 2^{n-1} :==$if the n bit of D is 0 then we write $Shi_1(D')$, otherwise we write $Shi_1(D')+2^{n-2}$.

Here we compute each number in the algorithm P modulo 2^{n-1} by the operations F and the constants $\{2^{n-2}, 2^{n-1}-1\}$. Since the algorithm P uses the constants $\{b2^n-1, A2^n\} \cup C$ as an initial constants and computes A, the new algorithm P' uses

$$(\{b2^n-1, A2^n\} \cup \{2^{n-2}, 2^{n-1}-1\} \cup C) \bmod 2^{n-1} = \{2^{n-1}-1, 2^{n-2}\} \cup C \bmod 2^{n-1}$$

as an initial constants and computes $A \bmod 2^{n-1} = A$. By (17) we have for some constant

$$Comp_{F\cup\{\leftarrow,E\}}(GCD) \geq Comp_F(P) \geq cComp_F(P') \geq$$

$$\Delta_{F,S}(\{2^{n-1}-1, 2^n-2\} \cup C \bmod 2^{n-1}, \{A\}) \geq \Omega\left(\frac{\log N}{\min(\log S, \log\log N)}\right). \quad \bigcirc$$

This lemma follows

Corollary 2. *Using* $\{C, +, -, \times, \wedge, \vee, \otimes, \sim, Shi_1, \leftarrow, E\}$, $|C| < \min(S, poly(\log N))$ *and space* $O(1)$, *Stein algorithm is optimal.*

7. DIRECT SUM COMPLEXITY OF GCD

Let $f : D \to \mathbf{Q}^s$ be a function over the domain $D \subseteq \mathbf{Q}^r$. The t-Direct sum complexity of f over the domain D in $RAM(F)$ is the minimal number of operations needed to compute $f(x_1), f(x_2) \cdots, f(x_t)$ over all $(x_1, \cdots, x_t) \in D^t$, i.e

$$t\text{-}DComp_F(f) \triangleq Comp_F(f \oplus f \oplus \overset{t}{\cdots} \oplus f) = Comp_F((f, f, \cdots, f)).$$

Obviously, $Comp_F(\oplus_{i=1}^t f) \leq tComp_F(f)$, but as we shall show in this small section, equation may not hold. The definition of the t-Direct sum complexity can be found in [B1] and [B8] for the straight line model and in [BBF] and [B6] for other models.

Our main result in this section is

Theorem 7. *Using* $F = \{+, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, \leftarrow, >\}$, *we can compute* $GCD(x_1, y_1), GCD(x_2, y_2), \cdots,$ $GCD(x_t, y_t)$, $t < N$, $x_i, y_i \in [0, N]$ *by*

$$O\left(\frac{t \log N}{\log\log N + \log t}\right)$$

operations with space $O((t \log N)^\epsilon)$ *for any constant* $0 < \epsilon < 1$.

Proof . We use the GCD algorithm with $k = \lfloor \tau(\log\log N + \log t) \rfloor$, $0 < \epsilon = 4\tau < 1$ and compute $GCD(x_i, y_i)$, $i = 1, \cdots, t$ saving the arrays $\alpha(), \beta(), \gamma()$ with thier previous content in each call. ○

Before we prove that this bound is tight we give the following

Lemma 19. *Let* $L_i \subset \mathbf{N}, i = 1, \cdots, t$, $|L_i| = N_i$. *If* L_i *contain no arithmetic progression of length* $k_i + 1$ *then each*

$$RAM\left(G = \mathbf{Q}[N], +, -, \times_G, /_G, F\left[\left[O\left(\log\left(\sum_{i=1}^{t} \log N_i\right)\right)\right]\right], \leftarrow, >\right)$$

for $L_1 \times L_2 \times \cdots \times L_t$ *has operation complexity*

$$\Omega\left(\frac{\sum_{i=1}^{t} \log N_i}{\log\left(\sum_{i=1}^{t} \log N_i\right) + \max_{1 \le i \le t} \log k_i} - t\right). \qquad (*)$$

This bound is also true even if we do not count the operations $\{+, -, \times_G, /_G\}$.

