Theoretical Computer Science 82 (1991) 113–129
Elsevier

The complexity of matrix transposition on one-tape off-line Turing machines

Martin Dietzfelbinger**
FB Mathematik/Informatik, Universität-GH-Paderborn, D-4790 Paderborn, Germany

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

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

Communicated by R. Karp
Received November 1986
Revised July 1988

Abstract

Dietzfelbinger, M., W. Maass and G. Schnitger, The complexity of matrix transposition on one-tape off-line Turing machines, Theoretical Computer Science, 82 (1991) 113–129.

This paper contains the first concrete lower bound argument for Turing machines with one worktape and a two-way input tape ("one-tape off-line Turing machines"): an optimal lower bound of $\Omega(n \cdot l/\lceil(\log(l)/p)^{1/2}\rceil)$ for transposing an $l\times l$-matrix with elements of bit length p on such machines is proved. (The length of the input is denoted by n.) A special case is a lower bound of $\Omega(n^{3/2}/(\log n)^{1/2})$ for transposing Boolean $l\times l$-matrices. $(n=l^2)$ on such Turing machines. The proof of the matching upper bound (which is nontrivial for $p<\log l$) uses the fact that one-tape off-line Turing machines can copy strings slightly faster than if the straightforward method is used. As a corollary of the lower bound it is shown that sorting $n/(3\log n)$ strings of $3\log n$ bits each takes $\Omega(n^{3/2}/(\log n)^{1/2})$ steps on one-tape off-line Turing machines. Further corollaries give the first non-linear lower bound for the version of the two-tapes-versus-one problem concerning one-tape off-line Turing machines, and separate one-tape off-line Turing machines from those Turing machines with one input tape, one worktape, and an additional write-only output tape.

* Supported in part by NSF Grants DCR-850424 and CCR8703889.
** Supported in part by DFG Grants ME872/1-1 and WE 1066/1-2.

1. Introduction

This paper is part of the project to develop lower bound techniques for increasingly powerful types of restricted Turing machines (TMs).

The current state of affairs with regard to lower bound results for multitape TMs is as follows. Besides the well-known hierarchy theorems for several types of complexity classes (see [6]) and the separation of DTIME(n) and DSPACE(n) by Hopcroft, Paul and Valiant [7] (they showed that DSPACE(n) $\not\subseteq$ DTIME($o(n \log n)$)), Paul, Pippenger, Szemerédi and Trotter [17] showed more recently that DTIME(n) $\neq$ NTIME(n) (in fact, they showed that NTIME(n) $\not\subseteq$ DTIME($o(n(\log^* n)^{1/4})$)). The last two results are consequences 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 TMs (in particular not beyond the $\Omega(n \log n)$ range). 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 such a fine structure analysis of computations on two-tape TMs, using only the methods available at present, one is paralyzed by the enormous number of diverse strategies that the TM might pursue. Furthermore, there are restricted types of TMs (for example, those where one of the two tapes is 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 TMs.

On the other hand, during the last two decades increasingly clever lower bound methods for restricted types of TMs have been developed. These techniques appear to be cumulative, that means, lower bound methods for more primitive types of TMs have often provided essential ingredients for subsequently developed lower bound arguments for more complex types of restricted TMs. Essential steps in this development were the following. An optimal quadratic lower bound for TMs with one worktape (no input tape) was shown by Hennie [5] (key tool of this argument: crossing sequences). Subsequently, Paul [16] and Duris, Galil, Paul and Reischuk [3] proved a number of lower bounds for "on-line simulation" by TMs. Here very powerful types of TMs are considered, but, by the definition of "on-line simulation", the simulating TM has to output specified intermediate results before it can read the next input bit (this restricts the types of simulation algorithms it can employ). These papers introduced and developed a powerful new tool: *Kolmogorov complexity*. This notion makes it possible 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, by replacing it by direct counting arguments, this notion simplifies the presentation so much that arguments that would otherwise be infeasible can now be carried out. In a further step the second author showed an optimal quadratic lower bound for the simulation of two-tape TMs by

TMs with one worktape and a one-way input tape [9, 10, 11]. Besides the tools mentioned previously, this proof used additional combinatorial arguments. (For related results see Li, Longpré, Vitányi [8]; cf. also the remark at the end of the Introduction.)

In the present paper, we carry this development one step further: we present the first concrete lower bound argument for TMs with one worktape and a two-way input tape, in short "one-tape off-line TMs". In Section 2 we prove an optimal lower bound of $\Omega(n \cdot l / \lceil(\log(l)/p)^{1/2}\rceil)$ on the number of steps needed by such a TM for the problem of transposing an $l \times l$-matrix whose elements are bitstrings of length p. (The length of the input is always denoted by n. In the input the matrix is given in row major order; it is to be output in column major order.) A special case ($p=1$) of this result is an optimal lower bound of $\Omega(n^{3/2}/(\log n)^{1/2})$ for the problem of transposing quadratic Boolean matrices.

The matrix transposition function requires the TM to permute the elements of the input matrix according to the permutation τ of $\{1, \ldots, l^2\}$ defined by $\tau((i-1) \cdot l + j) = (j-1) \cdot l + i$ for $1 \leq i, j \leq l$. This permutation τ is well suited for our purposes since it "scatters" the elements of its domain particularly well: the images of adjacent elements in the domain are $l = \sqrt{n}$ positions apart, as are the preimages of elements adjacent in the range (since $\tau = \tau^{-1}$). The main effect of this fact is that it takes a TM with two heads $\Omega(l^3)$ steps to "realize" τ. (This is proved and used implicitly in the proof of the lower bound. We say that a TM with l^2 tape cells on each of its two tapes "realizes" τ during some sequence of moves if for each $m \in \{1, \ldots, l^2\}$ there is a time step t at which the first head scans cell m and the second head scans cell $\tau(m)$.) Already Paul [14] and Stoss [18] made use of this special property of τ in the context of lower bounds for a computational model with a storage structure similar to that of two-tape Turing machines.

The lower bound argument in Section 2 is a combination of two different strategies. In the case where the TM constructs the transpose of the input matrix in a straightforward manner, the combinatorial structure of the problem (that is, the "well-scattering" property of the permutation τ mentioned above) allows us to show that either the worktape head or the input tape head has to make $\Omega(n \cdot l)$ moves (for $p = \log l$, say). (In fact, there are algorithms for matrix transposition on TMs of the type considered here where the worktape 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 worktape head makes $\Omega(n \cdot l)$ moves (again, for $p = \log l$).

