An Optimal Lower Bound for Turing Machines with One Work Tape and a Two-way Input Tape

by

Wolfgang Maass[*,**]

and

Georg Schnitger[***]

Research Report in Computer Science, No. 7, Sept. 1985

[*]Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, Illinois 60680.

[**]Supported in part by NSF Grant DCR-8504247.

[***]Department of Computer Science, Pennsylvania State University, University Park, Pennsylvania 16802.

An Optimal Lower Bound for Turing Machines with One Work Tape and a Two-way Input Tape

by

Wolfgang Maass*
Department of Mathematics, Statistics and Computer Science
University of Illinois at Chicago, Chicago, Illinois 60680

and

Georg Schnitger
Department of Computer Science
Pennsylvania State University, University Park, Pennsylvania 16802

ABSTRACT

This paper contains the first concrete lower bound argument for Turing machines with one work tape and a two-way input tape (these Turing machines are often called "offline 1-tape Turing machines"). In particular we prove an optimal lower bound of $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ for transposing a matrix with elements of bit length $\theta(\log n)$ (where n is the length of the total input). This implies a lower bound of $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ for sorting on the considered type of Turing machine. We also get as corollaries the first nonlinear lower bound for the most difficult version of the two tapes - versus - one problem, and a separation of the considered type of Turing machine from that with an additional write-only output tape.

§1. INTRODUCTION

This paper is part of the project to develop lower bound techniques for increasingly more powerful types of restricted Turing machines (TM's). The current state of affairs with regard to lower bound results for TM's is as follows. Besides the well known hierarchy theorems for several types of complexity classes (see [4]) and the separation of DTIME(n) and DSPACE(n) by Hopcroft, Paul and Valiant [5] (they show that DSPACE(n) $\notin$ DTIME(o(n · log n))), Paul, Pippenger, Szemeredi and Trotter [11] have more recently shown that DTIME(n) $\neq$ NTIME(n) (in fact that NTIME(n) $\notin$ DTIME(o(n · $(\log^* n)^{1/4}$))). Both of these results are consequence of relatively abstract graph theoretical facts. Although these methods are very elegant, there are some indications that such methods will not yield substantially larger lower bounds for multitape TM's (in particular not beyond the $\Omega(n \cdot \log n)$-range).

*Supported in part by NSF Grant DCR-8504247.

Therefore it is desirable to develop in addition more concrete lower bound techniques, that analyze the progress of a TM-computation on a specific computational problem. However if one attempts with the presently available methods such fine structure analysis of computations on two-tape TM's, one is paralyzed by the enormous number of diverse computational strategies which the TM might pursue. Furthermore one sees immediately, that there are much more restricted types of Turing machines (e.g., where one of the two tapes is only a read - only input tape) for which no lower bound argument for a concrete computational problem is available. Thus there is little hope of succeeding at this state of knowledge with a concrete lower bound argument for two-tape Turing machines.

On the positive side one could point out that one has succeeded over the last two decades in developing increasingly more clever lower bound methods for restricted types of TM's. Further this technology appears to be cumulative, i.e., lower bound methods for more primitive types of TM's have often provided essential ingredients for subsequently developed lower bound arguments for more complex types of restricted TM's.

An optimal quadratic lower bound for Turing machines with one work tape (no input tape) was shown by Hennie [3] (key tool of this argument: crossing sequences). Subsequently Paul et al. proved a number of lower bounds for "on-line simulation" by TM's (see [10], [1]). Here one considers very powerful types of TM's, but in the definition of "on-line simulation" the simulating TM is forced to output specified intermediate results before it can see the next input bit (this restricts the types of simulation algorithms that it could potentially use). These papers introduced and developed a powerful new tool: Kolmogorov complexity. This notion allows us to keep track of the flow of information in the computation on a single (suitably chosen) input. Although the use of Kolmogorov complexity could in principle be eliminated in favor of straightforward counting arguments, it simplifies the presentation so much that we can carry out arguments that would otherwise be unfeasible. In a next step the first author had shown ([6], [7], [8]) an optimal quadratic lower bound for the simulation of two-tape TM's by TM's with one work tape and a one-way input tape. In addition to the previously mentioned tools this proof used additional combinatorial arguments.

In this paper we carry the described development one step further: we present the first concrete lower bound argument for a TM with one work tape and a two-way input tape (sometimes these TM's are called

"off-line 1-tape TM's"). We prove in §2 of this paper for TM's with one work tape and a two-way input tape an optimal lower bound of $\Omega(\ell^3 \cdot p)$ for the problem of transposing an $\ell \times \ell$ matrix whose elements have bit length p. In particular in the case of matrices with elements of bit length $\theta(\log n)$, where n is the length of the total input, this yields an optimal lower bound of $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ for the considered problem.

Since matrix transposition can be reduced to sorting, we also get a lower bound of $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ for sorting on the considered type of TM (Theorem 3.1). Finally we achieve further progress with regard to the well-known two-tapes - versus - one problem, which has been solved in [7] except for the strongest possible formulation of this problem, where the considered 1-tape TM's are equipped with an additional <u>two</u>-way input tape (no nonlinear lower bound was previously known for this version of the problem, the best upper bound is $O(n^2)$). We show in §3 that there are in fact functions that can be computed in linear time on a two-tape TM, but which require time $\Omega\left(\frac{n^{3/2}}{(\log n)^3}\right)$ on the considered type of one-tape TM.