Proof . Let $M = 1 + \max \cup_{i=1}^{t} L_i$ and

$$\hat{L} = \{l_1 + l_2 M^2 + \cdots + l_t M^{2t-2} \mid l_i \in L_i,\ i = 1, \cdots, t\}.$$

We first prove the lower bound (*)(without $-t$) for $\hat{L}$.

Let $A, A + D, \cdots, A + \lambda D$ be an arithmetic progression of length $\lambda + 1$ in $\hat{L}$. Since $A \in \hat{L}$ we have

$$A = a_1 + a_2 M^2 + \cdots + a_t M^{2t}, \quad a_i \in L_i.$$

Let $0 < D = d_1 + d_{1.5} M + d_2 M^2 + \cdots + d_t M^{2t-2}$, $M > d_i \ge 0$. Since $A + iD \in \hat{L}$, $i = 0, \cdots, \lambda$ we have $(A + iD) \bmod M^2 \in L_1$ and therefore

$$L_1 \ni (a_1 + id_1) + (id_{1.5} M) \bmod M^2 < M. \qquad (18)$$

Then three cases can happen

(*i*) $d_{1.5} = 0$, $d_1 \neq 0$. In this case $a_1 + id_1 < M$ and $a_1 + id_1 \in L_1, i = 1, \cdots, \lambda$. Then since L_1 cannot contain arithmetic progression of length $k_1 + 1$ we have $\lambda < k_1$ and therefore (*)(without $-t$) follows for $\hat{L}$.

(*ii*) $d_{1.5} = M - 1$, $d_1 \neq 0$. In this case $(a_1 + id_1) + i(M-1)M \bmod M^2 = a_1 + id_1 - iM \in L_1$. Here again we have $a_1 + i(d_1 - M)$ is an arithmetic progression of length $\lambda < k_1$, which implies (*)(without $-t$) for $\hat{L}$.

(*iii*) $d_1 = 0$. In this case, (18) implies $d_{1.5} = 0$. Now the sequence $\lfloor (A-a_1)/M \rfloor, \lfloor (A-a_1)/M \rfloor + D, \cdots, \lfloor (A-a_1)/M \rfloor + \lambda D$ is arithmetic progression in $\hat{L}_1 = \{l_2 + l_3 M^2 + \cdots + l_t M^{2t-4} \mid l_i \in L_i,\ i = 2, \cdots, t\}$ and by induction we have (*)(without $-t$) .

Now since $l_1, \cdots, l_t$ can be computed from $l_1 + l_2 M^2 + \cdots + l_t M^{2t-2}$ by t floor operations we obtain the result (*)(with $-t$) for $L_1 \times \cdots \times L_t$. ○

Now we prove the lower bound for t-Direct sum of GCD

Theorem 8. *Let* $t = o(N)$ *and*

$$L = \{(x_1, x_2, \cdots, x_t, y) |\ GCD(x_1, y) = GCD(x_2, y) \cdots = GCD(x_t, y) = 1\} \subset [0, N]^{t+1}$$

Then any program in

$$RAM(G = \mathbf{Q}[y, N, t], +, -, \times_G, /_G, \lfloor\ \rfloor, F[[O(\log t + \log\log N)]], \leftarrow, >)$$

that recognize L *has operation complexity*

$$\Omega\left(\frac{t \log N}{\log\log N + \log t}\right).$$

Proof . As in the proof of Theorem 4 , we substitute $y = M = p_1 \cdots p_k$ and we look at the set $L' \times \overset{t}{\cdots} \times L'$ where $L' = \{x |\ GCD(x, M) = 1\}$. Now we have (see proof of Theorem 4)

$$|L' \times \cdots \times L'| = \left(c_5 \frac{\log N}{\log\log N}\right)^{c_5 \frac{\log N}{\log\log N} t}$$

and L' has no arithmetic progression of length $c_2 \log N$. This with lemma 19 implies the result. ○

8. COMPLEXITY OF GCD WITH TABLES

When we need to compute $GCD(x_i, y_i)$ for many $x_i, y_i \in [0, N]$ we can first build a table of the GCD values of size N^2 and then compute the GCD in $O(1)$ operations. The question we shall answer in this section is: For large N, how much can tables of size $M = o(N)$ accelerate the computation of $GCD(x, y)$, $x, y \in [0, N]$. We denote by $T(M)$ the set of all tables of size M.

