Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

Ming Li, P.M.B. Vitányi

# Tape versus queue and stacks: The lower bounds

Department of Computer Science/Algorithmics & Architecture Report CS-R8519 September

# Tape versus Queue and Stacks: The Lower Bounds

Ming Li*

*Department of Computer Science*
*Cornell University, Ithaca, NY 14853, USA*

Paul M.B. Vitányi

*Centre for Mathematics & Computer Science (CWI)*
*Kruislaan 413, 1098 SJ Amsterdam, The Netherlands*

Several optimal or nearly optimal lower bounds are derived on the time needed to simulate queue, stacks (stack = pushdown store) and tapes by one off-line single-head tape-unit with one-way input, both deterministic and nondeterministic. The techniques rely on algorithmic information theory (Kolmogorov complexity).

*1980 Mathematics Subject Classification* : 68C40, 68C25, 68C05, 94B60, 10-00
*CR Categories* : F.1.1, F.1.3, F.2.3 69 F11, 69F13, 69F 22
*Keywords & Phrases* : multitape Turing machine, stack, queue, pushdown stores, determinism, nondeterminism, on-line, off-line, time complexity, lower bound, simulation by one tape, algorithmic information theory, Kolmogorov complexity
*Note* : This report supersedes M. Li, On 1 tape versus 2 stacks, Report TR-84-591, Department of Computer Science, Cornell University, January 1984; M. Li, Lower bounds by Kolmogorov complexity, pp. 382-393 in: *Proceedings 12th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **194**, 1985, §2; and P.M.B. Vitányi, One queue or two pushdown stores take square time on a one-head tape unit, Report CS-R8406, Centre for Mathematics and Computer Science, Amsterdam, March 1984. It has been submitted for publication elsewhere.

## 1. Introduction

We derive the following *lower* time bounds for simulation by *off-line* machines with *one-way input*. Deterministic case:

* Work supported in part by an NSF grant DCR-8301766. Now at the Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210.

Report CS-R8519
Centre for Mathematics and Computer Science
P.O. Box 4079
1009 AB Amsterdam, The Netherlands

§2.1. Simulation of 2 stacks (stack = pushdown store) by 1 tape requires $\Omega(n^2)$ time. This is optimal.

§2.2. Simulation of 1 queue by 1 tape requires $\Omega(n^2)$ time. This is optimal.

Nondeterministic case:

§3.1. Simulating 2 stacks by 1 tape requires $\Omega(n^{1.5} / \sqrt{\log n})$ time. This bound nearly matches the $O(n^{1.5} \sqrt{\log n})$ upper bound in [9].

§3.2. Simulating 1 queue by 1 tape requires $\Omega(n^{4/3} / \log^{2/3} n)$ time. This bound leaves a gap with the $O(n^{1.5} \sqrt{\log n})$ upper bound in [9].

§3.3. Simulating 2 tapes by 1 tape requires $\Omega(n^2 / (\log n \ \log\log n))$ time. This is a multiplicative factor $\log n$ improvement of the $\Omega(n^2 / (\log^2 n \ \log\log n))$ lower bound in [10].

§3.4 For a precisely defined notion of *on-line nondeterministic* simulation, it takes $\Omega(n^2)$ time to simulate 1 queue or 2 stacks *on-line* by a nondeterministic one-head tape unit. This is optimal.

## *1.1. Historical Background*

It has been known for over twenty years that all multitape Turing machines can be simulated on-line by 2-tape Turing machines in time $n \log n$ [6], and by 1-tape Turing machines in time $n^2$ [7]. In [13] two single-head tapes were shown to be more powerful in real-time than one single-head tape. This result was generalized in [1] to $(k+1)$ tapes versus $k$ tapes. In [11] the proof was reduced to its essentials by introducing Kolmogorov complexity. The time penalty for the reduction of the number of tapes was only known to be at least linear, until the proof of a $\Omega(n \log^{1/(k+1)} n)$ lower bound for on-line simulation of $(k+1)$ tapes by $k$ tapes [12]. Thus, the simulation by 2 tapes was shown to be nearly optimal; for simulation by 1 tape the gap between the known lower bound and upper bound on the simulation time had hardly decreased. Unknown to each other, around 1983/1984 Wolfgang Maass at UC Berkeley, the present first author at Cornell and the present second author at CWI Amsterdam obtained a square lower bound on the time to simulate two tapes by one tape. The approaches used to obtain the results were pairwise different, as were the actual results themselves. All of them rely, however, on a remarkable notion introduced by Kolmogorov [8], Chaitin [2] and others in the 1960s and 1970s. The *Kolmogorov* or *algorithmic* complexity of a string is the length of the *shortest* binary string which describes it. Some strings cannot be described by shorter strings; they are random in the strongest possible sense and cannot be compressed. Besides being useful in logics and recursive function theory [2], this *algorithmic information theory* emerges as a powerful tool for various areas of Computing.

For the particular problem at issue, the first advance was reported at ICALP82 [14], a $\Omega(n^{1.5})$ lower bound on the time to simulate a pushdown store on-line by one *oblivious* tape unit. (Recall, that in an *oblivious* Turing machine the movement of the storage tape heads is independent of the input, and is a function of time alone.) In [15] this lower bound was improved to $\Omega(n^2)$, while [16] demonstrated a lower bound of $n^{1.618}$ on the time to simulate one queue or two pushdown stores by one (nonoblivious) tape unit. In [10] a language is exhibited which can be accepted by two deterministic one-head tape units in real-time, but an *off-line one-way input* one-head tape unit takes $\Omega(n^2)$ time in the deterministic case and $\Omega(n^2 / \log^{\alpha} n)$ time, for some small $\alpha$. in the nondeterministic case.

In §2 we report optimal square lower time bounds for simulation by one off-line deterministic tape with one-way input. (The machines have to produce an output only after having read all of the input.) We use one method to obtain the lower bound on the simulation time for two pushdown stores (improving [10] which shows the result for two tapes) and another 'adversary' argument for a *single* queue (with the result in §3.2 the only nontrivial such result on queues). For a natural concept of 'on-line' nondeterministic computation the latter adversary method straightforwardly generalizes to yield square time lower bounds for the simulation of one queue or two pushdown stores by one *on-line nondeterministic* one-head tape unit §3.4. In §3.1-§3.3 lower time bounds are obtained for simulation by one off-line nondeterministic tape with one-way input. Simulating two pushdown stores requires $\Omega(n^{1.5} / \sqrt{\log n})$ time and simulating one queue requires $\Omega(n^{4/3} / \log^{2/3} n)$ time. These lower bounds nearly match, respectively leave a gap with, the corresponding upper bounds of $O(n^{1.5} \sqrt{\log n})$ in [9]. Simulating two tapes requires $\Omega(n^2 / (\log n \log\log n))$ time, which is a multiplicative factor $\log n$ improvement of [10].