Remark 1.1. The main result of the present paper (Section 2, Theorem 2.4) is essentially the same as the main result of the preliminary version [12]; however, by using methods from [2], the lower bound argument has been simplified and sharpened, to work for a wider range of matrix transposition functions, in particular, to apply to the transposition of Boolean matrices.

It is obvious that the lower bound of Section 2 is optimal for $p=\Omega(\log l)$. For the case of smaller p (which was not considered in [12]), the matching upper bound is presented in Sections 3 and 4. (These new sections have been added to the preliminary version.) In Section 3 it is shown that the time required for copying short strings on the worktape of one-tape off-line TMs lies strictly between the time required for the same operation on one-tape TMs with a one-way input tape and on two-tape TMs. This fact indicates that one-tape off-line TMs are more powerful than one-tape on-line TMs, that is, those with a one-way input tape, even after the input has been read. (This phenomenon is discussed further in [1].) In Section 4 this "fast-copying trick" is employed to speed up the transposition of matrices with entries of length $p<\frac{1}{4}\log l$ (Theorem 4.1).

Using the fact that matrix transposition can be reduced to sorting, we obtain as a corollary of the main result a lower bound of $\Omega(n^{3/2}/(\log n)^{1/2})$ for sorting on one-tape off-line Turing machines (Section 5, Theorem 5.1). Moreover, the lower bound of Section 2 can be applied to make further progress with regard to the two-tapes-versus-one problem, which was solved completely in [10] for deterministic TMs except for the case of one-tape TMs with an additional two-way input tape. No nonlinear lower bound for the simulation of two-tape TMs on one-tape off-line TMs was previously known; the best known upper bound has been $O(n^2)$. (In [1] an upper bound of $O(n^2/\log n)$ is shown.) In Section 5 we show that some variant of the matrix transposition function can be computed in $O(n)$ steps on a two-tape TM, but requires time $\Omega(n^{3/2}/\log n)$ on one-tape off-line TMs (Theorem 5.2).

In Section 6, we use the main result in order to exhibit a significant difference between the computational power of one-tape off-line TMs and the variation of this model which has an additional write-only output tape (Theorem 6.1).

A key point in the lower bound argument is the fact that the "realization" of the permutation τ of the l^2 matrix elements induced by the matrix transposition function requires $\Omega(l)$ sweeps of at least one of the two tape heads. It is quite tempting to try to prove stronger lower bounds for sorting by considering other permutations π that possibly require even more sweeps. We show in Section 7 that this approach is not feasible: there is no permutation π that requires more sweeps for its realization than the permutation τ considered here. For this observation, we use a combinatorial result by Erdös and Szekeres [4].

The following definitions and conventions are used 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-way or two-way, according as the input head may move in one direction only or has no restriction on the direction of its movements. If the TM has a two-way input tape, it is also called "off-line".

In order to define the Kolmogorov complexity $K(X)$ of a string $X\in\{0,1\}^*$, we assume that all deterministic Turing machines (with any number of tapes) are coded by binary strings in some fixed, effective way, so that no code is a prefix of another one. Denote by $\lceil M \rceil$ the binary sequence that codes the TM M. Then define

$$K(X) := \min\{|\lceil M \rceil \frown u| \mid M \text{ a TM}, u \text{ a binary string so that } M \text{ on input } u \text{ gives output } X\}.$$

A string X is called "incompressible" if $K(X) \geq |X|$. Obviously, for every natural number n there is an incompressible string X of length n.

Remark 1.2. Since the preliminary version [12] of this paper has appeared, some further progress has been made in the general program outlined at the beginning of the Introduction. In [13] it was shown that two-tape TMs are more powerful than one-tape off-line TMs with regard to *decision problems.* (The decision problem considered there is closely related to matrix transposition.) However, the lower bound achieved there is only $\Omega(n \log n/\log\log n)$. Furthermore, a lower bound argument for one-tape off-line TMs *with output tape* has been developed [2] which shows that the upper bound of $O(n^{5/4})$ for computing the matrix transposition function on such machines (see Theorem 6.1) is in fact optimal.

2. The lower bound for matrix transposition

In the following, we consider the matrix transposition function, which transforms an $l \times l$-matrix $A = (a_{ij})_{1 \leq i,j \leq l}$ with entries $a_{ij} \in \{0, 1\}^p$ into the transposed matrix $A^T = (a_{ji})_{1 \leq i,j \leq l}$, for all natural numbers l and p. In order to have inputs and outputs in a format suitable for Turing machines, we need to represent matrices as strings of symbols. There are two natural ways for this, one "without addresses" and one "with addresses". Both list the entries of the matrix in row major order (that is, sorted according to $(l-1) \cdot i + j$), but in the latter, each entry a_{ij} is preceded by its "address" (i, j). The lower and upper bounds proved below apply to both representations. The version without addresses may seem more natural, in particular if $p < \log l$; on the other hand, the representation with addresses is more convenient for the applications in Section 5.

Let us describe the two representations more precisely. The representation of $A = (a_{ij})_{1 \leq i,j \leq l}$ "without addresses" is the string $A' \in \{0, 1\}^{l^2 p + p + 1}$ consisting of the string $0^p 1$ followed by the concatenation of the strings a_{ij} in row major order. The representation of A "with addresses" is the string $A'' \in \{0, 1, B\}^{(2 \cdot \lfloor \log l \rfloor + p + 5) \cdot l^2}$ of the strings $\text{bin}(i) \frown B \frown \text{bin}(j) \frown B \frown a_{ij} \frown B$, again sorted according to $(l-1) \cdot i + j$. (Here $\text{bin}(i)$ means the binary representation of i with exactly $\lfloor \log l \rfloor + 1$ bits.) In either case, $d = d(l, p)$ denotes the number of symbols occupied by one entry; thus, $d = p$ (without addresses) or $d = 2 \cdot \lfloor \log l \rfloor + p + 5$ (with addresses), respectively. In both cases, the number n of symbols used to represent a matrix A satisfies $l^2 d \leq n \leq l^2 d + p + 1$. The two representations induce two functions on strings that correspond to the operation of transposing a matrix: MATRIX TRANSPOSITION without addresses (resp. with addresses) maps A' to $(A^{\mathrm{T}})'$ (resp. A'' to $(A^{\mathrm{T}})''$) for all matrices A.