In §4 we use the preceding lower bound result in order to exhibit a significant difference between the computational power of the considered model of an offline 1-tape TM and the variation of this model which has an additional one-way write-only output tape.

The lower bound argument in §2 is a combination of two different strategies. In the case where the considered TM executes the matrix transposition in a relatively straightforward manner, the combinatorial structure of the problem (i.e., matrix transposition) itself allows us to show that either the work head <u>or</u> the input head has to make $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ moves (there are in fact algorithms for matrix transposition on the considered type of TM where the work head makes only $O(n)$ moves). In the other case we can show via Kolmogorov complexity that the TM can realize the required flow of information in the computation only if the work head makes $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ moves.

A key point of this lower bound argument is the fact that the permutation τ of the ℓ^2 matrix elements that is executed by the matrix transposition function requires $\Omega(\ell)$ sweeps of both TM-heads. It is quite tempting to try to prove stronger lower bounds for sorting on the considered type of TM by focusing on other permutations τ' that possibly require even more sweeps. We show in §5 that this approach is not feasible: there is no permutation τ' that requires for its reali-

zation more sweeps than the here considered permutation τ. We use for this observation a combinatorial result by Erdös and Szekeres [2].

We use the following definitions and conventions in this paper. An input-tape for a TM is a read-only tape (with one head) that contains the input (with endmarkers at both ends of the input). We call the input-tape one-wway or two-way, depending on whether the input head may only move in one direction, or whether it has no restriction on its movement (sometimes one calls TM's with a one-way input tape "online" and TM's with a two-way input tape "offline").

We assume that all TM's (with any number of tapes) are coded in some fixed way by binary sequences. We write $|M|$ for the length of the binary sequence that codes TM M. One defines the Kolmogorov complexity of a string X relative to another string Y by

$$K(X|Y) := \min\{|M| \mid M \text{ is a deterministic TM that gives for input } Y \text{ the output } X\}.$$

The Kolmogorov complexity of a string X is defined by

$$K(X) := K(X|\epsilon) \quad (\epsilon \text{ is the empty string}).$$

Obviously there is for every natural number n a binary string of length n with $K(X) \geq n$.

We use in this paper the notation $t(n) = \Omega(f(n))$, if there is some constant $c > 0$ such that $t(n) \geq c \cdot f(n)$ for infinitely many natural numbers n.

§2. THE LOWER BOUND FOR MATRIX TRANSPOSITION.

We consider in the following the matrix transposition function MATRIX TRANSPOSITION$_{\ell,p}$, which assigns to an $\ell \times \ell$ matrix $A = (a_{ij})_{i,j \leq \ell}$ (with elements a_{ij} of bit length p, given row by row as a linear sequence) the transposed matrix $A^t = (a_{ji})_{i,j \leq \ell}$ (in the same representation). Thus if we apply this function to a sequence $\langle b_1, \ldots, b_{\ell^2} \rangle$, the value has the form $\langle b_{\tau(1)}, \ldots, b_{\tau(\ell^2)} \rangle$, where τ is a permutation of $\{1, \ldots, \ell^2\}$ which maps $(i-1) \cdot \ell + j$ onto $(j-1) \cdot \ell + 1$ $(i,j \in \{1, \ldots, \ell\})$. This permutation τ is of particular interest for our lower bound argument, because it scatters the elements of its domain particularly well: for any $m \in \{1, \ldots, \ell^2 - 1\}$ one has $|\tau(m) - \tau(m+1)| \geq \ell$. Further, its inverse τ^{-1} has the same property (since $\tau^{-1} = \tau$).

We would like to mention that this permutation τ has also been studied by Paul [9] and Stoss [12]. They considered a different type of computational model, which was not allowed to perform "magic" (i.e.,

code and decode information, see [9] for a definition). Of course the considered type of TM is able to perform "magic" in this sense.

In the following definition of the function MATRIX TRANSPOSITION$_{s,p}$ (where p is the number of bits in a single entry a_{ij}) we add the indices i,j to the elements a_{ij} of the rowwise representation of the matrix $(a_{ij})_{i,j \le s}$. This makes the proof of the upper bound more straightforward, while it does not affect the lower bound argument. Also it is easier to reduce this version of the problem to sorting. The input of MATRIX TRANSPOSITION$_{s,p}$ is an element of $\{0,1,B\}^{s^2 \cdot (2(\log s)+p+5)}$, where B (blank) serves as a separator between entries (for convenience we always assume that s is a power of 2). Thus a matrix $(a_{ij})_{i,j \le s}$ with $a_{ij} \in \{0,1\}^p$ is given as the concatenation of strings of the form $bin(i) \frown B \frown bin(j) \frown B \frown a_{ij} \frown B$ (in lexicographical order according to the sequences $bin(i) \frown bin(j)$). We write here $bin(i)$ for the binary representation of the natural number i (for convenience we add leading 0's so that all binary strings $bin(i)$ for $i \in \{1,\dots,s\}$ have exactly length $1+\log s$).