## *1.2. Storage, Computation Mode and Simulation*

The machines we consider have *storage* consisting of either linear lists with sequential access, i.e., *single-head tape units*, or last-in-first-out storage, i.e., *pushdown stores*, or first-in-first-out storage, i.e., *queues* [7]. *Stack* is used as a synonym for *pushdown store*. A *single-head tape-unit* is a 1-(storage)tape Turing machine. Apart from the *storage* handling of a machine, the computation model specifies the way the *input* is accessed, and the *output* is delivered. The basic distinction here is between on-line and off-line computation, more or less corresponding to interactive computer use and batch processing [7]. In an *on-line* computation the machine has to produce a (yes-no) output in between each pair of polled input commands. We are interested in *off-line* computation with *one-way input* (no back-up on the input) [7]. In an off-line one-way input computation the machine only has to produce a (yes-no) answer at the end of the input (which is marked). The same device appears to be more powerful in off-line one-way input mode than in on-line mode. In an off-line one-way input computation the time saved by not having to compute a yes-no answer for each input prefix can be expended in one splurge at the end-of-input. Let $n$ and the expressions below have the usual meaning.

*Observation.* Let $M$ be a multitape Turing machine. Let $\alpha > 1$ be a constant. Each $O(n^{\alpha})$ time on-line computation of machine $M$ can be performed by an $O(n^{\alpha})$ time off-line one-way input computation of machine $M$. Each $O(n^{\alpha})$ time off-line one-way input computation of machine $M$ can be performed by a $O(n^{\alpha+1})$ time on-line computation of machine $M$.

Because off-line computation with one-way input has to obey less restrictions than on-line computation (but is more restricted than off-line computation with two-way input) it is also called *weak on-line* [9]. For off-line one-way computation, the input string is inscribed on a separate input tape, one symbol in each square. The input is terminated by a distinguished end-of-input marker. When the machine polls for input, a read-only head on the input tape reads the symbol under scan and then moves to the right adjacent symbol. The machine does not write any output until it polls the end-of-input marker. Then it writes a 0 or 1 indicating *rejection* or *acceptance*. A *deterministic* machine *accepts* in time $T(n)$ if, for all accepted input strings of length $n$, the computation accepts within $T(n)$ steps. 'Off-line one-way input' without end-of-input marker reduces to 'on-line' for deterministic machines. For a nondeterministic machine the presence or absence of an end-of-input marker makes no difference since it can 'guess'. The difference between 'off-line one-way input' and 'on-line' in the nondeterministic case is whether or not the input sequence is the same for all *legal* computation paths, see §3.4. A *nondeterministic* machine *accepts* an input string if there is a legal computation path for that input ending in an acceptance. It *accepts* in time $T(n)$ if, for all accepted input strings of length $n$, there is a legal computation path of at most $T(n)$ steps ending in an acceptance.

An *off-line* machine $A$ with *one-way input* simulates a machine $B$ if, when started on the same input string, $A$ accepts if and only if $B$ accepts. An off-line machine $A$ with one-way input simulates machine $B$ in time $T(n)$:

- *deterministically* if $A$ and $B$ are both deterministic and accept in $T_A(n)$ and $T_B(n)$, respectively, $A$ simulates $B$ and $T_A(n) \leqslant T(T_B(n))$.
- *nondeterministically* if $A$ is nondeterministic and $A$, $B$ accept in $T_A(n)$ and $T_B(n)$, respectively, $A$ simulates $B$ and $T_A(n) \leqslant T(T_B(n))$.

A simulation with $T(n)=n$ is *real-time* and one with $T(n) \in O(n)$ is *linear time.*

*Nota Bene.* If $B$ accepts in *real-time*, that is $T_B(n)=n$, then there is no difference in power between on-line and off-line mode ($B$ has to read all input and has no time left when the input ends). In all our results the *simulated* machine $B$ is real-time, so only the computation mode of the *simulator* $A$ matters. A lower time bound for simulation by off-line one-way input simulator $B$ is stronger than the same lower bound with simulator $B$ on-line.

Without loss of generality, the tape units in the sequel write only 0's and 1's on the storage tape at the cost of introducing a 'constant delay' for each step. A simulation is *constant delay* if there is a fixed constant $c$ such that there are at most $c$ computation steps in between simulating the $t$th and the $(t+1)$st step, for all $t$. Thus, constant delay with $c=1$ is the same as real-time. Each simulation of constant delay can be sped up to a real-time simulation by expanding the storage alphabet and the size of the finite control, see [5].

## *1.3. Kolmogorov Complexity*

Any of the usual definitions of Kolmogorov complexity [8,2,11] will do for the sequel. To fix thoughts, consider the problem of describing a string $x$ over 0's and 1's. Any computable function $f$ from strings over 0's and 1's to such strings, together with a string $y$, such that $f(y) = x$, is such a description. A descriptional complexity $K_f$ of $x$, relative to $f$ and $y$, is defined by

$$K_f(x|y) = \min\{|d| \mid d \in \{0,1\}^* \,\&\, f(dy) = x\} \ ,$$

where $|x|$ is the positive integer *length* of string $x$. For the *universal* computable partial function $f_0$ we have that, for all $f$ with appropriate constant $c_f$, for all strings $x,y$, $K_{f_0}(x|y) \leqslant K_f(x|y) + c_f$. So the canonical relative descriptional complexity $K(x|y)$ can be set equal to $K_{f_0}(x|y)$. Define the *descriptional complexity* of $x$ as $K(x) = K(x|\epsilon)$, where $\epsilon$ denotes the empty string. Since there are $2^n$ binary strings of length $n$, but only $2^n - 1$ possible shorter descriptions $d$, it follows that $K(x) \geqslant |x|$ for some binary string $x$ of each length. We call such strings *incompressible.* It also follows that $K(x|y) \geqslant |x|$ for some binary string $x$ of each length. As an illustration, a string $x = uvw$ can be specified by $v$, $|x|$, $|u|$ and the bits of $uw$. Thus,

$$K(x) \leqslant K(v) + O(\log|x|) + |uw| \ ,$$

so that with $K(x) \geqslant |x|$ we obtain

$$K(v) \geqslant |v| - O(\log|x|) \ .$$

## *1.4. Descriptions and Self-Delimiting Strings*

In the previous § we formalized the concept of a greatest lower bound on the length of a description. Now we look at feasibility. Variables $x$, $y$, $x_i$, $y_i$ ... denote strings in $\Sigma^*$ for $\Sigma = \{0,1\}$ throughout. Let $x$ be a binary string of length $n$ with $K(x) \geqslant n$. A *description* of $x$ can be given as follows.

(1) A piece of text containing several formal parameters $p_1, \ldots, p_m$. Think of this piece of text as a formal parametrized procedure in an algorithmic language like PASCAL. It is followed by

(2) an ordered list of the actual values of the parameters.

The purpose of this description will be to obtain, by way of contradiction, a description of $x$ of length $n - f(n)$ bits for some unbounded function $f$ of $n$. The piece of text of (1) can be thought of as being encoded over a given finite alphabet, each symbol of which is coded in bits. Therefore, the encoding of (1) as prefix of the binary description of $x$ takes $O(1)$ bits. This prefix is followed by the ordered list (2) of the actual values of $p_1, \ldots, p_m$ in binary. To distinguish one from the other, we encode (1) and the different items in (2) as self-delimiting strings. For natural numbers $n$, let $\text{bin}(n) \in \{0,1\}^*$ be the binary representation of $n$ without leading zeros. For each string $w$, the string $\overline{w}$ is obtained by doubling each letter $a$ in $w$. Let $w' = \overline{\text{bin}(|w|)}01w$. The string $w'$ is called the *self-delimiting* version of $w$. So '1100110101011' is the self-delimiting version of '01011'. The self-delimiting binary version of a positive integer $n$ takes $\log n + 2\log\log n + 2$ bits and the self-delimiting version of a binary string $w$ takes $|w| + 2\log|w| + 2$ bits. All logarithms are base 2 unless otherwise noted. For convenience, we denote the length $|\text{bin}(n)| = \lceil\log(n+1)\rceil + 1$ of a natural number $n$ by "$\log n$".

