In this paper we consider questions of the time of recognition of sets of words on a machine with arbitrary access to memory. The basic result obtained in the paper asserts that for any function $T(n)$, which can be calculated on a random access machine in time $T(n)$, there exists a set of words $A$ in a two-letter alphabet, recognizable by some machine in time $21T(n) + 6T(n) + 25n$, but which is not recognizable in time $T(n)$. The proof of this result consists of constructing such a set, using a diagonal procedure. It is a refinement of the theorem of Cook–Reckhow on the time hierarchy. Then we define a class of functions $\mathcal{F}$, containing many functions (e.g., polynomials of degree higher than one, $n \log n$, $2^\sqrt{n}$, etc.), which are of interest as complexity estimates. For the class $\mathcal{F}$ we prove a theorem refining the basic result for functions of this class. It consists of the fact that for any function $T(n) \in \mathcal{F}$ and any $c > 1$ there exists a set of words, recognizable in time $cT(n)$ by some random access machine, but not recognizable in time $T(n)$. We also consider random access machines with a bound on the length of register; results are given connecting the time of work of such a machine with the time of work of a machine without restriction on the length of register.

The basic result of the paper is a theorem on the time hierarchy for random access machines saying that under certain additional conditions an increase of 21 times the time of attempted recognition leads to an extension of the class of recognizable sets. This theorem is an improvement of a theorem of Cook–Reckhow (cf. [1]). For a similar model (of address machines) an analogous result is obtained by Slisenko in [2]. Below, a series of consequences of the basic theorem is given, relating to cases when certain additional restrictions are imposed on the time of computation. There are also considered random access machines with a restriction on the length of a register; results are given connecting the time of work of such a machine with the time of work of a machine without restrictions on the register length.

**Basic Definitions**

A machine with arbitrary access to memory (in accord with the convention made in [3] we shall call it an RAM) is an abstract model of calculations – cf. [2, 3]. Its memory is an unbounded sequence of registers, indexed by the natural numbers $0, 1, 2, \ldots$, each of which can contain an arbitrary integer. We take a system of instructions as in [3]. This system consists of the instructions to enter input information into the memory, to summon the content of a register onto the summator (register with index 0), to send with indirect address the content of the summator to the memory, to add, multiply, subtract, divide, to branch, and to stop. We shall particularly stipulate the case when we consider a machine without any of these operations (e.g., without multiplication and division).

By the working time of an RAM we shall mean the number of operations performed from the time the machine starts until it stops. (If the machine does not stop, then the working time on the given input is undefined.)

We consider the question of the recognition of sets of words in the finite alphabet $\{a_1, \ldots, a_N\}$. To each letter $a_j$ we encode the number $j$ and we consider each word $a_{j_1} \ldots a_{j_k}$ in this alphabet as a sequence (massif) of numbers $j_1, \ldots, j_k$. We denote the length of the word $W$ by $|W|$.

We shall say that an RAM recognizes the set of words $\mathcal{F}$ in the alphabet $\{a_1, \ldots, a_N\}$ in time $t(a)$ if upon feeding it the input of the massif $j_1, \ldots, j_n$ it stops and prints 1 or 2 depending on whether or not the word $a_{j_1} \ldots a_{j_n}$ belongs to the set $\mathcal{F}$; while, starting with some number $n_0$, its working time on any massif of $n \geq n_0$ numbers does not exceed $t(n)$. In the case of the answer 1, we shall say that the machine admits this word, and in the case of the answer 2, it discards it.
In what follows, by a function will be meant an RAM mapping \( \mathbb{N} \) into \( \mathbb{N} \). The working time of the RAM \( T \) upon input \( n \) will be denoted by \( T(n) \).

We denote by \( CT \) the class of all sets of words in the alphabet \( \{0, 1\} \), for each of which there exists an RAM, recognizing it in time \( T(n) \).

**Basic Theorem.** For any function \( T: \mathbb{N} \rightarrow \mathbb{N} \) there exists a set of words \( \delta \) in the alphabet \( \{0, 1\} \), recognizable by some RAM in time \( 21T(n) + 6T(n) + 25n \), and such that for any RAM recognizing \( \delta \), the set of those words \( W \), on which the working time of this RAM is greater than \( T(|W|) \), is infinite.