The output of MATRIX TRANSPOSITION for the described input is also an element of $\{0,1,B\}^{s^2 \cdot (2(\log s)+p+5)}$: the concatenation of the sequences $bin(i) \frown B \frown bin(j) \frown B \frown a_{ji} \frown B$, again arranged according to the lexicographical order of $bin(i) \frown bin(j)$. Thus if we write the entries a_{ij}, $i,j \le s$, rowwise as a linear sequence $b_1,\dots,b_{s^2}$, then for each $m \in \{1,\dots,s^2\}$ the binary string b_m in the input string is replaced in the output string by $b_{\tau(m)}$.

THEOREM 2.1.

The time complexity of MATRIX TRANSPOSITION on Turing machines with one work tape and a two-way input tape is precisely $\Theta(s^3 \cdot p) = \Theta(n \cdot s)$ (where n is the length of the input). In particular the time complexity of transposing an $s \times s$ matrix with entries a_{ij} of bit length $p = (1+\epsilon) \cdot \log s$ (where $\epsilon > 0$ is any constant) on such a machine is $\Theta\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ (n is the bit length of the input).

REMARK 2.2. The lower bound argument works already if the Turing machine computes MATRIX TRANSPOSITION$_{s,p}$ only for certain specific parameters $s(n)$, $p(n)$, provided that $p(n) \ge (1+\epsilon) \cdot \log s(n)$ (with $\epsilon > 0$). For example for the $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ bound it is sufficient to assume that the Turing machine computes the matrix transposition only for matrices whose elements have $\log n$ bits (where n is the length

of the input).

Note that on the other hand we construct in the upper bound proof a single Turing machine of the considered type that performs matrix transposition for arbitrary values of the parameters ℓ, p.

PROOF OF THEOREM 2.1: We first construct a TM M with one work tape and a two-way input tape that computes MATRIX TRANSPOSITION$_{\ell,p}$ in time $O(n \cdot \ell)$. M begins by copying a variation of the input string on its work tape, where every entry a_{ij} is replaced by blanks (this takes time $O(n)$). Then with ℓ further simultaneous sweeps over the input tape and the considered part of the work tape M brings each entry a_{ij} from the input tape into its proper position after the substring $bin(j) \frown B \frown bin(i) \frown B$ on the work tape. Since a_{ij} is preceded on the input tape by $bin(i) \frown B \frown bin(j) \frown B$, M can find this proper position via trivial pattern matching. Obviously M needs only $O(n)$ steps to bring all entries from one row of the matrix $(a_{ij})_{i,j \leq \ell}$ into their proper position.

For the lower bound argument we fix some $\epsilon > 0$ and a TM M of the considered type that computes MATRIX TRANSPOSITION$_{\ell,p}$, where $p \geq (1+\epsilon) \cdot \log \ell$. We fix a value of ℓ that is sufficiently large relative to the length $|M|$ of the description of TM M (precise conditions arise from the following calculations with Kolmogorov complexity; the goal is to make sure that the information that is contained in the program of M can be neglected in comparison with the large amount of information in the considered input). For convenience we choose ℓ to be a power of 2.

Let X be a binary sequence of length $\ell^2 \cdot p$ that is Kolmogorov random, i.e., $K(X) \geq |X|$. We partition X into ℓ^2 segments $b_1, \ldots, b_{\ell^2}$ of length p. In the following we analyze the computation of M on that input I of length $n = \ell^2 \cdot d$ with $d := p + 2 \cdot \log \ell + 3$, where $b_1, \ldots, b_{\ell^2}$ are the elements (in rowwise order) of the $\ell \times \ell$ matrix $(a_{ij})_{i,j \leq \ell}$ that is to be transposed.

We now analyze the computation of M on this input I and show that its length T is at least $K_\epsilon \cdot n \cdot \ell$, where the constant K_ϵ depends only on the constant $\epsilon > 0$. We partition the input tape of M into segments $A_1, \ldots, A_{\ell^2}$ of d cells each (thus each A_m holds exactly one sequence $bin(i) \frown B \frown bin(j) \frown B \frown a_{ij} \frown B$). Analogously we partition that interval of length n on the work tape which holds at the end of the computation the output into segments $B_1, \ldots, B_{\ell^2}$ of length d.

We define S as the sequence of all pairs $\langle k,m\rangle \in \Lambda^2 \times \Lambda^2$ s.t. at some step t of the considered computation the input head is in tape segment A_k while the work head is simultaneously at step t in tape segment B_m. We assume that these pairs occur in the sequence S in the same order as the time points t where they are "realized" in the computation. Further we assume that no pair occurs in S twice in immediate succession (but apart from this every pair occurs in S as often as it is "realized" in the computation).

The basic idea of the lower bound proof is to use two completely different arguments, depending on whether or not the positions of the two heads during the computation allow for most of the a_{ij} a direct copying from its position in the input to its final position in the output. However in the second case where the majority of the a_{ij} is not copied directly one can only achieve the optimal lower bound if one knows that in fact the majority of the a_{ij} is not even copied from the input tape into the neighborhood of their final position. Therefore we use the size of the following set D as the criterion for the partition into the two cases:

$$D := \{k \mid k \in \{1,\ldots,\Lambda^2\} \text{ and some pair } \langle k,m\rangle \text{ occurs in } S \text{ with } |m - \tau(k)| \leq \Lambda\}.$$

CASE I: $|D| \geq \frac{\Lambda^2}{2}$.