**Remark 2.1.** Let $x_1 \cdots x_k$ be a binary string of length $n$ on the input tape with the $x_i$'s $(1 \leq i \leq k)$ blocks of equal length $C$. Suppose that $d$ of these blocks are deleted and the relative distances in between deleted blocks are known. We can describe this information by: (1) a formalization of this discussion in O(1) bits, and (2) the actual values of

$$C, m, p_1, d_1, p_2, d_2, \ldots, p_m, d_m \ ,$$

where $m$ $(m \leq d)$ is the number of "holes" in the string, and the literal representation of

$$\hat{x} = \hat{x}_1 \hat{x}_2 \cdots \hat{x}_k \ .$$

Here $\hat{x}_i$ is $x_i$ if it is not deleted, and is the empty string otherwise; $p_j, d_j$ indicates that the next $p_j$ consecutive $x_i$'s (of length $C$ each) are one contiguous group followed by a gap of $d_j C$ bits long. Therefore, $k - d$ is the number of (non-empty) $\hat{x}_i$'s, with

$$k = \sum_{i=1}^{m} p_i + d_i \ \ \& \ \ d = \sum_{i=1}^{m} d_i \ .$$

The actual values of the parameters $C$, $m$, the $p_i$'s, the $d_i$'s and $\hat{x}$ are coded self-delimiting. Then, maximizing over all partitions, the total number of bits needed is no more than (loglog$\leq$log):

$$\textstyle\sum_{i=1}^{k} |\hat{x}_i| + 3d(\log(k/d) + 2) + O(\log n).$$

## *1.5. Crossing Sequences and a Lemma*

For a 1-tape off-line machine $M$ with one-way input we call $M$'s input tape head $h_1$ and its storage tape head $h_2$. Let $h_1(t)$ be the number of polls up to and including step $t$ of $M$. So the input head $h_1$'s position $h_1(t)$ at step $t$ is a nondecreasing function. Let $h_2(t)$ be the position of $h_2$ at step $t$. Let $M(t)$ be the state of $M$ at step $t$. Define a *crossing sequence (c.s.)* associated with a square $s$ (rather, the integer position $s$ of an intersquare boundary) on the storage tape of $M$ as a sequence of *ID*s of the form $(M(t), h_1(t))$ with $h_2(t) = s$. This sequence of *ID*'s gives the values of the parameters when $h_2$ crosses the square $s$ (c.q. intersquare boundary $s$), first (at step $t_1$) from left to right or *vice versa* and then alternating in direction with the $i$th crossing at step $t_i$ $(i > 1)$. We write $|c.s.|$ to denote the number of bits needed to represent the c.s. and $|M| \in O(1)$ for the number of states in $M$.

**Remark 2.2.** Since $h_1$ is nondecreasing, we can represent the $i$th *ID* $(ID_i)$ in a c.s. as follows:

$$ID_1 = (M(t_1), h_1(t_1))$$

$$ID_i = (M(t_i), h_1(t_i) - h_1(t_{i-1})) \quad (i > 1) \ .$$

If a c.s. has $d$ $ID$'s and the length of the input is $n$, then (by § 1.4 and loglog$\leqslant$log):

$$|c.s.| \leqslant 3(d\log|M| + \log k_1 + \cdots + \log k_d + 2d) + O(1) \ ,$$

with $\sum_{i=1}^{d} k_i = n$. Note that

$$|c.s.| \leqslant 6d(\log|M| + \log(n/d)) + O(1)$$

by a standard calculation (i.e. maximize the function).

**Definition 2.1.** Let $x_i$ be a block of input, and $R$ be a tape segment on the storage tape. We say that $M$ *maps* $x_i$ *into* $R$ if $h_2$ never leaves tape segment $R$ while $h_1$ is reading $x_i$. We say $M$ maps $x_i$ *onto* $R$ if $h_2$ traverses the *entire* tape segment $R$ while $h_1$ reads $x_i$.

We prove an intuitively straightforward lemma for one-tape machines with one-way input. The lemma states that a tape segment bordered by short c.s.'s cannot receive a lot of information without losing some. Formally:

**Jamming Lemma.** *Let the input string start with* $x\# = x_1 x_2 \cdots x_k\#$, *with the* $x_i$*'s blocks of equal length C. Let R be a segment of M's storage tape and let l be an integer such that M maps each block* $x_{i_1}, \ldots, x_{i_l}$ *of the* $x_i$*'s into tape segment R. The contents of the storage tape of M, at time* $t_\#$ *when* $h_1(t_\#) = |x\#|$ *and* $h_1(t_\# - 1) = |x|$, *can be reconstructed by using only the blocks* $x_{j_1} \cdots x_{j_{k-l}}$ *which remain from* $x_1 \cdots x_k$ *after deleting blocks* $x_{i_1}, \ldots, x_{i_l}$, *the final contents of R, the two final c.s.'s on the left and right boundaries of R, a description of M and a description of this discussion.*

**Remark 2.3.** Roughly speaking, if the number of missing bits $\sum_{j=1}^{l} |x_{i_j}|$ is greater than the number of added description bits $(< 3(|R| + 2|c.s.|) + O(\log|R|))$ then the Jamming Lemma implies that either $x = x_1 \cdots x_k$ is not incompressible or some information about $x$ has been lost.

**Proof of the Jamming Lemma.** Let the two positions at the left boundary and the right boundary of $R$ be $l_R$ and $r_R$, respectively. We now simulate $M$. Put the blocks $x_j$ of $x_{j_1} \cdots x_{j_{k-l}}$ in their correct positions on the input tape (as indicated by the $h_1$ values in the c.s.'s). Run $M$ with $h_2$ staying to the left of $R$. Whenever $h_2$ reaches point $l_R$, the left boundary of $R$, we interrupt $M$ and check whether the current $ID$ matches the next $ID$, say $ID_i$, in the c.s. at $l_R$ (in the nondeterministic case we also match $M$'s current state and $h_1$'s value). Subsequently, using $ID_{i+1}$, we skip the input up to and including $h_1(t_{i+1})$, adjust the state of $M$ to $M(t_{i+1})$, and continue running $M$. After we have finished left of $R$, we do the same thing right of $R$. At the end we have determined the appropriate contents of $M$'s tape, apart from the contents of $R$, at $t_\#$ (i.e., the time when $h_1$ reaches #). Inscribing $R$ with its final contents from the reconstruction description gives us $M$'s storage tape contents at time $t_\#$. Notice that although there are many unknown $x_i$'s, they are never polled since $h_1$ skips over them because $h_2$ never goes into $R$. $\square$

**Remark 2.4.** If $M$ is nondeterministic, then we need to rephrase "contents of storage tape" by "legal contents of storage tape", which simply means that some computation path for the same input would create this storage tape contents.