In [B3] we prove that using $RAM(+, -, \times, /, \leftarrow, E, T(N^\epsilon))$ the operation complexity of $GCD(x, y)$, $x, y \in [0, N]$ is $\Theta(\log N)$. In this section we prove

Theorem 9. *Using* $F = \{+, -, \times_{\mathbf{Z}}, \lfloor /2^i \rfloor, \leftarrow, >, T(M)\}$, *we can compute* $GCD(x, y)$, $x, y \in [0, N]$ *by*

$$O\left(\frac{\log N}{\log\log N + \log M}\right)$$

operations.

Therefore with tables of size N^ϵ, *where* $0 \le \epsilon \le 1$ *is a constant, we can compute* $GCD(x, y)$ *in* $O(1)$ *operations.*

Proof . We use the table of $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, j = 1,2$ produced from algorithm B for $0 \leq x_1, x_2, y_1, y_2 \leq 2^k$, where $k = \lfloor \frac{1}{4} \log M \rfloor$. The size of each table is $2^{4k} \leq M$ and by lemma 3 using these constants we can reduce the length of x and y by $O(k)$ bits in $O(1)$ operations. This implies that the operation complexity of GCD using these tables is $O\left(\frac{\log N}{\log M}\right)$. With the bound in Theorem 1 the result follows. ○

We also prove that this bound is tight

Theorem 10. *Let*

$$L = \{(x,y) |\ x, y \leq N \text{ are relatively primes } \}.$$

Then any program in

$$RAM(G = \mathbf{Q}[y,N], +, -, \times_G, /_G, \lfloor\ \rfloor, F[[O(\log\log N + \log M]], \leftarrow, >, T(M))$$

that recognize L has operation complexity

$$\Omega\left(\frac{\log N}{\log\log N + \log M}\right).$$

Proof . The tables are functions $f_\pi : [1,M] \rightarrow \mathbf{Q}$, $\pi \in \mathbf{Q}^M$. Since $\mathbf{Q}$ is given in the model, every function f_π can be computed in $logM$ operations without indirect addressing. Now as in the proof of lemma 15, if t is the operation complexity of L then there exist a program without tables and without indirect addressing that recognize L with $O(t(\log t + \log\log N + \log M))$ operations. Now if we proceed exactly as in the proof of lemma 15 we obtain the result. ○

9. GCD AND LCM OF MANY NUMBERS

In this section we study the operation complexity of computing $GCD(x_1, x_2, \cdots, x_n)$ and $LCM(x_1, x_2, \cdots, x_n)$ where $x_1, x_2, \cdots, x_n \in [0, N]$. Obviously since $LCM(x_1, x_2) = x_1 x_2 / GCD(x_1, x_2)$ the upper bound for $GCD(x_1, x_2)$ is also true for $LCM(x_1, x_2)$. This property is not true for GCD of n numbers. We first prove

Theorem 11. *Using* $\{+, -, \times, /, \lfloor\ \rfloor, \leftarrow, >\}$ *we can compute* $GCD(x_1, x_2, \cdots, x_n)$, $x_1, x_2, \cdots, x_n \in [0, N]$ *by*

$$O\left(\frac{\log N}{\log\log N} + n\right)$$

operations.

Proof . We use the following algorithm.