Consider any interval V of S, where at least for four different $k \in \{1,\ldots,\Lambda^2\}$ some pair $\langle k,m\rangle$ with $|m - \tau(k)| \leq \Lambda$ occurs in V. We exploit the combinatorial structure of the permutation τ to derive in the following simple combinatorial lemma a lower bound on the length of V. Set $\pi_1(\langle x,y\rangle) = x$ and $\pi_2(\langle x,y\rangle) = y$.

LEMMA 2.3. Assume that V is a sequence of elements from $\Lambda^2 \times \Lambda^2$ with $|\{x \in \Lambda^2 \mid \langle x,y\rangle \in V \text{ for some } y \in \Lambda^2 \text{ with } |y - \tau(x)| \leq \Lambda\}| \geq 4$. Then $\max \pi_1[V] - \min \pi_1[V] \geq \Lambda$ or $\max \pi_2[V] - \min \pi_2[V] \geq \Lambda$.

PROOF: Assume for a contradiction that this is not the case. This implies that for any two different elements x,x' of $\pi_1[V]$ we have $|x-x'| < \Lambda$ and therefore $|\tau(x) - \tau(x')| \geq \Lambda$. Thus, since also $|y - y'| < \Lambda$ for $y,y' \in \pi_2[V]$, there are at most three different $x \in \pi_1[V]$ with $\langle x,y\rangle \in V$ for some y with $|y - \tau(x)| \leq \Lambda$. This is a contradiction to the assumption of the lemma.

The preceding lemma implies that during the time interval T_V of the computation of M that corresponds to the interval V of S one of the heads of M runs over at least $\ell - 2$ tape segments of length d. Thus T_V consists of at least $(\ell - 2) \cdot d$ computation steps. Since $|D| \geq \frac{\ell^2}{2}$, S contains at least $\frac{\ell^2}{8}$ disjoint segments V as above. This implies that

$$T \geq \frac{\ell^2}{8} \cdot (\ell-2) \cdot d$$

$$= \frac{1}{8} n \cdot (\ell-2) .$$

CASE II: $|D| < \frac{\ell^2}{2}$.

Let P be the set of tape segments B_r such that at every step of the considered computation where the input head is in tape segment $A_{\tau^{-1}(r)}$ $(= A_{\tau(r)})$ the work head has distance $\geq d \cdot \ell$ from all cells in B_r. By the assumption of this case we have $|P| \geq \frac{\ell^2}{2}$.

We now partition the sequence $B_1, \ldots, B_{\ell^2}$ of tape segments into 2ℓ blocks $\tilde{B}$, where each block $\tilde{B}$ consists of $\frac{\ell}{2}$ consecutive tape segments. Since on average every second tape segment belongs to P, there are at least $\frac{\ell}{2}$ blocks $\tilde{B}$ in which at least one fourth of the tape segments belong to P (i.e., $\tilde{B}$ contains at least $\frac{\ell}{6}$ tape segments from P). We fix for the following a set Q of these blocks $\tilde{B}$, such that any two blocks in Q are separated by at least two blocks in between that do not belong to Q (thus any two blocks in Q have a distance of $\leq \ell \cdot d$ cells). We choose Q to be of size $\frac{\ell}{8}$.

We will show that for every block $\tilde{B}$ in Q the TM M spends $\Omega(\ell^2 \cdot d)$ steps in $\tilde{B}$ and the two adjacent blocks. Thus fix some $\tilde{B} \in Q$. We write U for the set of the first $\frac{\ell}{8}$ tape segments in $\tilde{B}$ that belong to P. In a slight abuse of notation we write $U^\cap$ for the concatenation of the final contents of the tape segments in U (in the order of these segments on the work tape of M).

In the following we consider for certain cells u on M's work tape the crossing sequence C_u for u, which records for every crossing of u the current state of M and the current position of M's input head (in binary code). We write $\#C_u$ for the number of crossings that are recorded in C_u, and $|C_u|$ for the bit-length of C_u (obviously one has $|C_u| \leq \#C_u \cdot (C_M + \log n)$, with a constant C_M that depends on the number of states in M). Let $\tilde{B}_1 (\tilde{B}_2)$ be the block immediately to the left (right) of the fixed block $\tilde{B} \in Q$. For $i = 1.2$

choose a cell c_1 in $\tilde{B}_1$ such that $\#C_1$ for the crossing sequence C_i at c_i is as small as possible.