## 2. Lower Bounds for Deterministic Simulation

### *2.1. Two Pushdown Stores Versus One Tape: Deterministic Case*

In this § we present a tight lower bound for off-line one-way input deterministic one-tape machines simulating 2 pushdown store machines. The witness language $L$ is defined by:

$$L = \{x_1@x_2@ \cdots @x_k\#y_1@ \cdots @y_l\#(1^{i_1},1^{j_1})(1^{i_2},1^{j_2})...(1^{i_s},1^{j_s}) \mid$$
$$x_p=y_q \;\; \& \;\; p = i_1+...+i_t, \;\; q = j_1+...+j_t \;\; \& \;\; 1 \leqslant t \leqslant s\}. \qquad (2.1)$$

**Theorem 2.1.** *It requires $\Omega(n^2)$ time to deterministically simulate two pushdown stores by one off-line tape with one-way input .*

The theorem will follow from Lemma 2.1 below.

**Lemma 2.1.** *A deterministic off-line one-tape Turing machine with one-way input accepting $L$ requires $\Omega(n^2)$ time.*

*Proof.* Assume, by way of contradiction, that a off-line one-way input deterministic 1-tape machine $M$ accepts $L$ in $T(n) \notin \Omega(n^2)$ time. We derive a contradiction by showing that then some incompressible string must have a too short description.

Assume, without loss of generality, that $M$ writes only 0's and 1's in its storage squares and that $|M| \in O(1)$ is the number of states of $M$. Fix a constant $C$ and the word length $n$ as large as needed to derive the desired contradictions below and such that the formulas in the sequel are meaningful.

First, choose an incompressible string $x \in \{0,1\}^*$ of length $|x| = n$ (i.e., $K(x) \geqslant n$). Let $x$ consist of the concatenation of $k = n / C$ substrings, $x_1, x_2, \ldots, x_k$, each substring $C$ bits long. Let

$$x_1@x_2@ \cdots @x_k\#$$

constructed from $x$ be the initial input segment polled by $M$. Let time $t_\#$ be the step at which $M$ polls #. If more than $k/2$ of the $x_i$'s are mapped *onto* (see Definition 2.1) a contiguous tape segment of size at least $n / C^3$ then $M$ requires $\Omega(n^2)$ time: contradiction. Therefore,

(a) there is a set of contiguous tape segments on the storage tape, each one of length $\leqslant n / C^3$, such that each one out of $k/2$ of the $x_i$'s is mapped *into* (see Definition 2.1) a tape segment from that set. Let $X$ be the (multi)set of these $x_i$'s ($|X| = k/2$).

(b) In the remainder of the proof we restrict attention to the $x_i$'s in this set $X$. Order the elements of $X$ according to the natural order of the left boundaries of the tape segments *into* which they are mapped. Let $x_c$ be the median.

Proof idea: We consider two cases. In the first case we assume that many $x_i$'s in $X$ are mapped (jammed) into a small tape segment $R$; that is, when $h_1$ (the input tape head) is reading them, $h_2$ (the storage tape head) is always in this small tape segment $R$. We show that then, contrary to assumption, $x$ can be compressed (by the Jamming Lemma). In the second case, we assume there is no such 'jammed' tape segment, and that the records of the $x_i$'s in $X$ are spread evenly over the storage tape. In that case, we will arrange the $y_j$'s so that there are many pairs $(x_i,y_j)$'s for which $x_i=y_j$ and $x_i$ and $y_j$ are mapped into tape segments that are far apart. For each of these pairs we will arrange the indices in language $L$ so as to force $M$ to match $x_i$ against $y_j$. Either $M$ spends too much time or we can compress $x$ again, yielding a second contradiction and therefore the lemma.

*Case 1 (jammed).* Assume there are $k / C$ substrings $x_i \in X$ and a fixed tape segment $R$ of length $n / C^2$ on the storage tape such that $M$ maps all of these $x_i$'s into $R$.

We will show that a short program can be constructed which accepts only $x$. Consider the two tape segments of length $|R|$ to the left and to the right of $R$ on the storage tape. Call them $R_l$ and $R_r$, respectively. Choose positions $p_l$ in $R_l$ and $p_r$ in $R_r$ with the shortest c.s.'s in their respective tape segments. These c.s.'s must both be shorter than $n / C^2$, for if the shortest c.s. in either tape segment is $n / C^2$ or longer then $M$ uses $\Omega(n^2)$ time: contradiction. Let tape segment $R_l'$ $(R_r')$ be the portion of $R_l$ $(R_r)$ right (left) of $p_l$ $(p_r)$.

Now, using the description of

- this discussion (including the text of the program below) and simulator $M$ in $O(1)$ bits,
- the values of $n$, $k$, $C=n / k$, and the positions of $p_l,p_r$ in $O(\log n)$ bits,
- at most $k-(k / C)$ of the $x_i$'s that are not mapped into $R_l'RR_r'$, in at most $(k-(k / C))C+3(k / C)(\log C+2)+O(\log n)$ bits (by Remark 2.1),
- the state of $M$ and the position of $h_2$ at time $t_{\#}$ in $O(\log n)$ bits,
- the two c.s.'s at time $t_{\#}$ in at most $6(n / C^2)(\log |M| + \log C^2) + O(1)$ bits (by Remark 2.2), and
- the contents at time $t_{\#}$ of tape segment $R_l'RR_r'$ in at most $3n / C^2 + O(\log n)$ bits (by §1.4),

we can construct a program to check if a string $y$ equals $x$ by running $M$ as follows.

Check if $|y|=|x|$. By the Jamming Lemma (using the above information as related to $M$'s processing of the initial input segment $x_1@ \cdots @x_k\#$) reconstruct the contents of $M$'s storage tape at time $t_{\#}$, the time $h_1$ gets to the first # sign. Divide $Y$ into $k$ equal pieces and form $y_1@ \cdots @y_k$. Simulate $M$, started on the input suffix

$$y_1@ \cdots @y_k\#(1,1)(1,1) \cdots (1,1)$$

($k$ pairs of 1's) from time $t_{\#}$ onwards. By definition $M$ accepts if and only if $y=x$. This description of $x$ takes not more than

$$n-\frac{n}{C}+\frac{3n(5\log C+2\log |M|+3)}{C^2}+O(\log n) \leq \epsilon n$$

bits, for some positive constant $\epsilon<1$ and large enough $C$ and $n$. However, this contradicts the incompressibility of $x$ $(K(x)\geqslant n)$.

*Case 2 (not jammed).* Assume that

(c) for each fixed tape segment $R$, with $|R|=n/C^2$, there are at most $k/C$ substrings $x_i \in X$ mapped into $R$.

Fix a tape segment of length $n/C^2$ that contains median $x_c$. Call this segment $R_c$. By (a), (b) and (c) it follows that a subset of the middle $k/C$ strings $x_i$ in the ordered set $X$ are mapped into $R_c$. Therefore, for large enough $C$ $(C>3)$, at least $k/6$ of the $x_i$'s in $X$ are mapped into the tape right of $R_c$. Let the set of those $x_i$'s be $S_r = \{x_{i_1}, \ldots, x_{i_{k/6}}\} \subset X$. Similarly, let $S_l = \{x_{j_1}, \ldots, x_{j_{k/6}}\} \subset X$, consist of $k/6$ strings $x_i$ which are mapped into the tape left of $R_c$. Without loss of generality, assume $i_1<i_2<\cdots<i_{k/6}$, and $j_1<j_2<\cdots<j_{k/6}$.