(1) $i \leftarrow 1,\ d_1 \leftarrow x_1,\ d_2 \leftarrow x_2$

(2) For $i := 3$ to n do $\{d_1 \leftarrow GCD(d_1, d_2),\ d_2 \leftarrow x_i \bmod d_1\}$

(3) Output$(GCD(d_1, d_2))$

Obviously, this algorithm compute $GCD(x_1, \cdots, x_n)$. Since by lemma 3 $GCD(d_1, d_2)$ can be computed by

$$c\frac{\|d_1\| + \|d_2\| - \|GCD(d_1, d_2)\|}{\log(\|d_1\| + \|d_2\|)} + 1$$

operations for some constant c, the operation complexity of this algorithm is at most

$$c\frac{\|N\| - \|GCD(x_1, x_2)\|}{\log \|N\|} + 1 + c\frac{\|GCD(x_1, x_2)\| - \|GCD(x_1, x_2, x_3)\|}{\log \|N\|} + 1 +$$

$$\cdots + c\frac{\|GCD(x_1, \cdots, x_{n-1}\| - GCD(x_1, \cdots, x_n)\|}{\log \|N\|} + 1 = c\frac{\|N\|}{\log \|N\|} + n = O\left(\frac{\log N}{\log \log N} + n\right). \quad \bigcirc$$

We do not know how to achieve this bound for LCM. The best bound we can prove for LCM is

Theorem 12. *Using* $\{+, -, \times, /, \lfloor\ \rfloor, \leftarrow, >\}$ *we can compute* $LCM(x_1, \cdots, x_n)$, $x_1, \cdots, x_n \in [0, N]$ *by*

$$O\left(\frac{n \log N}{\log \log N + \log n}\right)$$

operations.

Proof . We do this by the following algorithm

(1) Build a table of $\alpha(), \beta(), \gamma()$ of size $(n \log N)^\epsilon$, $\epsilon < 1$

(2) $z_1 \leftarrow x_1$, $z_2 \leftarrow x_2$

(3) For $i := 3$ to n do $\{z_1 \leftarrow z_1 z_2 / GCD^*(z_1, z_2),\ z_2 \leftarrow x_i\}$

(4) Output$(z_1 z_2 / GCD^*(z_1, z_2))$

Here $GCD^*(z_1, z_2) = GCD(z_i \bmod z_{3-i}, z_i)$ where $z_i > z_{3-i}$. With this algorithm the result follows $\bigcirc$

For lower bound, no technique is known that can prove tight bound for these problems using the operations $\{+, -, \times, /, \lfloor\ \rfloor, \leftarrow, >\}$ We shall prove this lower bound for the LCM problem in a weaker RAM model

Theorem 13. *Let*

$$L = \{(x_1, x_2, \cdots, x_n) |\ LCM(x_1, \cdots, x_n) = M\} \subset [0, N]^n.$$

There exist M *such that any program in*

$$RAM(G = Q[N], +, -, \times_G, /_G, \lfloor\ \rfloor, F[[O(\log \log N + \log n]], \leftarrow, >)$$

that recognize L *has operation complexity*

$$\Omega\left(\frac{n \log N}{\log \log N + \log n}\right).$$

Proof . Let $M = p_1 \cdots p_k$ where $M < N$ and $M p_{k+1} > N$. Let

$$L = \{(x_1, \cdots, x_n) |\ GCD(x_1, M) = \cdots = GCD(x_n, M) = 1\}$$

$$= \{(x_1, \cdots, x_n) |\ GCD(LCM(x_1, \cdots, x_n), M) = 1\}.$$

Since L can be recognized by first computing $y = LCM(x_1, \cdots, x_n)$ and then verify if $p_i|y$, $i = 1, \cdots, k$ we have that the operation complexity of $LCM(x_1, \cdots, x_n)$ is at least $Comp_F(\chi_L) - k$.

By theorem 4 and since $k \leq c\left(\frac{\log N}{\log\log N}\right)$ the result follows. ○

Notice that when $\{\times\} \subset F$ we can recognize L in the proof of Theorem ? by $O(n+k)$ operations and word length $O(n\|N\|)$. We first compute $y = x_1 \cdots x_n$ by $n-1$ operations and then verify if $p_i|y$, $i = 1, \cdots, k$. Therefore we cannot prove this lower bound when we also allow $\times$.