Theorem 2.1. *The time complexity of* MATRIX TRANSPOSITION (*with or without addresses*) *on Turing machines with one worktape and a two-way input tape is*

$$\Theta\left(n \cdot l \Big/ \left\lceil \left(\frac{\log l}{p}\right)^{1/2} \right\rceil\right),$$

where n is the length of the input, l the number of rows and columns, and p the length of the matrix entries in bits.

Corollary 2.2. (a) *The time complexity of transposing Boolean matrices* ($l \times l$-*matrices with entries in* $\{0, 1\}$) *on such TMs is* $\Theta((l^3/\log l)^{1/2}) = \Theta(n^{3/2}/(\log n)^{1/2})$ (*for the representation without addresses*).

(b) *The time complexity of transposing Boolean $l \times l$-matrices in the representation with addresses on such TMs is* $\Theta(n \cdot l/(\log l)^{1/2}) = \Theta(n^{3/2}/\log n)$.

Remark 2.3. The main differences between the proof of Theorem 2.1 and the proof of the corresponding theorem in the preliminary version [12, Theorem 2.1] are the following.

(a) The computation of the TM M that one considers in the lower bound argument (Theorem 2.4) is analyzed "backwards", separately for disjoint intervals on the worktape. In this way, the combinatorial lemma in [12], which regarded the overall structure of the head movements, becomes superfluous.

(b) In the Kolmogorov complexity argument, some new tricks (from [2]) are employed to save some bits in the short description of the incompressible string X. In particular, the use of "addresses" of substrings is avoided altogether; this makes it possible to deal with the case $p < \log l$.

(c) The upper bound was trivial in [12], since only the case $p \geqslant \log l$ was considered. The new upper bound proof for the case $p < \log l$ is given in Sections 3 and 4 (Theorem 4.1).

Theorem 2.4. *For every one-tape off-line Turing machine M that computes* MATRIX TRANSPOSITION (*with or without addresses*) *there is an l_M with the following property: for all $l \geqslant l_M$ and all p there is an input representing an $l \times l$-matrix A with p-bit entries for which M uses* $\geqslant (nl/\lceil(\log(l)/p)^{1/2}\rceil)/100$ *steps.*

Remark 2.5. Even if M computes MATRIX TRANSPOSITION only for specific values of l and p, the lower bound applies if $l \geqslant l_M$.

Proof of Theorem 2.4. Fix a one-tape off-line TM M that computes MATRIX TRANSPOSITION (with or without addresses). Fix a value of l that is sufficiently large. (How large l has to be depends on M; a bound l_M for l will arise from the calculations below.) Fix some p. Let $d := p$ (in the case without addresses) or $d := 2 \cdot \lfloor \log l \rfloor + p + 5$ (in the case with addresses). Let X be a binary string of length l^2p that is incompressible in the Kolmogorov sense, that is, $K(X) \geq |X| = l^2p$. Split X into l^2 segments $b_1, \ldots, b_{l^2}$ of length p each. We analyze the computation of M on the input I of length $n = l^2d$, where $I = A'$ (resp. $I = A''$) for the matrix $A = (a_{ij})_{1 \leq i,j \leq l}$ with $a_{ij} = b_{(i-1)l+j}$, for $1 \leq i, j \leq l$.

Let $\alpha := \lceil (\log(l)/p)^{1/2} \rceil$. Fix an arbitrary interval W of length $3\alpha \cdot ld$ on the worktape that (at the end of the computation) contains 3α columns of A. We will see that M spends $\geq l^2d/18 = n/18$ steps with its worktape head in W. (Since there are $\geq \lfloor l/3\alpha \rfloor$ many disjoint intervals of this type on the worktape, this will show that M makes $>(l^3d/\alpha)/100$ many steps altogether, which is what we want to prove.) Let L, V, R be the left, middle, and right third of W, respectively. Each of the intervals L, V, R consists of $\alpha \cdot ld$ tape cells.

Before setting out with the formal argument, let us sketch the intuition behind it. What are possible strategies the TM M might use to get the information contained in the $\alpha \cdot l$ entries of A that belong to the columns in V to their destination? First, there is the straightforward way (cf. Case 1 below): M copies these entries "directly" into V, that is, for all these entries a_{ij} there is some time during the computation at which the input head visits a_{ij} on the input tape and simultaneously the worktape head is in V, or at least close to V. Observe the way these entries a_{ij} are scattered over the input tape: in each row of A there are α entries that belong to V. There are two extreme possibilities how M can realize the necessary head movements: (a) The worktape head stays in (or close to) V while the input tape head performs one sweep across the input tape. (b) The worktape head enters and leaves the area around V about l times; each time the input tape head visits a different row of A on the input tape. (In both cases, M spends $\Omega(n)$ steps in W.) At the other extreme, it could be the case that M manages to carry the information about the entries in V into the destination area without ever realizing a "direct connection" between origin (on the input tape) and target area (cf. Case 2 below). That is, whenever the worktape head is in W, the input tape head is *not* reading any of the entries a_{ij} with destination V. In this case, we imagine that M has to pick up the required information while the worktape head is somewhere outside W, has to store it in its finite control and possibly in the position of its input tape head, and carry it across L (resp. R) into V. Only $O(\log n)$ bits of information can be carried into V at any one of these visits. Thus, M needs $\Omega(\alpha \cdot lp/\log n)$ crossings of L (resp. R) to get all the information needed into V, hence M spends $\Omega(\alpha^2 \cdot l^2pd/\log n)$ steps in W, which is $\Omega(n)$. Below, we employ a Kolmogorov complexity argument to make this idea precise.

Of course, M may use any mixture of the "pure" strategies just sketched. Thus, below we split into cases according to the strategy that seems to be predominant.