Now choose strings $y_l$ as follows. Set $y_1 = x_{i_1}$, $y_2 = x_{j_1}$, $y_3 = x_{i_2}$, $y_4 = x_{j_2}$, and so forth. In general, for all integers $m$, $1\leqslant m\leqslant k/6$,

$$y_{2m} = x_{j_m} \text{ and } y_{2m-1}=x_{i_m} \ , \tag{2.2}$$

We can now define an input prefix for $M$ to be:

$$x_1@\cdots@x_k\#y_1@\cdots@y_{k/3}\# \ . \tag{2.3}$$

*Claim 1.* There exist $k/12$ pairs $y_{2i-1}@y_{2i}$ such that while $h_1$ (the input head) reads them, $h_2$ (the storage tape head) travels a distance less than $n/(4C^2)$.

*Proof of Claim.* If the claim is false then $M$ uses $\Omega(n^2)$ time, a contradiction. □

*Claim 2.* There is a tape segment $R$ in $R_c$ $(R\subset R_c)$ with length $|R|=n/(4C^2)$ such that $k/24$ pairs $y_{2i-1}@y_{2i}$ are all mapped either into the tape right of $R$ or into the tape left of $R$.

*Proof of Claim.* At least half of the $k/12$ pairs $y_{2i-1}@y_{2i}$ are polled starting with $h_2$ either in the right half of $R_c$ or in the left half. The claim then follows by Claim 1. □

Let $R$ be as in Claim 2. By Claim 2 and the choice of the $y_j$'s above, $k/24$ of the $x_i$'s, all from either $S_r$ or $S_l$, are mapped into the tape on one side of $R$ and their corresponding $y_j$'s are mapped into the tape on the other side of $R$ ($x_i$ *corresponds* to $y_j$ if $x_i=y_j$ according to (2.2)). Let the set of these $x_i$'s be $S_x$, and the set of corresponding $y_j$'s be $S_y$. We now know that when $h_1$ reads anything in $S_x$, $h_2$ is on one side of $R$, and when $h_1$ reads anything in $S_y$, $h_2$ is on the other side of $R$. $|S_x|=|S_y|=k/24$. Let the indices of elements in $S_x$ be $a_1<a_2<\ldots<a_{k/24}$, and let the indices of the elements in $S_y$ be $b_1<b_2<\ldots<b_{k/24}$. By our previous arrangement (2.2) we know $x_{a_i}=y_{b_i}$. Now we complete our input to $M$ by appending

$$\#(1^{a_1},1^{b_1})(1^{a_2-a_1},1^{b_2-b_1})\cdots(1^{a_{k/24}-\sum_{l=1}^{k/24-1}a_l},1^{b_{k/24}-\sum_{l=1}^{k/24-1}b_l}) \ . \tag{2.4}$$

Determine a position $p$ in $R$ which has the shortest c.s. of $M$'s computation on the combined input (2.3)(2.4). If this c.s. is longer than $n/C$ then $M$ uses time $\Omega(n^2)$: contradiction. Therefore, assume it has length at most $n/C$. Then again we can construct a short program $P$, to accept $x$ by a 'cut and paste' argument, and show that it yields too short a description of $x$.

For a candidate input string $z$, program $P$ first partitions $z$ into $z_1@ \cdots @z_k$ and compares the appropriate literal substrings with the literally given strings in $\{x_1, \ldots, x_k\} - S_x$. The strings in $S_x$ are given in terms of the operation of $M$: to compare the appropriate substrings of $z$ with the $x_i$'s in $S_x$, we simulate $M$. First prepare an input according of the form (2.3) as follows. Put the elements of $\{x_1, \ldots, x_k\} - S_x$ literally into their correct places on the input tape, filling the places for $x_i$'s in $S_x$ arbitrarily. For the $y_i$'s in (2.3) substitute the appropriate substrings $z_i$ of candidate $z$ according to scheme (2.2) *i.e.*, use $z_{j_m}$ for $y_{2m}$ and $z_{i_m}$ for $y_{2m-1}$ $(1 \leq m \leq k/6)$. Note that among these are all those substrings of candidate $z$ which have not yet been checked against the corresponding substrings of $x$. Adding string (2.4) above completes the input to $M$.

Without loss of generality, assume that $S_x$ is mapped into the tape left of $R$ and $S_y$ is mapped into the tape right of $R$. Using the c.s. at point $p$ we run $M$ such that $h_2$ always stays in right of $p$ ($S_y$'s side). Whenever $h_2$ encounters $p$, we check if the current $ID$ matches the corresponding one in the c.s.. If it does then we use the next $ID$ of the c.s. to continue. If in the course of this simulation process $M$ rejects or there is a mismatch (that is, when $h_2$ gets to $p$, $M$ is not in the same state or $h_1$'s position is not as indicated in the c.s.), then $z \neq x$. Otherwise $z = x$. Note, that it is possible for $M$ to accept (or reject) on the left of $p$ ($S_x$'s side). However, once $h_2$ crosses $p$ right-to-left for the last time $M$ does not read any substring $z_i$ substituted for the members of $S_y$ any more and all other $z_i$'s are 'good' ones (we have already checked them). Therefore, if the crossing sequence of $ID$s at $p$ of $M$'s computation for candidate $z$ match those of the prescribed c.s. then we know that $M$ accepts.

Now describe $x$ by:

- this discussion (including the text of the program described above) and simulator $M$ in O(1) bits,
- the values of $n$, $k$, $C = n/k$, and the position of $p$ in $O(\log n)$ bits,
- $\leq n - n/24 + O(\log n)$ bits for the concatenated $k - k/24$ substrings $x_i$ of $x$ which are not in $S_x$ (by §1.4), together with
- $\leq 3(k - |S_x|)(\log(k/(k - |S_x|)) + 2) + O(\log n) < 12n/C + O(\log n)$ bits for the indices of these $x_i$'s to place them correctly on the input tape (by Remark 2.1),
- $\leq (6n/C)(\log |M| + \log C) + O(1)$ bits for the c.s. of length $n/C$ at $p$ (by Remark 2.2), and

- $\leqslant k(\log 3 + 2) + O(\log n)$ bits for indices of the $k/3$ indices out of $k$ of the $y_i$'s in (2.3) (by Remark 2.1).

Therefore, this description of $x$ takes not more than

$$n - \frac{n}{24} + \frac{6n(\log |M| + \log C + 3)}{C} + O(\log n) \leqslant \epsilon n$$

bits for some positive $\epsilon < 1$ and large enough $C$ and $n$. This contradicts the incompressibility of $x$ ($K(x) \geqslant n$) again.

