LOWER BOUNDS FOR THE COMPLEXITY OF FUNCTIONS
IN RANDOM ACCESS MACHINES

by

Nader H. Bshouty

Technical Report #583
August 1989
LOWER Bounds FOR THE COMPLEXITY OF FUNCTIONS IN RANDOM ACCESS MACHINES

Nader H. Bshouty

Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000
Israel

ABSTRACT:

We prove tight bounds for Sort, Merge, Insert, Gcd of integers, Gcd of polynomials and Rational functions with finite inputs domain in random access machine with arithmetic operations, direct and indirect addressing, unlimited power of answering YES/NO questions, branching and tables with bounded size. These bounds are also true even if we do not count additions, subtractions, multiplication and division by elements of the field.

In random access machine with finite initial constants and bounded types of instructions we prove that the complexity of a function in infinite countable domain is equal to the complexity of the function in sufficient large finite subdomain.
1. INTRODUCTION

Let $R$ be a commutative ring with no zero divisions (integer domain). Let $F = R(f)$ be the quotient field of $R$ and $E \subseteq F$ is any infinite extension of $F$. A $(E, \tau, \delta)$-random access machine (RAM) $M$ has unbounded number of registers $\{M[e]\}_{e \in E}$ each of which can store an element $a \in E$. The register $M[0]$ is called the accumulator $A$. The computation is directed by a finite program that consists of instructions of the following type: direct and indirect storage accesses, conditional branching (IF-THEN-ELSE), $\tau$ condition branching (CASE), tables of length $\delta$ and arithmetic operations $\{+,-,x,\div\}$. To formalize this we define the following: A $(E, \tau, \delta)$-program is a finite sequence $P = (1 : I_1), \ldots, (q : I_q)$ of instructions from the following set: (1) $A \leftarrow e$, $A \leftarrow M[e]$, $A \leftarrow M[M[e]]$. (2) $M[e] \leftarrow A$, $M[M[e]] \leftarrow A$. (3) $A \leftarrow A \circ M[e]$ where $\circ \in O(\{+,-,x,\div\})$. (4) IF $<\text{condition on } z>$ THEN GOTO $d_1$ ELSE GOTO $d_2$. (5) CASE $<\text{condition 1 on } z>$ GOTO $d_1$, $\ldots$, $<\text{condition } \tau - 1 \text{ on } z>$ GOTO $d_{\tau-1}$ ELSE GOTO $d_\tau$. (6) CASETABLE $A = v_1$ GOTO $d_1$, $\ldots$, $A = v_6$ GOTO $d_6$ ELSE GOTO $d_{11}$, where $v_i \in E$. Here $O(\{+,-,x,\div\})$ is any operation that can be performed by (1), (2), (4), (5), (6) and (3) with the arithmetic operations $\{+,-,x,\div\}$ by complexity $O(1)$ (See definitions bellow). In the program $P$ the first instruction $I_1$ is the input step of $z = (z_1, \ldots, z_r) \in X_{i=1}^r S_i$, $S_i \subseteq R$, and they assumed to be in some registers $e_1, \ldots, e_r \in E \setminus \{0\}$. The content of some registers $e_1^0, \ldots, e_r^0 \in E \setminus \{0\}$ after the execution of the program are the outputs.

We say that the program $P_f$ computes $f = (f_1, \ldots, f_s) : \times_{i=1}^r S_i \rightarrow R'$ if the execution of the program for the inputs $z = (z_1, \ldots, z_r) \in \times_{i=1}^r S_i$ stops with $M[e_j^0] = f_j(z)$, $j = 1, \ldots, s$. The complexity $\text{Comp}(P_f)$ of the program $P_f$ is the maximal number of steps of the form (1), (2), (3), (4), (5) and (6) over all possible inputs $z \in \times_{i=1}^r S_i$ needed to execute until the program stops. The complexity $\text{Comp}(f, \times_{i=1}^r S_i, \tau, \delta)$ of the function $f$ is $\min \text{Comp}(P_f)$ over all programs $P_f$ that computes $f$ in the domain $\times_{i=1}^r S_i$.