To start out with the formal proof, we define certain crossing sequences. For each cell boundary c at the right end of a cell in L we consider the "left-to-right crossing sequence", which records the state of M and the position of the input head for every time the worktape head crosses c from left to right. We choose a boundary c_L for which the number of such crossings is minimal among all cell boundaries in L. Let $\#C_L$ be the number of times c_L is crossed from left to right and let $|C_L|$ be the number of bits needed to code the crossing sequence in binary. Obviously, $|C_L| \leqslant \#C_L \cdot (a_M + \log n)$ for some constant a_M (which depends on M). Similarly, consider "right-to-left crossing sequences" in R, choose a boundary c_R in R that is crossed a minimal number of times, and define $\#C_R$ and $|C_R|$ accordingly. (As usual, the intended effect of choosing $\#C_L$ and $\#C_R$ minimal is that the worktape head spends $\geqslant \alpha \cdot ld \cdot (\#C_L + \#C_R)$ steps in $L \cup R \subseteq W$.) We consider several cases.

Case 1. *There are entries in $>l/3$ different rows of A that belong to columns in V and are visited by the input tape head at some time when the worktape head is in $[c_L, c_R]$.*

There are two subcases.

Case 1(a): *The worktape head enters $[c_L, c_R]$ more than $l/6$ times.* Then M spends $\geqslant (l/6) \cdot \alpha \cdot ld \geqslant l^2 d/6 = n/6$ steps with its worktape head in $L \cup R \subseteq W$, since c_L and c_R were chosen so as to minimize $\#C_L$ and $\#C_R$.

Case 1(b): *The worktape head does not enter $[c_L, c_R]$ more than $l/6$ times.* Then the input head inspects entries in the intersection of row A_i and the columns in V for several rows A_i during one visit of the worktape head to $[c_L, c_R]$. If during any one visit entries in k different rows are inspected on the input tape, this visit lasts at least $(k-1) \cdot ld - \alpha \cdot d$ steps (the time needed for the input tape head to travel the distance between these entries). Altogether $>l/3$ rows are inspected by the input head, and there are $\leqslant l/6$ visits of the worktape head in $[c_L, c_R]$, hence the worktape head spends $\geqslant (l/3) \cdot ld - (l/6) \cdot ld - (l/6) \cdot \alpha \cdot d \geqslant (l/10) \cdot ld = n/10$ steps in $[c_L, c_R] \subseteq W$, for l large enough.

Case 2: *There are $\leqslant l/3$ rows A_i of A so that the input tape head visits entries in A_i that belong to V while the worktape head is in $[c_L, c_R]$.*

We show by a Kolmogorov complexity argument that in this case the worktape head has to enter $[c_L, c_R]$ many times (intuitively, to "carry in" the information about the other entries of V). Suppose we are given:

(i) the program of M;

(ii) the length of $[c_L, c_R]$;

(iii) the left-to-right crossing sequence at c_L and the right-to-left crossing sequence at c_R (input head position and state of M for each crossing);

(iv) l, p, and α; the index of the first and last column in V; the position of V in $[c_L, c_R]$;

(v) the entries of A outside of V (ordered row by row, concatenated into a single string);

(vi) those entries in V that are visited on the input tape while the worktape head is in $[c_L, c_R]$ (these are arranged in the order these visits on the input tape take place, and concatenated into a single string);

(vii) the code $\lceil M' \rceil$ of a Turing machine M' that takes the string described in (i)-(vi) as input and works as follows. First, simulate the computation of M on input I during all time periods that the worktape head spends in $[c_L, c_R]$: Set up the input tape, inserting only the information given in (iv) (for the format) and (v), and leaving the entries that belong to V blank. Set up a blank worktape of the size given in (ii). Start simulating M at the first time step at which the worktape head enters $[c_L, c_R]$ (the necessary information is provided by (iii)). Each time the worktape head leaves $[c_L, c_R]$, interrupt the simulation and resume it at the step at which the worktape head re-enters $[c_L, c_R]$ (using (iii)). Whenever the input tape head visits some entry that has been blank since the beginning (one of the entries in the columns that belong to V), interrupt the simulation, and copy the next p bits from the string given in (vi) to the blank "slot" of length p corresponding to this entry on the input tape, and then resume the simulation. The simulation is finished when the worktape head leaves $[c_L, c_R]$ for the last time, or when M halts. After this happens, fill in the entries still missing on the input tape, using the entries in V that have been built up on the worktape during the simulation (the position of V within $[c_L, c_R]$ is given in (iv)). Output the "information parts" in I (without addresses), concatenated into a single string.

It is clear that the procedure described in (vii) outputs X, the incompressible string from which the input I was constructed. So the number of bits needed to code the information described in (i)-(vii) is an upper bound for $K(X)$, by the definition of Kolmogorov complexity. For the number of bits needed for different parts of the string, we get the following estimates:

(i) a_M bits, for some constant a_M;
(ii) $\leq 2 \cdot \log n$ bits;
(iii) $\leq (\# C_L + \# C_R) \cdot (a_M + \log n)$ bits;
(iv) $\leq 6 \cdot \log n$ bits;
(v) $lp \cdot (l - \alpha)$ bits;
(vi) $\leq \alpha \cdot lp/3$ bits;
(vii) $a_{M'}$ bits, for some constant $a_{M'}$.

Thus,

$$l^2 p \leq K(X) \leq \mathrm{O}(\log n) + (\# C_L + \# C_R) \cdot (a_M + \log n) + l^2 p - \alpha \cdot lp + \alpha \cdot lp/3,$$

or, slightly rewritten,

$$\tfrac{2}{3}\alpha \cdot lp/\log n \leq a'_M + (\# C_L + \# C_R) \cdot \left(1 + \frac{a_M}{\log n}\right),$$

for some constant a'_M. For l large enough, this entails

$$\tfrac{1}{2}\alpha \cdot lp/\log n \leq \# C_L + \# C_R.$$

Since we have chosen $\# C_L$ and $\# C_R$ minimal in L and R, and L and R have length

$\alpha \cdot ld$, we may conclude that M spends $\geq(\frac{1}{2}\alpha \cdot lp/\log n) \cdot \alpha \cdot ld = \frac{1}{2}\alpha^2 \cdot l^2pd/\log n$ steps with its worktape head in W. There are two cases.

Case (i): $p \geq \frac{1}{4}\log l$. Then it is easy to see that $p/\log n \geq \frac{1}{9}$ for l large enough. Further, $\alpha \geq 1$ by definition. Thus the worktape head spends $\geq \frac{1}{18}l^2d = n/18$ steps in W.