Let H be the interval between the cells c_1 and c_2. Note that the fixed block $\tilde{B}$ is contained in H. It is obvious that the final content of tape interval H can be computed from C_1, C_2, those parts of the input that are inspected by the input head while the work head is inside H, the length of H, and a description of the TM M. This implies that $U^{\cap}$ can be computed from $C_1, C_2, (A_U)$ (the sequence of "addresses" of the tape segments from U inside H; in order to be concise we just specify for any two consecutive tape segments from U the number of tape segments that lie in between), I^c (a variation of the input I where for each tape segment A_k with $B_{\tau(k)} \in U$ the matrix element a_{ij} in A_k has been "censored", i.e., replaced by an equally long string from $\blacksquare^*$ - where $\blacksquare$ is a new symbol), the length of interval H, and the program of M. We use here the fact that each time the input head of M is in a part A_k of the input I that has been censored (i.e., in some A_k with $B_{\tau(k)} \in U$), the work head of M is at least $d \cdot \ell$ cells away from $B_{\tau(k)}$ and is therefore located outside of the tape interval H.

The described situation can now be used to derive a lower bound on $(\#C_1) + (\#C_2)$ for the crossing sequences C_1 and C_2. The preceding observation implies that

$$(1) \qquad K(U^{\cap} \mid I^c) \leq |C_1| + |C_2| + |A_U| + \lceil \log |H| \rceil$$

$$\leq |C_1| + |C_2| + \frac{\ell}{8} \cdot \left(1 + \log\left(\frac{3/2 \cdot \ell}{\ell/8}\right)\right) + \log n$$

$$= |C_1| + |C_2| + \frac{\ell}{8}(1 + \log 12) + \log n$$

(we have used here that the $\ell/8$ differences of tape segment numbers that serve as "addresses" for the tape segments of U add up to at most $3/2 \cdot \ell$; further we use the convexity of the log - function).

We adhere to the usual convention of ignoring additive constants like $|M|$ that do not grow with the size of the input. This can be justified by making the input sufficiently long. Further it follows from the definition of the considered input I (in particular from the fact that the matrix elements a_{ij} in I form a Kolmogorov random string X of length $\ell^2 \cdot p$) that

$$(2) \qquad \ell^2 \cdot p \leq K(X) \leq K(I) \leq K(I^c) + K(U^{\cap} \mid I^c).$$

Since $|U| = \frac{\ell}{8}$ we have

$$(3)\qquad K(I^c) \le \ell^2 \cdot p - \frac{\ell}{8}p + \frac{\ell}{8}\left(1 + \log\left(\frac{\ell^2}{\ell/8}\right)\right) + \log \ell$$

$$= \ell^2 \cdot p - \frac{\ell}{8}p + \frac{\ell}{8} \cdot (5 + \log \ell)$$

(we use here that the locations of the $\ell/8$ censored strings in I^c can be described in a concise way by the number of tape segments in between; these differences add up to at most ℓ^2, thus we can use again the convexity of the log-function). (2) and (3) together yield

$$(4)\qquad \frac{\ell}{8}p - \frac{\ell}{8}(5 + \log \ell) \le K(U^{\cap} \mid I^c).$$

(1) and (4) imply (for sufficiently large ℓ)

$$\frac{\ell}{8}p - \frac{\ell}{8}(5 + \log \ell) \le (\#C_1 + \#C_2) \cdot (C_M + \log n) + \frac{5}{8}\ell + \log n.$$

Thus we get for large ℓ

$$\frac{1}{\sqrt{1+\epsilon}} \cdot \frac{\ell}{8}p - \frac{\ell}{8}\log \ell \le (\#C_1 + \#C_2) \cdot (C_M + \log n).$$

This implies (together with the assumption $p \ge (1+\epsilon) \cdot \log \ell$) that

$$\#C_1 + \#C_2 \ge \frac{\ell}{8} \cdot \frac{\left(\frac{p}{\sqrt{1+\epsilon}} - \log \ell\right)}{C_M + 2\log \ell + 2 + \log p}$$

$$\ge \frac{\ell}{8} \cdot \frac{\left(\frac{p}{\sqrt{1+\epsilon}} - \log \ell\right)}{4p}$$

$$= \frac{\ell}{8} \cdot \frac{\left(\frac{p}{K \cdot \sqrt{1+\epsilon}} + \frac{K-1}{K} \cdot \frac{1}{\sqrt{1+\epsilon}} \cdot p - \log \ell\right)}{4p}$$

$$= \frac{\ell}{8} \cdot \frac{\left(\frac{p}{K \cdot \sqrt{1+\epsilon}} + \frac{1}{1+\epsilon} \cdot p - \log \ell\right)}{4p} \ge \frac{\ell}{32K\sqrt{1+\epsilon}}$$

for $K := \dfrac{\sqrt{1+\epsilon}}{\sqrt{1+\epsilon} - 1}$ $\left(\text{thus } \dfrac{K-1}{K} = \dfrac{1}{\sqrt{1+\epsilon}}\right)$.

Since each block consists of $\frac{\ell}{2} \cdot d$ cells, our choice of the cells c_1 and c_2 together with the preceding lower bound on $\#C_1 + \#C_2$ imply that M spends at least $\dfrac{\ell}{32K\sqrt{1+\epsilon}} \cdot \frac{\ell}{2} \cdot d$ steps in $\tilde{B}$ and the two adjacent blocks. Since this holds for any of the $\ell/8$ blocks $\tilde{B}$ in Q, and since any two blocks in Q have a "buffer" of at least two blocks in between, we can conclude that M uses in Case II at least

$$\frac{\ell^3 \cdot d}{32 \cdot 16K\sqrt{1+\epsilon}} = \frac{n \cdot \ell}{32 \cdot 16K\sqrt{1+\epsilon}} \quad \text{steps.}$$

This completes the proof of Theorem 2.1.