In this paper we prove tight bounds for Sort, Merge, Insert, Gcd of integers, Gcd of polynomials and Rational functions with finite inputs domain with bounded size tables. These bounds are also true even if we do not count additions, subtractions, multiplication and division by elements of the field. These results are proved in [3] without indirect addressing and tables. When the domain is not finite then many tight bounds are known in the literature, see [2], [4-7], [9].

When we restrict $e \in L$, $L$ is finite (for example $\{0,1\}$ for the integers) and the condition in the IF command to finite types (for example $A > 0$, Is odd?, Is prime?) and we allow any finite set of binary operations (for example, $+, -, x, \div$, $\text{mod}$, $\text{div}$, $\ldots$) then we prove that the complexity of $f$ over infinite countable domain is equal to the complexity of $f$ over sufficient large finite subdomain. This result follows that all the results in [2], [5], [6] and [9], are also true for sufficient large finite subdomains. Since the decision trees are
equivalent to a program that compute a function with the range \{0, 1\} we can also applied this result for decision trees.

2. BASIC DEFINITIONS AND PRELIMINARY RESULTS

In this section we introduce some notation and prove the major lemmas and theorems needed to prove our results.

Throughout the paper \( R \) is a commutative ring with no zero divisions and \( F = R(f) \) is a quotient field of \( R \), \( F[x_1, \ldots, x_r] \) is the ring of multivariate polynomials in the indeterminates \( x_1, \ldots, x_r \) with coefficients in \( F \), \( F(x_1, \ldots, x_r) \) is the field of rational functions in the indeterminates \( x_1, \ldots, x_r \) with coefficients in \( F \).

For a set \( A \) the notation \( |A| \) is the cardinality of \( A \).

We first start by studying the behavior of the roots of rational functions in finite domains.

**Definition 1.** For a vector of rational functions \((f_1, \ldots, f_s) \in F(x_1, \ldots, x_r)^s\) we define

\[
\text{max-deg}(f_1, \ldots, f_s) = \max_{1 \leq i \leq s} \text{max-deg}(f_i),
\]

where for \( f_{i,1}, f_{i,2} \in F[x_1, \ldots, x_r] \)

\[
\text{max-deg} \left( \frac{f_{i,1}}{f_{i,2}} \right) = \max \left( \deg \left( \frac{f_{i,1}}{\gcd(f_{i,1}, f_{i,2})} \right), \, \deg \left( \frac{f_{i,2}}{\gcd(f_{i,1}, f_{i,2})} \right) \right).
\]

**Lemma 1.** [3]. Let \( S_i = \{s_{i,1}, \ldots, s_{i,n}\} \subset F \) where \( |S_i| = n, i = 1, \ldots, r \). For two vectors of rational functions \( f, g \in F(x_1, \ldots, x_r)^s \) if \( f \neq g \) then the equation \( f = g \) has at most

\[
(\text{max-deg } f + \text{max-deg } g)n^{r-1}
\]
solutions in \( S = S_1 \times \cdots \times S_r \).

Let \( f \in F(x_1, \ldots, x_n)^s \). We denote by \( \mu(f) \) the number of arithmetic operations needed to compute \( f \) by a straight line algorithm.

**Lemma 2.** [8]. Let \( f \in F(x_1, \ldots, x_r)^s \). Then

\[
\mu(f) \geq \lceil \log(\text{max-deg } f) \rceil.
\]

Our main result is

**Theorem 1.** Let \( S_{i,n} = \{s_{i,1}, \ldots, s_{i,n}\} \subset R \) where \( |S_{i,n}| = n, i = 1, \ldots, r \). Let \( f : x_{i=1}^{r} S_{i,n} \rightarrow R^w \) be a function on the domain \( x_{i=1}^{r} S_{i,n} \). Let \( \text{Comp}(f, x_{i=1}^{r} S_{i,n}, \tau, \delta') \leq C(n) \). Assume that there exist distinct sequences of rational functions \( f_1, \ldots, f_w \in R(x_1, \ldots, x_r)^s \) such that