Case (ii): $p < \frac{1}{4}\log l$. Then the worktape head spends

$$\geq \tfrac{1}{2}(\log(l)/p) \cdot l^2pd/\log n = \tfrac{1}{2}l^2d \cdot \log(l)/\log n \geq n/12$$

steps in W.

This finishes Case 2 and the proof of Theorem 2.4. □

3. The speed of copying on one-tape off-line Turing machines

In this section we describe a trick that enables one-tape off-line Turing machines to copy strings from one place on the worktape to another a little faster than one would expect. This observation will be used in the upper bound proof of the next section.

The following subtask occurs frequently in programs for TMs: Copy a string (of s binary digits, say) from one place on a worktape to another place on the same worktape (which is q cells away). Clearly, executing this task takes $O(s+q)$ steps on TMs with at least two worktapes and $\Omega(s \cdot q)$ steps on TMs with one worktape and no input tape (see [5]). Under certain circumstances a one-tape off-line TM can perform this task faster than in $O(s \cdot q)$ steps.

Proposition 3.1. *A one-tape off-line Turing machine can copy a binary string of length $s = \frac{1}{2}\log n$ across q cells in $O(n^{1/2}+q)$ steps, provided that it starts from a configuration with the input head at the left end of the input tape and the worktape head within the string to be copied. (As usual, n is the length of the input.)*

Proof. We assume that the worktape has sufficiently many tracks to accommodate counters and markers needed in the following. Regard the string to be copied as a binary number $k \in \{0, 1, \ldots, n^{1/2}-1\}$. Store k *in unary* in the distance of the input head from the left end of the input tape. This requires counting from 0 to k on a binary counter (placed in a track parallel to the string to be copied), which takes $O(k) = O(n^{1/2})$ steps. Then move the worktape head to the area where the string is to be copied; this takes $O(q)$ steps. Now translate k back into binary notation, by moving the input tape head back to the left end of the tape and simultaneously counting the number of the moves of the input tape head in a binary counter placed in the cells to which the string is to be copied. This again takes $O(k)$ steps. □

Remark 3.2. (a) The proposition entails that one-tape off-line TMs can copy strings of length s across q cells in $O(q+(s/\log n) \cdot n^{1/2})$ steps. Thus, this type of TM occupies an intermediate position between one-tape TMs with one-way input tape or without input tape ($O(q \cdot s)$ steps) and two-tape TMs ($O(q+s)$ steps). (The observation made in Proposition 3.1 is discussed in more detail in [1].)

(b) Up to now, only the negative side of this feature of one-tape off-line TMs has been noticed: in the context of lower bound proofs via crossing sequence arguments (e.g., the proof in Section 2) one always had to include the position of the input head in the crossing sequence, thus weakening the resulting lower bound.

In the context of Section 4, where this method is to be applied, we need a slight generalization of Proposition 3.1 to the case where the bits to be copied are not stored in a contiguous interval but rather are spread out over the worktape, both the original bits and the copies. More precisely, we consider the following task.

Task. Assume $p \leq \frac{1}{8}\log n$. Given are $r := (1/2p) \cdot \log n$ equidistant intervals $I_1, \ldots, I_r$ of p cells each and equidistant intervals $J_1, \ldots, J_r$ of p cells each on the worktape. The distance between the leftmost cells of I_ρ and $I_{\rho+1}$ (J_ρ and $J_{\rho+1}$), $1 \leq \rho < r$, is called s_1 (s_2). Each interval I_ρ contains a binary string of p bits. The content of I_ρ is to be copied to J_ρ, $1 \leq \rho \leq r$. The distance between I_1 and J_1 is q.

Corollary 3.3. *A one-tape off-line TM can perform this task in $O(q+(s_1+s_2) \cdot n^{1/2}/p)$ steps, provided that it starts from a configuration with the input head at the left end of the input tape and the worktape head in one of the I_ρ.*

Proof (This is a slight variation of the proof of Proposition 3.1). The contents of $I_1, \ldots, I_r$ are concatenated and considered as the representation of a binary number $k \in \{0, 1, \ldots, n^{1/2}-1\}$. The counter in which we count up to k (while the input tape head is moved to the right) is located in $I_1, \ldots, I_r$, that is, it has r blocks of p bits each, s_1 cells apart. It is easy to see that counting up to k takes $O(k \cdot s_1/p) = O(n^{1/2} \cdot s_1/p)$ steps. (A counter consisting of $\frac{1}{2}\log n$ single bits spaced s_1/p cells apart needs at least as many steps as this "blocked" counter.) Analogously, the decoding phase takes $O(n^{1/2} \cdot s_2/p)$ steps. □

4. The upper bound for matrix transposition

In this section we prove the upper bound claimed in Theorem 2.1.

Theorem 4.1. *There is a one-tape off-line Turing machine that computes* MATRIX TRANSPOSITION (*with or without addresses*) *and needs*

$$O\left(n \cdot l \Big/ \left\lceil \left(\frac{\log l}{p}\right)^{1/2} \right\rceil\right)$$

steps on inputs of length n that represent a matrix with l rows and columns and p-bit entries.

Proof. We describe a TM M of this type that computes MATRIX TRANSPOSITION, for the case where matrices are represented with addresses. (Thus, $d = 2 \cdot \lfloor \log l \rfloor + p + 5$ in the following. Since the addresses are not used at all during the computation, the case "without addresses" can be treated in exactly the same way.) As in Section 2, we let $\alpha := \lceil (\log(l)/p)^{1/2} \rceil$. The machine M begins the computation by copying the address parts of the input string to the worktape, but replacing each entry a_{ij} by p blanks. (From here on, the address parts are never changed.) This leaves l^2 "slots" or "positions" of length p bits each. For $1 \leqslant i, j \leqslant l$, the entry a_{ij}, that is, the $((i-1) \cdot l + j)$th entry on the input tape, has to be inserted in the $((j-1) \cdot l + i)$th position on the worktape.

In a further preliminary phase, M constructs strings of length l^2, $\lfloor \log l \rfloor$, l, d, α, ld, $\alpha \cdot d$ (and some other distances that are needed in the following) on different tracks of the worktape. Also, M marks off blocks of length d (=entries), $\alpha \cdot d$, ld (=columns), $\lfloor \log(l)/p \rfloor \cdot ld$, etc. on the worktape. All this can easily be done in $O(n)$ steps. (Recall that in $O(t)$ steps M can count up to t in a binary counter consisting of $\lfloor \log t \rfloor + 1$ consecutive bits, and notice that the input tape can be used as a unary counter, thus as a "yardstick" for marking off blocks on the worktape.)