Appendix A.

In this appendix we give two lemmas needed in the paper

Lemma A.1. Using $\{\{1\}, +, -, \times_{\mathbf{Z}}, \lfloor/2^i\rfloor, >\}$ we can compute $\lfloor\log_2 x\rfloor$ and $\|x\|$ by

$$O(\log\log x)$$

operations with word length $\|x\|$ and space $\log\log x$.

Proof .

Compuing $d = \lfloor\log_2 x\rfloor$, $x \geq 2$.

(i) $c_0 \leftarrow 2$, $i \leftarrow 0$, $x \leftarrow \lfloor x/2\rfloor$, $d \leftarrow 1$, $l_0 \leftarrow 1$.

(ii) While $x \geq c_i$ do $\{x \leftarrow \lfloor x/c_i\rfloor$, $d \leftarrow d + l_i$, $i \leftarrow i+1$, $l_{i+1} \leftarrow l_i + l_i$, $c_{i+1} \leftarrow c_i \times c_i\}$

(iii) For $j := i-1$ to 0 do { If $x \geq c_i$ then $\{x \leftarrow \lfloor x/c_j\rfloor$, $d \leftarrow d + l_j\}\}$

Computing $d = \|x\|$

$d \leftarrow \lfloor\log_2 x\rfloor + 1$. ○

Lemma A.2. Let $\alpha, \beta \in [1, \sqrt{N}]$, $\alpha, \beta \in [0, N]$ and $\alpha, \beta < \gamma = 2^i$. Then

$$C = \frac{\alpha x + \beta y}{\gamma} \in [0, N]$$

can be computed by $O(1)$ operations with word length $\|N\|$.

Proof . We have

(1) $x_2 \leftarrow \lfloor x/\gamma\rfloor$, $x_1 \leftarrow x - \gamma x_2$, $y_2 \leftarrow \lfloor y/\gamma\rfloor$, $y_1 \leftarrow y - \gamma y_2$.

(2) $c_2 \leftarrow \alpha x_2 + \beta y_2$, $c_1 \leftarrow \lfloor(\alpha x_1 + \beta y_1)/\gamma\rfloor$.

(3) $c \leftarrow c_1 + c_2$. ○

REFERENCES

[AHU] A.A. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms* , Addison-Wesley, Reading, Massachusetts, 1974.

[B] M. Ben-Or, Lower bounds for algebraic computation trees, In *Proc. 15th ACM Symp. on Theory of Computing*, pp 80-86, May 1983.

[BGH] A. Borodin, J. von zur Gathen, J. Hopcroft, Fast parallel matrix and GCD computations, Proc. 23rd Annual Symp. FOCS, 1982, 65-71.

[BR] R. P. Brent, Analysis of the binary Euclidean algorithm, *Algorithms and Complexity*, ed. J. F. Traub, 321-355, (1976).

[Br] W. S. Brown, On Euclid's algorithm and the computation of polynomial greatest common divisors, *J. of ACM*, **18**, 4, 478-504, (1971).

[BT] W. S. Brown, J. F. Traub, On Euclid's algorithm and the theory of subresultants, *J. of ACM*, **18**, 4, 505-514, (1971).

[B1] N. H. Bshouty, On the extended direct sum conjecture, Proceeding of the 21st annual ACM Symposium on Theory of Computing, (STOC), (May 1989)

[B2] N. H. Bshouty, Lower bounds for algebraic computation trees of functions with finite domains, TR-576, Technion, Israel, July 1989.

[B3] N. H. Bshouty, Lower bound for the complexity of functions in random access machines. TR 583, Technion, August 1989.

[B4] N. H. Bshouty, $\Omega(\log\log(1/\epsilon))$ lower bound for approximating the square root. TR 89/367/29, University of Calgary, October 1989.

[B5] N. H. Bshouty, Lower bounds for computation with the floor operation and bitwise boolean operations. In preperation.