\[
\frac{1}{n^{r-1}} \left| \{x \in x_{i=1}^{r} S_{i,n} | f(x) = f_i(x) \} \right| \geq \Phi(n), \quad i = 1, \ldots, w.
\]
If
\[ C(n) = \Theta \left( \max \left( \frac{\log w}{\log \tau}, \max_{1 \leq i \leq w} (\mu(f_i)) \right) \right), \]
and there exist a constant \( \lambda \) such that
\[ \frac{C(n)}{\lambda} = o(\Phi(n)) \]
then for every constant \( \lambda_0 > 1 \) and
\[ \delta(n) = \frac{\Phi(n)}{\tau C(n)/\lambda_0} > \delta' \]
there exist \( n_0 \) such that for every \( n > n_0 \) we have
\[ \text{Comp}(f, X_{i=1}^n S_{i,n}, \tau, \delta(n)) = \Theta(C(n)). \]

Proof. Since \( \text{Comp}(f, X_{i=1}^n S_{i,n}, \tau, \delta') \leq C(n) \) we have
\[ \text{Comp}(f, X_{i=1}^n S_{i,n}, \tau, \delta) \leq C(n), \]
for every \( \delta > \delta' \).

By (7) there exist a constant \( b \) such that
\[ C(n) \leq b \max \left( \frac{\log w}{\log \tau}, \max_{1 \leq i \leq w} (\mu(f_i)) \right). \]

Since \( \frac{C(n)}{\lambda_1} = o(\Phi(n)) \) there exist a constant
\[ \lambda_1 > \max\{2\lambda, \lambda_0, b\} \]
such that
\[ (2r)^{\frac{C(n)}{\lambda_1}} \frac{\delta(n)}{\lambda_1} = o(\Phi(n)), \quad (2r)^{\frac{C(n)}{\lambda_1}} \frac{C(n)^2}{\lambda_1} = o(\Phi(n)), \quad 2^{\frac{C(n)}{\lambda_1}} = o(\Phi(n)). \]

Let \( P = (1 : I_1), \ldots, (q : I_q) \) be a minimal program that computes \( f \) in some domain \( X_{i=1}^n S_{i,n} \). We shall prove that using \( \{+,-,\times,\div\} \), the complexity of \( f \) is \( O(C(n)) \). This will implies the result (10) for the operations \( O(\{+,-,\times,\div\}) \) since we can transfer each instruction \( A \leftarrow A \circ M[l] \) with \( o \in O(\{+,-,\times,\div\}) \) to a sequence of \( O(1) \) complexity program that uses the instructions (1), (2), (4), (5), (6) and (3) with the arithmetic operations \( \{+,-,\times,\div\} \).

If there exist \( n_0 \) such that for every \( n \geq n_0 \) we have \( \text{Comp}(f, X_{i=1}^n S_{i,n}, \tau, \delta(n)) \geq \frac{C(n)}{\lambda_1} \) then with (11) we have \( \text{Comp}(f, X_{i=1}^n S_{i,n}, \tau, \delta) = \Theta(C(n)) \). Therefore there exist infinity many integers \( n \in \mathbb{N} \) such that
\[ \text{Comp}(f, X_{i=1}^n S_{i,n}, \tau, \delta(n)) \leq \frac{C(n)}{\lambda_1}. \]

All over the proof we assume \( n \in \mathbb{N} \).
We associate with \( P \) a computation \( \tau \)-tree \( T \) that defined in [3] and [7]. The tree \( T \) has instructions of the type (1), (2) and (3) with \( o \in \{ +, -, \times, \div \} \). The CASE instruction (also the IF instruction) is replaced by figure 1.a and the CASETABLE instruction is replaced by figure 1.b

\[
\begin{array}{c|l}
\text{x} \in P_1 & A = v_1 \ A = v_2 \ \cdots \ A = v_{\delta(n)} \\
\vdots & \vdots \\
\text{e}_1 \land e_2 \land \cdots \land e_{\tau-1} & \delta(n) \ \delta(n)+1 \\
\text{e}_\tau & \\
\end{array}
\]

1.a

\[
\begin{array}{c|l}
\text{A} = v_1 & A = v_2 \ \cdots \ A = v_{\delta(n)} \\
\vdots & \vdots \\
\text{e}_1 \land e_2 \land \cdots \land e_{\tau} & \delta(n) \ \delta(n)+1 \\
\text{e}_\tau & \\
\end{array}
\]