The machine M acts differently in the cases $\alpha \leqslant 2$ and $\alpha > 2$. First, we dispose of the simple case $\alpha \leqslant 2$. Here M directly copies the entry a_{ij} of A from the input tape to the corresponding position $(j-1) \cdot l + i$ on the worktape for $1 \leqslant i, j \leqslant l$. This is done row by row, processing the input from left to right, and requires l sweeps over the worktape. (To orient itself, M uses the markers in distance ld on the worktape, and some other markers as needed.) The number of steps required in this case is $O(n \cdot l) = O(n \cdot l/\alpha)$ (since $\alpha \leqslant 2$).

Now we turn to the more interesting case $\alpha > 2$. Here, M can transpose the matrix A a little faster than the straightforward algorithm, by using the "fast copying method" described in Section 3. We will show that A can be transposed in $O(n \cdot l/\alpha)$ steps.

The computation proceeds in two phases. We think of A as partitioned into $\lceil l/\alpha \rceil$ submatrices B_r, $1 \leqslant r \leqslant \lceil l/\alpha \rceil$, consisting of α consecutive columns of A. (The last submatrix may have fewer than α columns; for simplicity, we disregard this special case.) We regard the matrix B_r as the rth column of an $l \times \lceil l/\alpha \rceil$-matrix $\tilde{A}$ (an entry of $\tilde{A}$ is a block of α entries of A, that is, a row of some B_r). The ordinary matrix transposition algorithm is applied to $\tilde{A}$; that is, the α consecutive entries of A that belong to a block are copied from the input tape to α consecutive "slots" on the worktape. The rows of A are treated one after another, that is, the input tape is read once from left to right. (Here M uses the markers in distance $\alpha \cdot d$ to orient itself.) This results in each submatrix B_r being stored on the worktape in row major order, in positions $(r-1) \cdot l \cdot \alpha + 1$ through $r \cdot l \cdot \alpha$; this is exactly the interval where the entries of B_r eventually have to be written. We may assume that at the end of this phase the worktape head and the input tape head are at the left end of the respective tapes. The time required for the first phase is the time needed for $\lceil l/\alpha \rceil$ sweeps over the worktape, that is, $O(n \cdot l/\alpha)$ steps.

In the second phase the entries of B_r are rearranged so as to appear in column major order, for each r separately. The *contents* of the input tape are not used anymore in this phase (but the input tape is used as a unary counter). Let r be fixed. We employ the "fast copying method" of Section 3, in the version described in Corollary 3.3. Consider a fixed column of B_r, and split this column into $\lceil l/\lfloor \log(l)/p \rfloor \rceil$ blocks of $\lfloor \log(l)/p \rfloor$ consecutive entries each. Fix such a block D. We explain how to get the entries of D to their correct positions. Before the beginning of the second phase, the entries in D are stored in $\lfloor \log(l)/p \rfloor$ intervals of length p each that are spaced $\alpha \cdot d$ cells apart, since B_r is stored on the worktape row by row. They are to be stored in $\lfloor \log(l)/p \rfloor$ intervals of length p that are spaced d cells apart (consecutive positions). Both the source and target areas are contained in the area occupied by B_r, hence are $\leq \alpha \cdot ld$ cells apart. By Corollary 3.3, the copying process takes $O(\alpha \cdot ld + l \cdot (\alpha \cdot d + d)/p) = O(\alpha \cdot ld)$ steps. At the end the input tape head is again at the left end of the input tape and the worktape head is at the left end of the area for B_r. The total number of blocks D into which the columns of B_r are split is

$$\alpha \cdot \left\lceil l \Big/ \left\lfloor \frac{\log l}{p} \right\rfloor \right\rceil = O(\alpha \cdot (l/\alpha^2)) = O(l/\alpha).$$

Thus, getting the entries in B_r to their correct positions takes $O(l/\alpha) \cdot O(\alpha \cdot ld) = O(l^2 \cdot d) = O(n)$ steps.

Since there are $\lceil l/\alpha \rceil$ submatrices B_r, the second phase altogether takes $O(n \cdot l/\alpha)$ steps, just as the first phase. This finishes the proof of Theorem 4.1. □

5. Implications for lower bounds on sorting and the two-tapes-versus-one problem

If the representation of matrices with addresses is used (cf. Section 2), MATRIX TRANSPOSITION immediately reduces to sorting. Thus, the lower bound of Section 2 implies a lower bound for sorting on one-tape off-line TMs. Since a TM with two worktapes can transpose binary $l \times l$-matrices in $O(n \log n)$ steps (here, $n = l^2$ is the length of the input), we get lower bounds for the simulation of multitape TMs by one-tape off-line TMs. (Here "simulation" just means that the same function is computed.) These observations are discussed in detail in this section.

We consider the following version of the sorting problem:
The input is $x_1 {}^\frown B {}^\frown x_2 {}^\frown B {}^\frown \dots {}^\frown B {}^\frown x_r$, where $x_1, \dots, x_r \in \{0, 1\}^*$ and the output is $x_{\pi(1)} {}^\frown B {}^\frown x_{\pi(2)} {}^\frown B {}^\frown \dots {}^\frown B {}^\frown x_{\pi(r)}$, where π is a permutation of $\{1, \dots, r\}$ that satisfies $x_{\pi(1)} \leq \cdots \leq x_{\pi(r)}$ in the lexicographic sense. As usual, n denotes the length of the input.