[B6] N. H. Bshouty, On the direct sum complexity of problems. In preperation.

[B7] N. H. Bshouty, On the GCD of polynomials over finite fields. In preperation.

[B8] N. H. Bshouty, On the direct sum conjecture in the straight line model. TR 580, Technion, July 1989.

[BBF] D. H. Bshouty, N. H. Bshouty, F. Fich, Bounds for the number of binary operations needed to compute functions by straight line algorithms. In preperation.

[C] G. E. Collins, The computing time of the Euclidean algorithm, *SIAM J. Comput*, **3**, 1-10, (1974).

[D] J. D. Dixon, The number of steps in the Euclidean algorithm, *J. Number Theory*, **2**, 414-422, (1970).

[H] J. Hong, On lower bounds of time complexity of some algorithms, *Scientia Sinica*, **22** , 890-900, (1979).

[JMW] B. Just, F. Meyer auf der Heide, A. Wigderson. On computation with integer division. In *Proc. 5th STACS, Lecture Notes in Computer Science.* 294,pp. 29-37. Springer-Verlage, February 1988.

[KR] D. Kirkpatrick, S. Reisch, Upper bounds for sorting integers on random access machines. *Theoretical Computer Science*, **28**, (1984), 263-276.

[K] D. Knuth, Art of computer programming, Vol. II, Addison-Wesley.

[La] G. Lamé, Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers. *C. R. Acad. Sci. Paris* **19**, (1944), 867-870.

[Le] D. H. Lehmer, Euclid's algorithm for large numbers. *Amer. Math Monthly*, **45**, (1938), 227-233.

[Ma] K. Ma, Analysis of polynomial GCD computations over finite fields. *Tech. Rep*, **195**, Dept. of Computer Science, University of Toronto.

[MG] K. Ma, J. von zur Gathen, Analysis of Euclidean algorithm for polynomials over finite fields. *J. Symb. Comp.*, to appear.

[MST1] Y. Mansour, B. Schieber, P. Tiwari, Lower bounds for integer greatest common divisor computation, In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pp. 54-63, October 1988.

[MST2] Y. Mansour, B. Schieber, and P. Tiwari, Lower bounds for computations with the floor operation. TR, IBM Watson, 63861. (1988).

[MST3] Y. Mansour, B. Schieber, and P. Tiwari, The complexity of approximating the square root. *Proc. 30th IEEE Symp. on Foundations of Computer Science* , 1989.

[M2] F. Meyer auf der Heide, On genuinely time bounded computations. TR, Dortmund Univ.

[Mo] R. T. Moenck, Fast computation of GCDs, Proc Fifth ACM Sump. on Theory of Computing, New York, NY , 142-151, (1973). preprint.

[PS] W. Paul, J. Simon, Decision trees and random access machines, in *Monographie 30, L'Enseignement Mathematique, Logic and Algorithmic-An International Symposium Held in Honor of Ernst Specker. Univ. Geneva Press,*331-340, (1982).

[Sc0] A. Schönhage, Schnelle Berechnung von kettenbruchentwicklugen, *Acta Informatica*, **1**, 139-144, (1971).

[Sc1] A. Schönhage, Probabilistic computation of integer polynomial GCD, *J. of Algorithms* ,**9**, 365-371, (1988).

[Sc2] A. Schönhage, Quasi-GCD computations , *J. Complexity* ,**1**, 118-137, (1985).

[Sh] J. Shamir, Factoring numbers in $O(\log n)$ arithmetic steps, *Info. Proc. Letters*, **8**, (1979), 28-31.

[S] J. Stein, Computational problems associated with Racah Algebra , *J. Comp. Phys*, (1967), 397-405.

[Y] A. Yao, Lower bounds for Algebraic Computation Trees with Integer Inputs, *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989.

[YK] A. Yao, D. E. Knuth, Analysis of the subtractive algorithm for greatest common divisors, *Proc. Nat. Acad. Sci. USA*, **72**, 4720-4722, (1975).