1.b

where \( P_i = \{ x | x \text{ satisfies condition } i \}, i = 1, \ldots, \tau - 1 \) and \( P_\tau = \{ x | x \text{ does not satisfy any condition } i \}, i = 1, \ldots, \tau - 1 \). In this vertex the computation proceed to the \( i \)-th child if \( x \in P_i \). In figure 1.b the computation proceed to child \( i \) if the content of \( A \) is \( v_i \) and proceed to the \( \delta(n) + 1 \) child if the content of \( A \) is not in \( \{ v_1, \ldots, v_{\delta(n)} \} \). We call the leaves that have ancestor which is \( i \)-th child of some CASETABLE vertex, \( i \leq \delta(n) \), nonactive leaves. Since the height of the tree is less than \( C(n)/\lambda_1 \), (the complexity of \( f \)), the number of active leaves in the tree is

\[
l \leq \tau^{\frac{C(n)}{\lambda_1}}. \tag{16}
\]

Now we take each vertex \( v \) in this tree. If the instruction in vertex \( v \) is \( A \leftarrow M[M[e]] \) or \( M[M[e]] \leftarrow A \) and \( M[e] \equiv e' \) for some \( e' \in E \), then we replace this instruction by \( A \leftarrow M[e'] \) or \( M[e'] \leftarrow A \), respectively. Therefore we can assume that if we have indirect addressing \( M[M[e]] \) then \( M[e] \) is depending on the inputs \( x_1, \ldots, x_\tau \).

Let \( A(E) \) be all the addresses that used for all the possible inputs \( x_1, \ldots, x_\tau \). Since the domain \( x_1 \in S_i, n \) is finite \( A(E) \) is also finite. Since \( E \setminus A(E) \) is infinite we can choose new addresses \( a_1, a_2, \ldots \) that never used in the computation.

We now remove the first \( \delta(n) \) leaves in all the CASETABLE vertices and obtain a new computation tree that computes \( f \) in all inputs \( x \) except for inputs in some set \( P \).

Now we use the technique in [7] to transfer the computation tree to a computation tree without indirect addressing. Let \( v_1, \ldots, v_k \) be a path from the root \( v_i \) of \( T \) to some leaf \( v_h \). We define \( I_{v_i}(M[j]) \in F(x_1, \ldots, x_\tau) \) to be the rational function that the register \( M[j] \) contains when the computation arrived at vertex \( v_i \). Since we use indirect addressing in the tree this is not well defined so we let this notation to be defined for all the first \( k \) vertices \( v_1, \ldots, v_k \) that does not use indirect addressing. Thus \( I_{v_i}(M[j]) = x_j, j \in \{ e'_1, \ldots, e'_1 \} \) and \( I_{v_i}(M[j]) = 0 \) for \( j \not\in \{ e'_1, \ldots, e'_1 \} \). To make \( I_{v_i} \) well defined for all the vertices in the path \( v_1, \ldots, v_h \) we change the indirect addressing instructions in the following way:
Let \( v_i, \ldots, v_n \) be the vertices that contains the instructions \( M[M[l_i]] \rightarrow A \) or \( M[e_i] \rightarrow A, \ldots, M[M[l_n]] \rightarrow A \) or \( M[e_n] \rightarrow A \), respectively, and no other vertex in the path contains an instruction of this type. If the instruction in \( v_i \) is \( M[M[l_i]] \rightarrow A \) then we change the instructions in \( v_i \) to \( M[a_i] \rightarrow A \). Let \( v_j, \ldots, v_n \) be the vertices that contains the instructions \( A \rightarrow M[M[d_j]] \) or \( A \rightarrow M[\Delta_j] \), respectively, and no other vertex in the path contains an instruction of this type. For every \( e \), if \( i_\phi > j > i_{\psi+1} \), then find the last vertex \( v_{i_\phi} \) such that \( I_{v_{i_\phi}}(M[l_{i_\phi}]) \equiv I_{v_{i_{\psi+1}}}(M[d_{i_{\psi+1}}]) \) and then transfer the instruction in \( v_{i_{\phi}} \) to \( A \rightarrow M[a_j] \). If no such vertex exist then transfer the instruction in \( v_{i_{\phi}} \) to \( A \rightarrow 0 \). In this case the computation tree does not give the desired results for some inputs from \( D = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} D_{v_i, v_j} \) where