§3. IMPLICATIONS FOR LOWER BOUNDS ON SORTING AND THE TWO-TAPES - VERSUS - ONE PROBLEM.

Obviously one can use any sorting algorithm in order to transpose a matrix. Therefore lower bounds for matrix transposition imply lower bounds for sorting. Further, since a Turing machine with 3 tapes (actually 3 pushdown stores are sufficient) can sort via merge-sort in $O(n \log n)$ steps, we get lower bounds for the simulation of multi-tape Turing machines by Turing machines with one work tape and a two-way input tape (here "simulation" just requires that it computes the same function). No nonlinear lower bounds for this simulation were previously known. The upper bound is $O(n^2)$, due to Hartmanis and Stearns (see [4]).

THEOREM 3.1. Sorting requires $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ steps on a Turing machine with one work tape and a two-way input tape (where n is the bit length of the input; one sorts $\Theta(n/\log n)$ binary strings of length $\Theta(\log n)$).

PROOF: In the lower bound argument of Theorem 2.1 we considered an input that consisted of $\ell(n)^2$ substrings of the form

$$\text{bin}(i) \frown B \frown \text{bin}(j) \frown B \frown a_{ij} \frown B,$$

with binary strings a_{ij} of length $p(n)$. We now consider a variation of this input for the sorting problem, where each such substring is replaced by $\text{bin}(j) \frown 0 \frown \text{bin}(i) \frown 0 \frown a_{ij} \frown B$. When one sorts these binary substrings, then the $((i-1) \cdot \ell + j)$-th substring in the input becomes the $((j-1) \cdot \ell + i)$-th substring in the output sequence. In particular the string a_{ij} is moved from its position in the input string to its position in the output string in exactly the same way as in the matrix transposition problem. Since the lower bound argument in Theorem 2.1 depended only on this movement of the substrings a_{ij}, it also applies to the considered sorting problem. In particular for substrings a_{ij} of length $p(n) = (1+\epsilon) \cdot \log \ell(n)$ (where $\epsilon > 0$ is any constant) we get again a lower bound of $\Omega\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ computation steps on the considered type of Turing machine.

THEOREM 3.2. There are functions that can be computed in linear time on a Turing machine with two tapes (no additional input tape is needed, actually one tape plus one pushdown store is enough), but that require

time $\Omega\left(\frac{n^{3/2}}{(\log n)^{9/2}}\right)$ on a Turing machine with one work tape and a two-way input tape.

PROOF: A TM M_1 with 3 tapes (actually 3 pushdown stores are enough) can execute the familiar merge-sort algorithm in time $O(n \cdot \log n)$ (where n is the bit length of the input string). The standard technique of Hennie and Stearns (see [3]) allows to simulate M_1 by a Turing machine M_2 with 2 tapes, which sorts in time $O(n \cdot \log^2 n)$ (a straightforward analysis shows that this simulation remains valid for the computation of functions where the output appears at the end of the computation on a work tape of M_2). A closer look reveals that M_2 needs actually only one work tape and a pushdown store (no input tape) for this simulation.

One now applies the usual padding-technique to get a function f with (essentially) the same output that requires only linear time instead of time $O(n \cdot \log^2 n)$ on a TM M_3 of the same type as M_2. One extends the input of length m for sorting by a string that consists of $m \cdot ((\log m)^2 - 1)$ copies of a new symbol $\#$. Obviously for this padded input (of length $n = \Theta(m \cdot \log^2 m)$) the same sorting algorithm applied to the first m input bits runs in time $O(n)$. Further the considered TM can check in linear time whether the input consists of a string from $\{0,1,B\}^m$ followed by at least $m \cdot ((\log m)^2 - 1)$ copies of $\#$ (for some natural number m). If the input is not of this form, we let the TM not apply the sorting algorithm, but print immediately the output 0 (we define 0 as the value of the computed function f for such inputs, otherwise the value of f is the result of sorting the binary numbers that are given in the initial part of the input). M_3 computes this function f in linear time.

Let M_4 be any TM with one work tape and a two-way input tape that computes the same function f. Let $t(n)$ be the maximal number of steps that M_4 needs for an input of length n. We will show that $t(n) = \Omega\left(\frac{n^{3/2}}{(\log n)^{9/2}}\right)$. For this purpose we simulate M_4 by a TM M_5 of the same type that sorts a sequence of binary numbers of total length m in $O(m \cdot \log^2 m + t(m \cdot \log^2 m) \cdot \log m)$ steps. By Theorem 3.1 this implies that there is some $\delta > 0$ such that $m \cdot \log^2 m + t(m \cdot \log^2 m) \cdot \log m \geq \delta \cdot \frac{m^{3/2}}{(\log m)^{1/2}}$ for infinitely many m. For $n = m \cdot \log^2 m$ this yields $t(n) = \Omega\left(\frac{n^{3/2}}{(\log n)^{9/2}}\right)$.