Case 1 and Case 2 complete the proof of the lemma. □

**Proof of Theorem 2.1.** Obviously, $L$ can be accepted in linear time by a 2-tape machine. For two deterministic pushdown stores we define a language $L_p$ which is essentially $L$ in (2.1).

$$L_p = \{x_k @ \cdots @ x_1 \# y_l @ \cdots @ y_1 \# (1^{i_1}, 1^{j_1})(1^{i_2}, 1^{j_2}) \cdots (1^{i_t}, 1^{j_t}) \mid x_p = y_q$$
$$\& \; p = i_1 + \ldots + i_t, \; q = j_1 + \ldots + j_t \; \& \; 1 \leqslant t \leqslant s \} .$$

$L_p$ can be accepted on-line in linear time by a deterministic 2-pushdown store machine in the obvious way. The lower bound above holds also for $L_p$ by about the same proof. By padding the 'tail' of the strings in $L_p$ we obtain a language $L_p^{pad}$ which can be accepted in real-time by a deterministic 2-pushdown machine and for which the lower bound proof of Lemma 2.1 works as well. (Replace

$$p = i_1 + \ldots + i_t, \; q = j_1 + \ldots + j_t$$

in the definition of $L_p$ by

$$\sum_{i=1}^{p-1} |x_i| + p = i_1 + \ldots + i_t, \; \sum_{i=1}^{q-1} |y_i| + q = j_1 + \ldots + j_t$$

to obtain $L_p^{pad}$.) This concludes the proof of Theorem 2.1. □

## *2.2. One Queue Versus One Tape: Deterministic Case*

Only in this § and §3.4 $g(n) \in \Omega(f(n))$ means "there is a positive constant $\delta$ such that $g(n) \geqslant \delta f(n)$ infinitely often". Everywhere else the results hold for the stronger variant: "there are positive constant $\delta$ and positive integer $n_0$ such that $g(n) \geqslant \delta f(n)$ for all $n \geqslant n_0$".

We present a tight lower bound for the deterministic simulation of one queue by one off-line tape one-way input. This is the first nontrivial such lower bound for off-line simulation. The witness language is the *queue language*:

$$L_q = \{ w \in \{0, 1, \bar{0}, \bar{1}\}^* \mid uv \in L_q \,\&\, v \in \{\bar{0}, \bar{1}\}^* \,\&\, (w = u0v\bar{0} \text{ or } w = u1v\bar{1}) \} . \qquad (2.5)$$

**Remark 2.5.** This is the quintessential queue language: with '0/1' meaning 'append 0/1 to the rear' and '$\bar{0}/\bar{1}$' meaning 'delete 0/1 from the front', the strings in $L_q$ determine precisely all computation histories of a binary queue starting and ending empty. Therefore, a device accepting $L_q$ simulates a binary queue. $L_q$ is the queue equivalent of the Dyck set on two generators for the pushdown store (cf. §3.4).

**Theorem 2.2.** *It requires $\Omega(n^2)$ time to deterministically simulate one queue by one off-line tape with one-way input.*

The theorem will follow from Lemma 2.2.

**Lemma 2.2.** *A deterministic off-line one-tape Turing machine with one-way input accepting $L_q$ requires $\Omega(n^2)$ time.*

*Proof.* Assume, by way of contradiction, that an off-line deterministic 1-tape machine $M$ with one-way input accepts $L_q$ in time $T(n) \notin \Omega(n^2)$. We derive a contradiction by showing that then some incompressible string has a too short description. Without loss of generality, $M$ has a semi-infinite storage tape $[0,\infty)$ on which it writes only 0's and 1's, and $|M|$ is the number of states of $M$. Fix a constant $C$ and the word length $n$ as large as needed to derive the desired contradictions below and such that the formulas in the sequel are meaningful. First, choose an incompressible string $x \in \{0,1\}^*$ of length $n$ (i.e., $K(x) \geq n$). We consider the behavior of $M$ on a particular input. This is a string $z \in L_q$ selected in an adversary way with respect to $M$ so as to maximize the running time. Divide $M$'s storage tape in segments $[0,n/3)$ and $[n/3,\infty)$.

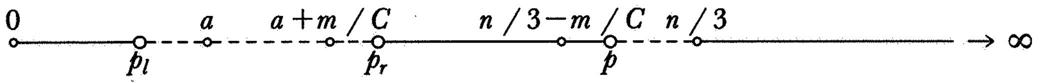

**Figure.** $p_l$, $p_r$ and $p$ are "bottlenecks" with shortest c.s.'s in $[a-m/(4C),a)$, $[a+m/C,a+5m/(4C))$ and $[n/3-m/C,n/3)$, respectively.

For all $s \in \{0,1\}^*$, let $\bar{s}$ be the string resulting from $s$ by replacing all 0's by $\bar{0}$'s and all 1's by $\bar{1}$'s. Let $x=vw$ and let $\bar{x}=\bar{v}\bar{w}$ ($|v|=|\bar{v}|=m \leq n$), such that $z$ depends on $M$ in the following way:

$$z = v_1\bar{u}_1 \cdots v_k\bar{u}_k\ , \quad v=v_1 \cdots v_k,\ \bar{v}=\bar{u}_1 \cdots \bar{u}_k,$$

such that all polls with $h_2$ on $[0,n/3)$ have $h_1$ scanning a symbol in $\{0,1\}$ and all polls with $h_2$ on $[n/3,\infty)$ have $h_1$ scanning a symbol in $\{\bar{0},\bar{1}\}$. However, if the number of polls with $h_2$ on $[0,n/3)$ reaches $n$ then all subsequent polls have $h_1$ scanning a symbol in $\{\bar{0},\bar{1}\}$. Therefore, $z$ is well defined ($|z| \leq 2n$) and, because $T(2m) \notin \Omega(m^2)$ by contradictory assumption, $m \in \Omega(n^{1/2}) \cap O(n)$.

**Remark 2.6.** In the *augmented* crossing sequences in this § we record not only the number of polls in between crossings, but also the number of polls of unbarred 0's and 1's in between crossings. Moreover, the last crossing also records acceptance or rejection. An augmented c.s. has at most double the number of bits of the normal c.s. as estimated in

Remark 2.2.

*Case 1.* Assume, by way of contradiction, that in the accepting computation on $z$ at least $2m / C$ polls of unbarred symbols occur with $h_2$ on a particular $(m / C)$-length tape segment $R = [a, a + m / C)$ contained in $[0,n / 3)$. Let $z_1$ be the prefix of $z = z_1 \bar{z}_2$ such that $z_1$ ends with 0 or 1 and $\bar{z}_2$ contains only $\bar{0}$'s and $\bar{1}$'s. Now all unbarred symbols in $z_1$ are polled with $h_2$ on $[0,n / 3)$ and all barred symbols in $z_1$ are polled with $h_2$ on $[n / 3,\infty)$. Moreover, $z_1$ contains all of $v$. Consider the two tape segments $R_l$ and $R_r$ of length $|R| / 4$ left and right of $R$. Choose positions $p_l$ in $R_l$ and $p_r$ in $R_r$ with the shortest c.s.'s in their respective tape segments. These c.s.'s must both be shorter than $m / C^2$, for if the shortest c.s. in either tape segment is longer than $m / C^2$ then $M$ uses $T(2m) \in \Omega(m^2)$ time: contradiction. We show that a short program can be constructed which accepts only $v$. Using the description of:

- this discussion (including the recovery algorithm below) and of simulator $M$ in O(1) bits,
- the value of $n$, $m$, $|z_1|$ and $a$ in $O(\log n)$ bits,
- the location of $p_l$, $p_r$ in $O(\log n)$ bits,
- 2 augmented c.s.'s at $p_l$, $p_r$ in $\leqslant (24m / C^2)(2 \log C + \log |M|) + O(1)$ bits (by Remark 2.6),
- the bits of $v$ polled with $h_2$ outside tape segment $[p_l, p_r]$, concatenated in the order in which they occur in $x$, form a string $y$. The self-delimiting version of $y$ takes not more than $m - 2m / C + O(\log n)$ bits (by assumption and §1.4),
- The final contents of $[p_l, p_r]$ at time $t_{\#}$, where $h_1(t_{\#}) = |z_1| + 1$ and $h_1(t_{\#}) = |z_1|$, in not more than $3m / (2C) + O(\log n)$ bits (by §1.4),

we can construct a program to check if a string $v' \in \{0,1\}^*$ equals $v$. Check $|v'| = m$. Let $y'$ be the result of deleting the bits in $v'$ in the same positions as used to obtain $y$ from $v$. These positions are determined by the augmented crossing sequences. Check $y' = y$. If a test is negative then $v' \neq v$; else $v'$ can only differ from $v$ on positions where $v$'s bits are polled with $h_2$ on $[p_l, p_r]$. Try all $|z_1|$-length prefixes of strings in $L_q$ which can be constructed from $v'$ and its barred version $\bar{v}'$. Let $z_1'$ be such a candidate, that is, there is a $z' \in L_q$ and $z' = z_1' z_2'$, $|z'| = 2m$ and $|z_1'| = |z_1|$. Run $M$ on input $z_1'$ with $h_2$ staying right of $p_l$ and left of $p_r$. Whenever $h_2$ reaches $p_l$ $[p_r]$ we interrupt $M$ and check if the current *ID* and ratio between polled barred/unbarred symbols in the computation matches the corresponding *ID* in the augmented c.s. at $p_l$ $[p_r]$. If so, then go on running $M$ using the next *ID* by skipping the input up to and including the new value of $h_1$, barred and unbarred symbols in proportion as indicated by the augmented c.s.. Everything matches up to the end of processing $z_1'$, and the final tape contents of $[p_l, p_r]$ at $t_{\#}$ (when $h_1$ polls the $|z_1'\#|$th input symbol) equals the one from the description above, only if $v' = v$. For suppose the contrary and $v' \neq v$. By the Jamming Lemma, reconstruct the contents of $M$'s storage tape and the *ID* at time $t_{\#}$ for input $z_1\#$, using the above description. Let $h_2$ be outside $[p_l, p_r]$ at time $t_{\#}$ of the computation on $z_1\#$. (If $h_2$ is inside interchange $z_1$ and $z_1'$ below.) Cut and paste the computations on $z_1$ and $z_1'$ such that $M$ runs on input $z_1$ with $h_2$ outside $[p_l,p_r]$ and on input $z_1'$ with $h_2$ inside $[p_l,p_r]$. Call the thus constructed input string $\zeta$, $|\zeta| = |z_1|$. By construction the concatenation of the ordered sequence of unbarred bits of $\zeta$ equals $v'$ and the ordered sequence of barred bits equals that of $z_1$. By construction too, the computation on $\zeta$ matches the description of the computation on $z_1$, and therefore $\zeta\#$ drives $M$ into the same final tape contents and $ID$ at time $t_\#$ as does $z_1\#$. Hence, $M$ must either accept both $z_1z_2$ and $\zeta z_2$ or reject them both. But $z_1z_2 \in L_q$ and $\zeta z_2 \notin L_q$ because $\zeta z_2$ is composed from $v'$ and $\bar{v}$: contradiction.

This description of $v$ takes not more than:

$$O(\log n) + \frac{24m\,(2\log C + \log|M|)}{C^2} + m - \frac{m}{2C} \leqslant \epsilon m$$

bits, for some positive constant $\epsilon<1$ and large enough $C$ and $n$. However, this contradicts the incompressibility of $x$ since that implies $K(v) \geqslant m - O(\log n)$ with $m > n^{1/2}$.

*Case 2.* Assume that for each fixed tape segment $R$ in $[0,n/3)$, with $|R| = m/C$, there are at most $2m/C$ polls of unbarred symbols with $h_2$ on $R$. Therefore, there are at most $2n/3$ polls of unbarred symbols with $h_2$ on $[0,n/3)$, and consequently all barred symbols in $z$ are polled with $h_2$ on $[n/3,\infty)$. Choose the point $p$ with the shortest c.s. in $[(n/3)-(m/C), n/3)$. This c.s. is shorter than $m/C^2$ since otherwise the running time $T(2m)$ of $M$ is $\Omega(m^2)$: contradiction.

Now describe $v$ by:

- this discussion (including the text of the program to retrieve $v$ below) and simulator $M$ in $O(1)$ bits,
- the values of $n$, $m$, and the position of $p$ in $O(\log n)$ bits,
- the augmented c.s. at $p$ in $\leqslant (12m/C^2)(2\log C + \log|M|) + O(1)$ bits (by Remark 2.6),
- the string $y$ of concatenated bits of $v$ polled with $h_2$ on $[p, n/3)$ in $\leqslant 2m/C + O(\log n)$ bits (by assumption and §1.4),

we can construct a program to check if a string $v' \in \{0,1\}^*$ equals $v$. Check $|v'| = m$. Let $y'$ be the result of deleting the bits in $v'$ in the same positions as used to obtain $y$ from $v$. These positions are determined by the augmented crossing sequence at $p$. Check $y' = y$. If a test is negative then $v' \neq v$; else $v'$ can only differ from $v$ on positions where $v$'s bits are polled with $h_2$ on $[0,p)$. Try all strings in $L_q$ which can be constructed from $v'$ and its barred version $\bar{v}'$. Let $z'$ be such a candidate. Run $M$ with $h_2$ staying left of $p$ with input $z'$. Whenever $h_2$ reaches $p$ we interrupt $M$ and check if the current $ID$ in the computation matches the corresponding $ID$ in the c.s. at $p$. If so, then go on running $M$ using the next $ID$ by skipping the input up to and including the new value of $h_1$, barred and unbarred symbols in proportion as indicated by the augmented c.s.. If everything matches, then run $M$ with $h_2$ staying right of $p$. Everything matches up to the end of processing $z'$ and $M$ accepts only if $v' = v$. For, suppose the contrary and $v' \neq v$. By definition running $M$ on input $z$ on both sides of $p$ will match the description. Let the final accepting position of $h_2$ for $M$'s computation on $z$ be right of $p$. (If it is left of $p$ interchange $z$ and $z'$ below.) Cut and paste the computations on $z'$ and $z$ such that $M$ runs on input $z'$ with $h_2$ left of $p$ and on input $z$ with $h_2$ right of $p$. The resulting computation matches the description of $z$'s computation, and the corresponding input $\zeta$ is accepted by $M$. But $\zeta$ is composed of $v'$ and $\overline{v}$ and therefore $\zeta \notin L_q$: contradiction.

The description of $v$ takes not more than:

$$O(\log n) + \frac{12m\,(2\log C + \log |M|)}{C^2} + \frac{2m}{C} \leqslant \epsilon m$$

bits, for some positive constant $\epsilon<1$ and large enough $C$ and $n$. However, this contradicts the incompressibility of $x$ since $K(v) \geqslant m - O(\log n)$ with $m \geqslant n^{1/2}$.