\[
D_{v_i, v_j} = \left\{ \begin{array}{l}
\{ x : I_{v_{i_\phi}}(M[l_{i_\phi}]) = I_{v_{i_{\psi+1}}}(M[d_{i_{\psi+1}}]) \} & \text{if } M[M[l_{i_\phi}]] \rightarrow A \text{ in } v_{i_\phi} \text{ and } A \rightarrow M[M[d_{i_{\psi+1}}]] \text{ in } v_{i_{\psi+1}} \\
\{ x : I_{v_{i_\phi}}(M[l_{i_\phi}]) \neq I_{v_{i_{\psi+1}}}(M[d_{i_{\psi+1}}]) \} & \text{if } M[e_{i_\phi}] \rightarrow A \text{ in } v_{i_\phi} \text{ and } A \rightarrow M[M[d_{i_{\psi+1}}]] \text{ in } v_{i_{\psi+1}} \\
\{ x : I_{v_{i_\phi}}(M[l_{i_\phi}]) = \Delta_{l_{i_\phi}} \} & \text{if } M[M[l_{i_\phi}]] \rightarrow A \text{ in } v_{i_\phi} \text{ and } A \rightarrow M[\Delta_{l_{i_\phi}}] \text{ in } v_{i_{\psi+1}}
\end{array} \right.
\]

By lemma 2 we have

\[
h = \max_{v_j} \max_{v_i} \deg I_v(M[l]) \leq 2 \frac{C(\alpha)}{\lambda_1},
\]

and by lemma 1 we have

\[
|D| \leq \lambda(2h)UWn^{r-1}
\]

where \( l \leq \frac{C(\alpha)}{\lambda_1} \) is the number of paths from the root to the leaves and \( V, U \leq C(n)/\lambda_1 \) are the maximal number of instructions of the type \( M[M[l]] \rightarrow A, M[e] \rightarrow A \) and \( A \rightarrow M[M[d]], A \rightarrow M[\Delta] \), respectively, in all paths. Therefore

\[
|D| \leq 2(2r) \frac{C(\alpha) C(n)^2}{\lambda_1^2} n^{r-1} = o(\Phi(n))n^{r-1}.
\]

We now estimate \( |P \cap (x_{i=1}^{r} S_{i,n} \setminus D)| \). Since we have at most \( I \leq \frac{C(\alpha)}{\lambda_1} \) paths from the root to the active leaves and each path has at most \( C(n)/\lambda_1 \) CASETABLE vertices and since by (17) we have \( \max_{v_j} \deg I_v(A) = \max_{v_i} \deg I_v(M[0]) \leq 2 \frac{C(\alpha)}{\lambda_1} \) we have, by lemma 1 and (14),

\[
|P \cap (x_{i=1}^{r} S_{i,n} \setminus D)| \leq \frac{C(\alpha) C(n)}{\lambda_1} \left( 2 \frac{C(\alpha)}{\lambda_1} n^{r-1} \right) \delta(n) = o(\Phi(n))n^{r-1}.
\]

Thus, by (18) and (19), the computation tree now computes \( f \) correctly for a domain \( x_{i=1}^{r} S_{i,n} \setminus (D \cup P) \)

where

\[
| x_{i=1}^{r} S_{i,n} \setminus (D \cup P) | = n^r - \psi(n)n^{r-1} = n^r - o(\Phi(n))n^{r-1}.
\]

Let \( N = \psi(n) + \alpha \) where \( \alpha = I/\beta = I(2 \frac{C(\alpha)}{\lambda_1}) + \max_{v_j} \deg(f_1, \ldots, f_w) \). Since

\[
l \frac{C(\alpha)}{\lambda_1} + 1 \leq (2r)^{\frac{C(\alpha)}{\lambda_1} + 1} = o(\Phi(n))
\]

we have \( N = o(\Phi(n)) \) and therefore there exist \( n_3 \) such that for every \( n \geq n_0 \)

\[
| x_{i=1}^{r} S_{i,n} \setminus f(x) = f_i(x) | \geq (\psi(n) + \alpha)n^{r-1}, \ i = 1, \ldots, w.
\]
Then, by (20), for \( G_i = \{ x \in x_{i=1}^r S_i \setminus (D \cup P) \} \) we have

\[
|G_i| \geq \alpha n^{r-1}. \tag{23}
\]