The proof of this theorem is based on the method of "diagonalization." For this, we introduce, similarly to the way this was done in [1], the concept of code of a program. By the code of a program we shall understand the word obtained by successive assignment of codes of instructions, where the code of each instruction is a word consisting of an operation code and an operand code. Here instructions of one type, having different types of operands, will have different operation codes, as was done in [3]. For example, we distinguish the instructions ADDi and ADD•i.

We do not impose any restrictions on the coding of operations and operands other than the condition of the single-valuedness of their decoding. We shall denote the code of the program \( P \) by \( W_P \).

We note that a program is unchanged if at its end one writes some stopping instructions (HALT). Hence, in particular, equivalent programs can have arbitrarily long encodings.

Now we define the set \( \delta \) as follows:

\[
\delta = \{ W: \text{for some program } P \text{ one has } W = W_P, \text{ and } P \text{ either discards } W_P \text{ or works on } W_P \text{ longer than } T(|W_P|) \}.
\]

We shall show that any machine \( P_\delta \), recognizing this set, works a longer time than \( T(|W|) \) on infinitely many words \( W \). For a given program \( P_\delta \) we consider the infinite sequence of programs equivalent to it obtained by adding to its end several stopping instructions \( P_1, P_2, \ldots, P_k, \ldots \), and the sequence of their codes \( W_{P_1}, W_{P_2}, \ldots, W_{P_k}, \ldots \). We shall show that on any word \( W_{P_j} \) from this sequence \( P_\delta \) works longer than \( T(|W_{P_j}|) \). Let us assume the contrary: let the working time of \( P_\delta \) on the word \( W_{P_j} \) not exceed \( T(|W_{P_j}|) \). Then by the construction of \( \delta \), we have:

\[
W_{P_j} \in \delta \implies P_\delta \text{ admits } W_{P_j} \text{ and the working time } \leq T(|W_{P_j}|)
\]

\[
\implies W_{P_j} \notin \delta \implies W_{P_j} \in \delta
\]

\[
W_{P_j} \in \delta \implies P_\delta \text{ discards } W_{P_j} \implies P_\delta \text{ discards } W_{P_j} \implies W_{P_j} \notin \delta
\]

In both cases we arrive at a contradiction, which means our assumption is false, and \( P_\delta \) on the word \( W_{P_j} \) from this sequence works longer than \( T(|W_{P_j}|) \).

To prove the first assertion of the theorem, we write a program for an RAM, recognizing the set \( \delta \) in the time indicated. The program will consist of two basic parts: "translator" and "processor." The "translator" is intended for carrying out the decoding of the introduced word \( W \), i.e., finding a machine \( P_\delta \), such that \( W = W_{P_\delta} \). The "translator" also reduces the memory of the RAM to the form indicated below and calculates the function \( T(|W|) \). In the case of impossibility of decoding \( W \) the "translator" discards the word and stops.

The structure of the memory after the work of the "translator" is this: registers of the form \( 5k \) (\( k = 0, 1, \ldots \)) correspond to registers with the indices \( k \) for the machine \( P_\delta \); registers of the form \( 5k + 1 \) and \( 5k + 3 \) (\( k = 1, 2, \ldots \), \(|W|\)) contain the code and address of the operand of the \( k \)-th instruction of \( P_\delta \); registers of the form \( 5k + 2 \) (\( k = 1, 2, \ldots \)) are used for the work of the "processor"; registers of the form \( 5k + 4 \) (\( k = 0, 1, \ldots \), \(|W|\)) contain the code of the letters with index \( k \) in the input word \( W \).

For reasonable coding the working time of the "translator" is a linear function of the length of the input word. By choosing a sufficiently simple method of coding one can assure that the working time of the decoding will not exceed \( 25n \).

The "translator" has to calculate the function \( T(|W|) \), using not all the registers, but only certain ones (e.g., only each tenth), hence the calculation is complicated, which leads to an increase of the time necessary for calculating \( T(n) \). It is easy to organize the work of the "translator" so that it requires time \( 6T(n) \) to calculate \( T(n) \).

Thus, the total working time of the "translator" does not exceed \( 6T(n) + 25n \).