Theorem 5.1. *Sorting requires $\Omega(n^{3/2}/(\log n)^{1/2})$ steps on one-tape off-line Turing machines.*

Proof. We restrict our attention to inputs of the form $x_1{}^\frown B^\frown \ldots {}^\frown B^\frown x_r$, where $r = l^2$ and $x_q = \text{bin}(j)^\frown 0^\frown \text{bin}(i)^\frown 0^\frown a_{ij}$ for $q = (i-1) \cdot l + j$ and $a_{ij} \in \{0, 1\}^{\lfloor \log l \rfloor}$, $1 \leq i, j \leq l$. Sorting such an input means nothing but moving x_q to position $\tau(q) = (j-1) \cdot l + i$, for $q = (i-1) \cdot l + j$, which is exactly the same permutation as in the matrix transposition case. The proof of Theorem 2.4 depends only on the way the substrings are permuted, and hence the result applies here, too: for l large enough, a one-tape off-line TM needs $\geq n \cdot l/100 = \Omega(n^{3/2}/(\log n)^{1/2})$ steps to sort $r = l^2$ bitstrings of length $3 \cdot \lfloor \log l \rfloor + 4$. □

Theorem 5.2. *There are functions that can be computed in linear time on a Turing machine with two tapes (actually, one tape and one pushdown store are enough), but require time $\Omega(n^{3/2}/\log n)$ on a one-tape off-line Turing machine.*

Proof. We take as an example the function MATRIX TRANSPOSITION (with addresses) restricted to binary matrices. That is, the inputs have the form A'', where $A = (a_{ij})_{1 \leq i,j \leq l} \in \{0, 1\}^{l \times l}$. (See Section 2 for the definition of A''.) The length of the input A'' is $n = l^2 \cdot (2 \cdot \log l + 6)$, and from Theorem 2.4 we know that computing MATRIX TRANSPOSITION for such inputs on one-tape off-line TMs takes $\geq n \cdot l/100 \cdot (\log l)^{1/2} = \Omega(n^{3/2}/\log n)$ steps.

On the other hand, the same function can be computed on a two-tape TM in linear time, as follows. First, it is checked (in $O(n)$ steps) if the input is of the correct form A'' for a binary matrix $A = (a_{ij})_{1 \leq i,j \leq l}$, for some l. Then the "compressed" version A' of A (elements listed row by row, no addresses) is generated on the worktape; this is a string of length l^2. By Lemma 5.3 (see below), the representation $(A^{\mathrm{T}})'$ of the transpose A^{T} can be constructed in $O(l^2 \cdot \log(l^2)) = O(n)$ steps, on one tape and one pushdown store. The string $(A^{\mathrm{T}})'$ can then be expanded to the representation $(A^{\mathrm{T}})''$ with addresses in $O(n)$ steps; for example, one may copy the bits of A^{T} back into the "information positions" of the input string. □

For completeness, we briefly sketch the (classical) algorithm for transposing quadratic matrices on two-tape TMs, which was presented (in German) in [14]. (The elements are permuted in $\log l$ phases according to the butterfly graph.)

Lemma 5.3. MATRIX TRANSPOSITION *for quadratic Boolean matrices (represented without addresses) can be computed in $O(n \log n)$ steps on a Turing machine with two worktapes. (In fact, one worktape and one pushdown store are enough.)*

Proof. We describe a TM with one worktape and one pushdown store that on input

$$a_{00}a_{01} \ldots a_{0,l-1} \ldots a_{l-1,0}a_{l-1,1} \ldots a_{l-1,l-1} \in \{0, 1\}^{l^2}$$

(given on the worktape) produces the output

$$b_{00}b_{01}\ldots b_{0,l-1}\ldots b_{l-1,0}b_{l-1,1}\ldots b_{l-1,l-1},$$

where $b_{ij} = a_{ji}$ for $0 \leq i, j < l$, in $O(l^2 \cdot \log l)$ steps. The algorithm works in $r = \log l$ phases. (Assume that r is an integer.) Denote the content of the worktape at the end of phase s by

$$a^s_{00}a^s_{01}\ldots a^s_{0,l-1}\ldots a^s_{l-1,0}a^s_{l-1,1}\ldots a^s_{l-1,l-1}.$$

($a^0_{ij} = a_{ij}$; we want $a^r_{ij} = b_{ij} = a_{ji}$.) For $0 \leq k < l$, denote by $k(s)$ the sth least significant bit of $\text{bin}(k)$, the binary representation of k, and by $k[s]$ the number obtained by changing exactly this sth bit in $\text{bin}(k)$. We want the action of the TM in phase s to result in the following:

$$a^s_{ij} = \begin{cases} a^{s-1}_{i[s]j[s]} & \text{if } i(s) \neq j(s), \\ a^{s-1}_{ij} & \text{if } i(s) = j(s). \end{cases}$$

For this, all elements a^{s-1}_{ij} with $i(s) = 0, j(s) = 1$ have to be moved $(i[s] \cdot l + j[s]) - (i \cdot l + j) = 2^{s-1} \cdot (l-1)$ cells to the right, and the elements a^{s-1}_{ij} with $i(s) = 1, j(s) = 0$ have to be moved $2^{s-1} \cdot (l-1)$ cells to the left. This can be done in $O(l^2)$ steps, for instance, by storing two copies of the worktape in the pushdown store and using the top copy to insert the elements that have to be moved to the right on the worktape, and the bottom copy for the elements that are moved to the left. (The bookkeeping details are easily filled in.) To see that $a^r_{ij} = a_{ji}$, note (inductively) that $a^s_{ij} = a_{g(s,i,j)g(s,j,i)}$, where $g(s, k, m)$ denotes the number with binary representation $k(r)k(r-1)\ldots k(s+1)m(s)\ldots m(1)$. □

6. Separation of one-tape off-line Turing machines with and without output tape

In this section we consider one-tape off-line Turing machines with an additional write-only output tape. We show that even if the head on the extra tape can move only in one direction (that is, the output tape is "one-way"), this type of TM is more powerful than the type considered so far in this paper.

Theorem 6.1. *A one-tape off-line Turing machine with an additional one-way write-only output tape can compute* MATRIX TRANSPOSITION *for Boolean* $l \times l$*-matrices in* $O(l^{5/2}) = O(n^{5/4})$ *steps.* (*n is the bitlength of the input*).

Remark 6.2. (a) Recall that in Theorem 2.4 a lower bound of $\Omega(n^{3/2}/(\log n)^{1/2})$ was shown for this problem on one-tape off-line TMs without output tape.

(b) The upper bound is optimal: in [2] it is shown that TMs of the type considered in Theorem 6.1 need $\Omega(n^{5/4})$ steps to compute MATRIX TRANSPOSITION.