Since the new tree \( T' \) is now a computation tree that computes \( f \) in \( x_{i=1}^r S_i \setminus (D \cup P) \) without indirect addressing, we can look at each path in the tree from the root to a leaf as a straight line algorithm that computes a rational function. Let \( v^{(1)}, \ldots, v^{(l)} \) be the leaves of the tree and \( g_1, \ldots, g_l \) be the rational functions that are computed in the paths from the root to each leaf, respectively. Let \( Q_1, \ldots, Q_l \) be subsets of \( x_{i=1}^r S_i \setminus (D \cup P) \) where \( Q_j \) is the set of all inputs \( x \in x_{i=1}^r S_i \setminus (D \cup P) \) that terminate in the computation at leaf \( v^{(j)} \). Then

\[
f(x) = g_i(x) \text{ for } x \in Q_i \text{ i} = 1, \ldots, l. \tag{24}
\]

We shall prove that for every \( f_i, i = 1, \ldots, w \) there exist a path in the tree that computes \( f_i \). I.e \( f_i \in \{ g_1, \ldots, g_l \} \) for every \( i = 1, \ldots, w \). Assume that

\[
f_i \notin \{ g_1, \ldots, g_l \}. \tag{25}
\]

Then since

\[
|G_i| = |(\bigcup_{i=1}^l Q_i) \cap G_i| = |\bigcup_{i=1}^l (Q_i \cap G_i)| \geq \alpha n^{r-1}
\]

there exist \( i = j_0 \) such that

\[
|Q_{j_0} \cap G_i| \geq \frac{\alpha}{\tau} n^{r-1} \geq \beta n^{r-1}. \tag{26}
\]

If \( x \in Q_{j_0} \cap G_i \) then \( f(x) = g_{j_0}(x) = f_{j_0}(x) \). Since by (25) we have \( g_{j_0} \neq f_{j_0} \), by lemma 2 we must have

\[
|Q_{j_0} \cap G_i| \leq \left| \{ x | g_{j_0}(x) = f_{j_0}(x) \} \right| \leq (\max - \deg f_{j_0} + \max - \deg g_{j_0}) n^{r-1}
\]

with (26) we obtain that

\[
\max - \deg g_{j_0}(x) \geq 2 \frac{\alpha \gamma}{\alpha^2} + 1.
\]

Therefore computing \( g_{j_0}(x) \) requires more than \( \frac{\alpha \gamma}{\alpha^2} \) arithmetic operations. By (15) we have a contradiction. Therefore

\[
\{ f_1, \ldots, f_w \} \subseteq \{ g_1, \ldots, g_l \}.
\]

Since \( f_1, \ldots, f_w \) are distinct functions we must have \( l \geq w \) and therefore

\[
\text{height}(T') \geq \frac{\log w}{\log \tau}. \tag{27}
\]

Since also we have a path that computes \( f_i, i = 1, \ldots, w \) we also must have

\[
\text{height}(T') \geq \max_{1 \leq i \leq w} \mu(f_i), \tag{28}
\]
Therefore by (12), (13), (27) and (28) we have

\[
\text{Comp}(f, x_{i=1}^r S_i, \tau, \delta(n)) = \text{height}(T') \geq \max \left( \frac{\log w}{\log \tau}, \max_{1 \leq i \leq w} \mu(f_i) \right) \geq \frac{C(n)}{b} > \frac{C(n)}{\lambda_1}.
\]

A contradiction.  