Inputs for M_5 are strings w from $\{0,1,B\}^*$ (sequences of binary numbers that are to be sorted). Everything would be easy for M_5, if it could extend its input by the string $\#^{m \cdot ((\log m)^2 - 1)}$, where $m = |w|$, and then simulate M_4 on this padded input. But M_5 is not allowed to write on its input tape, and therefore it has to work with a "virtual padding" of the input. For this purpose M_5 constructs at the beginning of its computation in $O(m \cdot \log^2 m)$ steps the string $\mathrm{bin}(m \cdot ((\log m)^2 - 1))$ on its work tape. Thus M_5 "knows" how long the padding ought to be. It then simulates M_4 on the input $w \# \#^{m \cdot ((\log m)^2 - 1)}$ as follows. Whenever the input head of M_4 enters the suffix $\#^{m \cdot ((\log m)^2 - 1)}$ of M_4's input, the input head of M_5 remains at the right end of its input tape. However the work head of M_5 carries along with it a binary counter where it keeps track of the current position of M_4's input head inside $\#^{m \cdot ((\log m)^2 - 1)}$. The work head of M_5 carries also along with it the previously constructed string $\mathrm{bin}(m \cdot ((\log m)^2 - 1))$. Thus M_5 recognizes easily (by comparing this string with the content of the mentioned binary counter), when the input head of M_4 has reached the "end-of-input" marker. In this way M_5 simulates each step of M_4 in $O(\log m)$ steps. By definition of the function f the output of M_4 consists of the binary numbers from w in sorted order. Thus M_5 sorts a sequence of binary numbers with total bit length m in $O(m \cdot \log^2 m + t(m \cdot \log^2 m) \cdot \log m)$ steps.

This completes the proof of Theorem 3.2.

COROLLARY 3.3. There are functions that can be computed in linear time on a Turing machine with three tapes (no additional input tape is needed, actually three pushdown stores are enough), but which require time $\Omega\left(\frac{n^{3/2}}{(\log n)^3}\right)$ on a Turing machine with one work tape and a two-way input tape.

PROOF: One uses the same argument as for Theorem 3.2 (with padding of length $\theta(m \cdot \log m)$ instead of $\theta(m \cdot \log^2 m)$). Actually the proof is somewhat simpler, since we need not consider an intermediate TM M_2 with 2 tapes. M_3 is a linear time TM with 3 pushdown stores that computes an analogous function f as before (of course M_3 is just another version of M_1, whose time bound is artificially lowered by padding the input). Let M_4 be a TM of time complexity $t(n)$ with one work tape and a two-way input tape that computes the same function f as M_3. Analogously as in the proof of Theorem 3.2 one shows with

the help of Theorem 3.1 that $m \cdot \log m + t(m \cdot \log m) \cdot \log m = \Omega\left(\frac{m^{3/2}}{(\log m)^{1/2}}\right)$. For $n = m \cdot \log m$ this implies that $t(n) = \Omega\left(\frac{n^{3/2}}{\log^3 n}\right)$.

REMARK 3.4. It is not too difficult to see that a 2-tape TM can execute matrix transposition for an input of bitlength n in $O(n \cdot \log n)$ steps (realize the butterfly-graph via the movement of both heads). In this way one can improve the lower bound of Theorem 3.2 (for some different function that can be computed in linear time on a 2-tape TM) for $\Omega\left(\frac{n^{3/2}}{(\log n)^{9/2}}\right)$ to $\Omega\left(\frac{n^{3/2}}{(\log n)^2}\right)$.

§4. SEPARATION OF OFFLINE 1-TAPE TURING MACHINES WITH AND WITHOUT OUTPUT TAPE.

In this section we consider offline 1-tape TM's with an additional write-only output tape. We show that even if the head on the extra output tape is required to move only in one direction ("one-way write-only output tape") the resulting TM is more powerful than the type that we have considered so far in this paper.

THEOREM 4.1. An offline 1-tape Turing machine with an additional one-way write-only output tape can compute MATRIX TRANSPOSITION$_{\ell,p}$ in $O(\ell^{5/2} \cdot p)$ steps. In particular for $p = (1+\epsilon) \cdot \log \ell$ this machine needs only $O\left(\frac{n^{5/4}}{(\log n)^{1/4}}\right)$ steps (as opposed to $\Theta\left(\frac{n^{3/2}}{(\log n)^{1/2}}\right)$ steps for the version without output tape; n is the bitlength of the input).