Proof of Theorem 6.1. For simplicity we assume that $l^{1/2}$ is an integer. The permutation τ of $\{1,\ldots,l^2\}$ that maps $(i-1)\cdot l+j$ to $(j-1)\cdot l+i$ can be written as $\tau=\rho\circ\rho$, where ρ is the permutation of $\{1,\ldots,l^2\}$ defined by

$$\begin{aligned}&(i_1-1)\cdot l^{3/2}+(i_2-1)\cdot l+(j_1-1)\cdot l^{1/2}+j_2\\ &\quad\mapsto(j_2-1)\cdot l^{3/2}+(i_1-1)\cdot l+(i_2-1)\cdot l^{1/2}+j_1,\end{aligned}$$

for $i_1, i_2, j_1, j_2\in\{1,\ldots,l^{1/2}\}$.

On a two-tape TM this permutation ρ can easily be realized: the head on the first tape sweeps across its tape $l^{1/2}$ times while the head on the second tape simultaneously performs one (slow) sweep. In this way, a one-tape off-line TM with an additional one-way write-only output tape can compute MATRIX TRANSPOSITION in $O(l^{5/2})=O(n^{5/4})$ steps as follows: it applies the permutation ρ to the l^2 matrix elements given on the input tape and writes this intermediate result to the worktape (this takes $O(n\cdot l^{1/2})=O(n^{5/4})$ steps, by the observation above). Subsequently the TM applies the permutation ρ to the l^2 matrix elements on the worktape and writes the result to the output tape (again, this takes $O(n^{5/4})$ steps). Clearly, the output tape head moves only from left to right. The details of the algorithm (computation of l, $l^{1/2}$; synchronization of the sweeps; modifications in the case where l is not a square) are easily filled in. $\square$

7. No permutation requires substantially more sweeps than matrix transposition

The proof of Theorem 2.4 rests on the fact that matrix transposition corresponds to a permutation that "scatters" adjacent elements in its domain. This gives rise to the question whether there are other permutations π that are even more difficult to realize by the head movement of Turing machines of the type considered here (and thus might yield even larger lower bounds for such machines). We show in Proposition 7.1 that a well-known result by Erdös and Szekeres implies a negative answer: *every* permutation π of m numbers can be realized by $O(m^{1/2})$ simultaneous sweeps over the input tape and the output area on the worktape of a one-tape off-line TM. Thus, provided that the machine already "knows" where each element has to be placed, a one-tape off-line TM can realize an arbitrary permutation π of m elements of bitlength p each in time $O(m^{3/2}\cdot p)=O(n\cdot m^{1/2})$, where $n=m\cdot p$ is the length of the input.

Proposition 7.1. *Any sequence* $(x_1,\ldots,x_m)$ *of* m *pairwise different numbers (in arbitrary order) can be written as a disjoint union of* $O(m^{1/2})$ *monotone (increasing or decreasing) subsequences.*

Proof. According to Erdös and Szekeres [4], every sequence of length i^2+1 contains a monotone subsequence of length $i+1$. Thus every sequence of length j contains a monotone subsequence of length $\lceil j^{1/2}\rceil$. We use this to show that every sequence of length m consists of $\leq 2 \cdot \lceil m^{1/2} \rceil$ monotone subsequences. We proceed by induction on $\lceil m^{1/2} \rceil$, the cases $m = 1, 2, 3$ being trivial. The result of Erdös and Szekeres is applied twice to the given sequence of length $m \geq 4$. The first time we remove a monotone subsequence of length $\lceil m^{1/2} \rceil$ and the second time a monotone subsequence of length $\lceil (m - \lceil m^{1/2} \rceil)^{1/2} \rceil$. Then we have for the length k of the remaining sequence

$$k = (m - \lceil m^{1/2} \rceil) - \lceil (m - \lceil m^{1/2} \rceil)^{1/2} \rceil \leq m - 2 \cdot m^{1/2} + 1$$

for all m, as a simple calculation shows. Thus

$$\lceil k^{1/2} \rceil \leq \lceil m^{1/2} - 1 \rceil = \lceil m^{1/2} \rceil - 1;$$

hence the induction hypothesis can be applied to the remaining sequence of length k. □

References

[1] M. Dietzfelbinger, The speed of copying on one-tape off-line Turing machines, *Inf. Process. Lett.* **33** (1989/90) 83-89.

[2] M. Dietzfelbinger and W. Maass, The complexity of matrix transposition on one-tape off-line Turing machines with output tape, in: T. Lepistö and A. Salomaa, eds., *Proc. 15th ICALP*, Lecture Notes in Computer Science **317** (Springer, Berlin, 1988) 188-200.

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

[4] P. Erdös and G. Szekeres, A combinatorial problem in geometry, *Compositio Math.* **2** (1935) 463-470.

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

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

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

[8] M. Li, L. Longpré and P.M.B. Vitányi, On the power of the queue, in: A.L. Selman, ed., *Structure in Complexity Theory, Proceedings*, Lecture Notes in Computer Science **223** (Springer, Berlin, 1986) 219-233.

[9] W. Maass, Are recursion theoretic arguments useful in complexity theory? in: *Proc. 7th Internat. Conf. Logic, Methodology, and Philosophy of Science*, Salzburg (1983) (North-Holland, Amsterdam, 1985).

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

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

[12] W. Maass and G. Schnitger, An optimal lower bound for Turing machines with one work tape and a two-way input tape, in: A.L. Selman, ed., *Structure in Complexity Theory, Proceedings*, Lecture Notes in Computer Science **223** (Springer, Berlin, 1986) 249-264.

[13] W. Maass, G. Schnitger and E. Szemerédi, Two tapes are better than one for off-line Turing machines, in: *Proc. 19th ACM STOC* (1987) 94-100.

[14] W.J. Paul, Optimales Transponieren quadratischer Matrizen, in: *Gesellschaft für Informatik, 3. Jahrestagung, Hamburg 1973*, Lecture Notes in Computer Science **1** (Springer, Berlin, 1973) 72-80.

[15] W.J. Paul, Kolmogorov complexity and lower bounds, in: L. Budach, ed., *Proc. 2nd Internat. Conf. Fund. Computation Theory* (Akademie-Verlag, Berlin, 1979) 325-334.

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

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

[18] H.J. Stoss, Rangierkomplexität von Permutationen, *Acta Inform.* **2** (1973) 80-96.