The BRAM \((L, P, G)\) (bounded RAM) is a random access machine that has accumulator \(A\), address register \(B\) and instruction of the following type:

1. \(A \leftarrow e\), where \(e \in L \subseteq E\) and \(|L|\) is finite.
2. \(B \leftarrow A, M[B] \leftarrow A, M[M[B]] \leftarrow A\).
3. \(A \leftarrow M[B], A \leftarrow M[M[B]]\).
4. \(A \leftarrow A \circ M[B]\), where \(\circ \in P\) and \(|P|\) is finite.
5. \(\text{IF } A \in G_i \text{ THEN GOTO } d_1 \text{ ELSE GOTO } d_2\), where \(G \ni G_i \subseteq E, |G|\) is finite.

For a function \(f : x_{i=1}^r S_i \rightarrow R^r\) we denote by \(B\text{Comp}(f, x_{i=1}^r S_i)\) the complexity of \(f\) in the domain \(x_{i=1}^r S_i\) in the model BRAM.

For BRAMs we prove

**Theorem 2.** Let \(S_{i,n} = \{s_{i,1}, \ldots, s_{i,n}\}\) and \(|S_{i,n}| = n, i = 1, \ldots, r\). Let \(f : x_{i=1}^r S_{i,\infty} \rightarrow R^r\). If

\[
B\text{Comp}(f, x_{i=1}^r S_{i,\infty}) < \infty
\]

then there exist \(N\) such that

\[
B\text{Comp}(f, x_{i=1}^r S_{i,N}) = B\text{Comp}(f, x_{i=1}^r S_{i,\infty}).
\]

**Proof.** Obviously,

\[
B\text{Comp}(f, x_{i=1}^r S_{i,N}) \leq B\text{Comp}(f, x_{i=1}^r S_{i,\infty}).
\]

Assume that there exist \(N\) such that for every \(j > 0\)

\[
B\text{Comp}(f, x_{i=1}^r S_{i,N+j}) < B\text{Comp}(f, x_{i=1}^r S_{i,\infty}). \quad (29)
\]

Let \(P_1, P_2, \ldots\) be minimal algorithms for \(f\) in \(x_{i=1}^r S_{i,N+1}, x_{i=1}^r S_{i,N+2}, \ldots\), respectively. Since the number of programs of length less than or equal to \(B\text{Comp}(f, x_{i=1}^r S_{i,\infty})\) in the BRAM is finite, there exist a subsequence of identical programs \(P \equiv P_{N+k_1} \equiv P_{N+k_2} \equiv \ldots, k_1 < k_2 < \ldots\). Since \(P\) is an algorithm for \(f\) in the domain \(x_{i=1}^r S_{N+k_q}, q = 1, 2, \ldots\), then \(P\) is a program for \(f\) in the domain

\[
\bigcup_{q=1}^{\infty} x_{i=1}^r S_{N+k_q} = x_{i=1}^r S_{i,\infty}.
\]

This contradict (29).  

3. RESULTS
In this section we prove our main result. The corollaries that are not proved can be easily proved using theorem 1 and 2 in the same manner as in [3].

Corollary 1. (Rational functions). Let \( f \in P(z_1, \ldots, z_n) \) be a rational function. Then there exist a constant \( c_{f,r} \) such that for sufficient large \( n \) we have

\[
\text{Comp} \left( f, x_{i=1}^n S_{i,n}, \tau, \frac{n}{c_{f,r}} \right) = \Theta(\mu(f)).
\]

Proof. We add to our model the instruction

\text{CASETABLEX} x = u_1 \text{ GOTO } d_1, \ldots, x = u_{d_n-1} \text{ GOTO } d_{d_n-1} \text{ ELSE GOTO } d_{d_n-1+1}, \text{ where } u_i \in \mathbb{R}^r.

Then Theorem 1 is also true when we add this command. Let \( \sigma_1, \ldots, \sigma_{\mu(f)} \) be a straight line algorithm (sequence of rational functions) that computes \( f \). We replace the division steps by CASETABLEX vertices. Since each polynomial in the sequence is of degree less than \( 2^{\mu(f)} \) we need a CASETABLEX vertices with \( \delta = 2^{\mu(f)} < n/c_{f,r} \). Now by theorem 1 with \( f(z) = f(z) \) the result follows.  

Let \( F \) be a field. Let \( S = \{s_1, \ldots, s_N\} \subset F \). Define the order \( s_1 < s_2 < \cdots < s_N \) on \( S \).