PROOF: The permutation τ of $\{1,\dots,\ell^2\}$ which maps $(i-1) \cdot \ell + j$ onto $(j-1) \cdot \ell + i$ can be written as $\tau = \rho \cdot \rho$, where ρ is the permutation of $\{1,\dots,\ell^2\}$ defined by

$$(i_1 - 1) \cdot \ell^{3/2} + (i_2 - 1) \cdot \ell + (j_1 - 1) \cdot \ell^{1/2} + j_2 \mapsto$$

$$(j_2 - 1) \cdot \ell^{3/2} + (i_1 - 1) \cdot \ell + (i_2 - 1) \cdot \ell^{1/2} + j_1$$

$$(\text{for } i_1, i_2, j_1, j_2 \in \{1,\dots,\ell^{1/2}\}).$$

On a TM with two tapes this permutation ρ can easily be realized: the head on the first tape executes $\sqrt{\ell}$ sweeps while the head on the second tape makes simultaneously one (slow) sweep. Therefore an offline 1-tape TM with an additional one-way write-only output tape can compute MATRIX TRANSPOSITION$_{\ell,p}$ in $O(\ell^{5/2} \cdot p)$ steps as follows: It first applies the permutation ρ to the given ℓ^2 matrix elements on the

input tape and writes this intermediate result on the work tape (this can be done in $O(\ell^{3/2} \cdot p)$ steps by the observation above). Subsequently the TM applies again the permutation ρ to the ℓ^2 matrix elements on the work tape and writes the result on the output tape (again this requires only $O(\ell^{5/2} \cdot p)$ steps, further the output head has to move only in one direction).

§5. NO PERMUTATION REQUIRES SUBSTANTIALLY MORE SWEEPS THAN MATRIX TRANSPOSITION.

The proof of Theorem 2.1 relied on the fact that matrix transposition corresponds to a permutation τ that "scatters" any adjacent elements in its domain. This gives rise to the question whether there are perhaps other permutations π that are even more difficult to "realize" by the head movement of the considered TM-model (and which might yield even larger lower bounds for this TM). We show in Proposition 5.1 that a well known result by Erdös and Szekeres implies a negative answer: every permutation π of m numbers can be realized by $O(\sqrt{m})$ simultaneous sweeps over the input tape and the output area on the work tape of an offline 1-tape TM. Thus provided that the machine "knows" already where each number has to be placed, an offline 1-tape TM can realize any permutation π of m numbers of bitlength p in time $O(m^{3/2} \cdot p)$ $(= O(n \cdot \sqrt{m})$ in terms of the input length $n = m \cdot p)$.

PROPOSITION 5.1. Any sequence $\langle x_1, \ldots, x_m \rangle$ of m pairwise different numbers (with arbitrary order) can be written as a disjoint union of $O(\sqrt{m})$ monotone (increasing or decreasing) subsequences.

PROOF: According to Erdös and Szekeres [2] any sequence of length $i^2 + 1$ contains a monotone subsequence of length $i + 1$. Thus any sequence of length j contains a monotone subsequence of length $\lfloor \sqrt{j} \rfloor$. We use this to show that any sequence of length m consists of $\leq 3 \cdot \lceil \sqrt{m} \rceil$ monotone subsequences.

We proceed by induction on $\lceil \sqrt{m} \rceil$. We apply two consecutive times the result by Erdös and Szekeres to the given sequence of length m. The first time we remove a monotone subsequence of length $\lfloor \sqrt{m} \rfloor$ and the second time a monotone subsequence of length $\lfloor \sqrt{m - \lfloor \sqrt{m} \rfloor} \rfloor$. Finally we remove another monotone subsequence of 2 elements. Then we have for the length k of the remaining sequence

$k = m - \lfloor\sqrt{m}\rfloor - \lfloor\sqrt{m - \lfloor\sqrt{m}\rfloor}\rfloor - 2 \leq m - 2\sqrt{m} + 1$. This implies that $\sqrt{k} \leq \sqrt{m} - 1$, thus $\lceil\sqrt{k}\rceil \leq \lceil\sqrt{m}\rceil - 1$. Therefore one can apply the induction hypothesis to the remaining sequence of length k.

REFERENCES

[1] P. Duris, Z. Galil, W.J. Paul, and R. Reischuk, Two nonlinear lower bounds, Proc. 15th ACM STOC (1983) 127-132.

[2] P. Erdös and G. Szekeres, A combinatorial problem in Geometry, Composito Math. 2(1935) 464-470.

[3] F.C. Hennie, One-tape off-line Turing machine computations, Inf. and Control 8(1965) 553-578.

[4] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley (Reading, 1979).

[5] J.E. Hopcroft, W.J. Paul, and L. Valiant, On time versus space, J. of the ACM 24(1977) 332-337.

[6] W. Maass, Are recursion theoretic arguments useful in complexity theory? Proc. of the 7th International Conference on Logic, Methodology and Philosophy of Science, Salzburg 1983 (North-Holland, Amsterdam 1985).

[7] W. Maass, Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines, Proc. of the 16th ACM STOC (1984) 401-408.

[8] W. Maass, Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines, Trans. of the Amer. Math. Soc. 292(1985) 675-693.

[9] W.J. Paul, Kolmogorov complexity and lower bounds, Proc. of the Second Int. Conf. on Fund. of Computation Theory, L. Budach ed., 325-334 (Akademie-Verlag, Berlin 1979).

[10] W.J. Paul, On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, Proc. 23rd IEEE FOCS (1982) 53-56.

[11] W.J. Paul, N. Pippenger, E. Szemeredi and W. Trotter, On determinism versus nondeterminism and related problems, Proc. 24th IEEE FOCS (1983) 429-438.

[12] H.J. Stoss, Rangierkomplexität von Permutationen, Acta Informatica 2 (1973) 80-96.