Let

\[
A = \left\{ (z_{1,1}, \ldots, z_{1,n}, \ldots, z_{f,1}, \ldots, z_{f,n}) \in S^n \mid n_1 + \cdots + n_f = n, z_{i,j} \text{ distinct} \right\}
\]

Define

\[
\text{Merge}(x_{1,1}, \ldots, x_{1,n}, \ldots, x_{f,1}, \ldots, x_{f,n}) = (y_1, \ldots, y_n)
\]

where \( \{x_{1,1}, \ldots, x_{1,n}, \ldots, x_{f,1}, \ldots, x_{f,n}\} = \{y_1, \ldots, y_n\} \) and \( y_1 < \cdots < y_n \).

We have

Corollary 2. (Merge) There exist a function \( f(n) \) such that for sufficient large \( N \) we have

\[
\text{Comp} \left( \text{Merge}_{n_1, \ldots, n_f} x_{i=1}^n S, \tau, \frac{N}{f(n)} \right) = \Theta \left( \log_r \left( \frac{n}{n_1, \ldots, n_f} \right) \right)
\]

Let \( M = \text{BRAM}(L, \{+, -, \times, [\_], G) \), where \([\_]\) is the integer truncation, \( L \) is finite set of constants and \( G \) finite collection of sets. In [5], Hong proved that in the model \( M \) we have

\[
\text{Comp}(\text{Merge}_{n_1, \ldots, n_f} x_{i=1}^n S) = \Theta \left( \log_r \left( \frac{n}{n_1, \ldots, n_f} \right) \right).
\]

where \( S = \{i/j\} \) integers. By theorem 2 we have

Corollary 3. There exist \( N \) such that for \( S = \{i/j\} \) \( i < N \) we have

\[
\text{BComp}(\text{Merge}_{n_1, \ldots, n_f} x_{i=1}^n S) = \Theta \left( \log_r \left( \frac{n}{n_1, \ldots, n_f} \right) \right).
\]

The tight bound on the \( \text{Merge} \) function give tight bounds for the \( \text{Sort} \) and \( \text{Insert} \) functions in the following way:

Define \( \text{Sort}_n = \text{Merge}_{n, \ldots, 1} \) we obtain
Corollary 4. (Sort) There exist a function $f(n)$ such that for sufficient large $N$ we have

$$\text{Comp}\left(\text{Sort}_n, \times_{i=0}^{n-1} S, \frac{N}{f(n)}\right) = \Theta(n \log_r n).$$

Define $\text{Insert}_{n,i} = \text{Merge}_{1,\cdots,i,1,n}$ we obtain

Corollary 5. (Insert) There exist a function $f(n)$ such that for sufficient large $N$ we have

$$\text{Comp}\left(\text{Insert}_{n,i}, \times_{i=0}^{n-1} S, \frac{N}{f(n)}\right) = \Theta\left(\sum_{i=1}^{j} \log_r (n + i)\right).$$

Corollary 6. (Gcd of Numbers) Let $S_n = \{1, \cdots, n\}$ and let $gcd: S_n^2 \rightarrow S_n$, $gcd(a, b)$ is the greatest common divisor of $a$ and $b$. Then for every $1 > \epsilon > 0$ there exist $N$ such that for every $n > N$ we have

$$\text{Comp}(gcd, S_n \times S_n, \tau, n^\epsilon) = \Theta(\log_r n).$$

Corollary 7. (Gcd of Polynomials) Let $F$ be a finite field and let $F_n = F_n[z]$ be the set of all monic polynomials of rank less than $n+1$. Let $gcd: F_n \times F_n \rightarrow F_n$, $gcd(f, g)$ be the greatest common divisor (monic polynomial). Then for every $1 > \epsilon > 0$ there exist $N$ such that for every $n > N$ we have

$$\text{Comp}(gcd, F_n \times F_n, |F|, |F|^\epsilon) = \Theta(n).$$

The reason we take $\tau = |F|$ is that in $(\ast, |F|, \ast)$ random access machine we can find the coefficient of any $z^i$ of any polynomial in one CASE instruction. We also can substitute any element in $F$ in any polynomial in one CASE instruction.

REFERENCES