Since $m \in \Omega(n^{1/2}) \cap O(n)$, Case 1 and Case 2 complete the proof of the lemma. □

**Proof of Theorem 2.2.** The standard binary queue recognizes $L_q$ in real-time in the obvious way. The theorem then follows from Lemma 2.2. □

## 3. Lower Bounds for Nondeterministic Simulation

### *3.1. Two Pushdown Stores Versus One Tape: Nondeterministic Case*

In this §, we present a nearly optimal lower bound on the time required to simulate two deterministic pushdown stores by one off-line nondeterministic tape with one-way input. Define $L$ by

$$L = \{ x_1 @ x_0 @ x_2 @ x_0 \cdots @ x_t @ x_0 \# x_1 x_2 \cdots x_t \# \mid x_i \in \{0,1\}^* \text{ for } i=0,\ldots,t \} .$$

**Theorem 3.1.** *It requires $\Omega(n^{1.5} / \sqrt{\log n})$ time to simulate two deterministic pushdown stores off-line by one nondeterministic tape with one-way input.*

The theorem will follow from Lemma 3.1.

**Lemma 3.1.** *It requires $\Omega(n^{1.5} / \sqrt{\log n})$ time to accept $L$ by any off-line one-way input nondeterministic one-tape machine.*

*Proof.* Assume by way of contradiction that a off-line one-way input nondeterministic one-tape machine $M$ accepts $L$ in time $T(n) \notin \Omega(n^{1.5} / \sqrt{\log n})$. Without loss of generality $M$ writes only 0's and 1's on its storage tape and the number of states is $|M| \in O(1)$. Fix a constant $c$ and the word length $n$ as large as necessary to obtain the desired contradictions below and such that the formulas are meaningful.

Choose an incompressible string $x$ of length $n$ $(K(x) \geqslant n)$. Equally partition $x$ into $x_0 x_1 \cdots x_k$ into $k+1$ blocks $x_i$ of length $n/(k+1)$ each. Let

$$y = x_1 @ x_0 @ x_2 @ x_0 \cdots @ x_k @ x_0 \# x_1 x_2 \cdots x_k \# \qquad (3.1)$$

be the input string polled by $M$. Observe that $|y| < 3n$. Since $M$ accepts this input $y$, let us fix a shortest accepting computation, say $P$, of $M$ on input $y$. We shall show that the length of $P$ is $\Omega(n^{1.5} / \sqrt{\log n})$.

Choose

$$k = \left[ \frac{n}{\log n} \right]^{1/2} . \tag{3.2}$$

Consider the $k$ pairs $x_i@x_0@$ in $y$. If more than $k/2$ of them are mapped *onto* tape segments of sizes larger than $n/c$, then $M$ uses time $\Omega(n^{1.5}/\sqrt{\log n})$: contradiction. Therefore, $M$ must map at least $k/2$ pairs $x_i@x_0@$ *into* tape segments of sizes at most $n/c$. Let $S$ be the set of such pairs. Time $t_{\#}$ is the step at which $M$ polls the first # marker. We consider the computation up to time $t_{\#}$ and distinguish two cases:

*Case 1 (jammed).* Assume there do not exist two pairs in $S$ that are mapped into two tape segments $n/c$ apart. Since also each pair in $S$ is mapped into a tape segment of size at most $n/c$, all pairs in S are mapped into a single tape segment $R$ of size $3n/c$. Let $R_l$ and $R_r$ be the left and right adjacent tape segments of $R$, $|R_l| = |R| = |R_r|$. Find a point $l$ in $R_l$ and a point $r$ in $R_r$ with the shortest c.s. in $R_l$ and $R_r$, respectively. These c.s.'s must both be shorter than $d$,

$$d = \frac{1}{c} \left[ \frac{n}{\log n} \right]^{1/2} , \tag{3.3}$$

for if the shortest c.s. on either tape segment has length $d$ or more then $M$ uses $\Omega(n^{1.5}/\sqrt{\log n})$ time: contradiction.

We can reconstruct the contents of the storage tape at time $t_{\#}$ by the Jamming Lemma. The reconstruction only requires the following descriptions:

- a description of this discussion (including the text of the program below) and a description of $M$ in O(1) bits,
- the values of $n$ and $k$ and the positions of $l$ and $r$ in $O(\log n)$ bits,
- a description of the $\leq k/2$ elements (with indices) of $\{x_0, x_1, \ldots, x_k\} - \{x_i \mid x_i x_0 \in S\}$, which requires at most $n/2 + 5k + O(\log n)$ bits (by Remark 2.1),
- two c.s.'s that require at most $12d(\log|M| + \log(n/d)) + O(1)$ bits (by Remark 2.2),
- the final tape contents at time $t_{\#}$ of tape segments $R$, $R_l$, and $R_r$ which requires no more than $9n/c + O(\log n)$ bits (by §1.4).

We can find $x$ as follows. For each $y$ such that $|y| = |x|$, divide $y = y_0 y_1 \cdots y_k$ into the same number of equal length substrings as $x$. Check if $y_0 = x_0$. Assume that $\#y_1 \cdots y_k\#$ is the input suffix in (3.1) and continue to simulate $M$ from $t_{\#}$ on, with the storage tape constructed as above, with the additional information of $M$'s state and the position of storage head $h_2$ at time $t_{\#}$ in $O(\log n)$ extra bits. Obviously, $M$ accepts iff $y = x$.

This description of $x$ takes not more than

$$\frac{n}{2} + 5k + 12d(\log|M| + \log\frac{n}{d}) + \frac{9n}{c} + O(\log n) \leq \epsilon n$$

bits for large enough $c$ and $n$ and some positive $\epsilon$, $\epsilon<1$, by (3.2) and (3.3). This contradicts the incompressibility of $x$ ($K(x)\geqslant n$). One might worry about the nondeterminism here, but note that the nondeterminism does not matter. We can simply simulate $M$ nondeterministically in the above, making sure that the c.s.'s are matched. A shortest path with the right sequence of guesses for a successful computation is computation path $P$.

*Case 2 (not jammed).* Assume there are two pairs, say $x_i@x_0$ and $x_j@x_0$, that are mapped $n/c$ apart. Therefore, the distance between the two tape segments *onto* which these two pairs are mapped is at least $n/c$. Let $R_0$ be the tape segment in between and $|R_0|\geqslant n/c$. As before, we look for a point $p$ in $R_0$ with shortest c.s in $R_0$. If the shortest c.s. has length $d$ as in (3.3) or more, then $M$ runs in time $\Omega(n^{1.5}/\sqrt{\log n})$: contradiction.

We use this shortest c.s. to reconstruct $x_0$ below. But notice that a simple minded approach such as finding a shortest c.s. in the middle is not enough here, because some $x_0$ can be mapped to both sides of the c.s.. To overcome this difficulty, observe that since the size of shortest c.s. is $d$ and the input is one-way there can only be this order of bits in $x_0$ that are mapped to both sides of the c.s.. A description of all of $x$ is given by:

- A description of this discussion (including the text of the program below) and a description of $M$ in O(1) bits,
- the values of $n$, $k$, and the location of $p$ in $O(\log n)$ bits,
- a literal description of $x_1x_2\cdots x_k$ in $nk/(k+1)+O(\log n)$ bits (by §1.4),
- a description of the c.s. at $p$ of length $d$ in $6d(\log|M|+\log(n/d))+O(1)$ bits (by Remark 2.2). In each $ID$ of this shortest c.s., we add a bit which specifies the last bit read by $h_1$. This to overcome the mentioned problem. This can be stored in $M$'s state (by doubling the number of states).

A program to test a candidate string $y$ for equality with $x$ proceeds as follows. If $|y|\neq n$ then $y\neq x$. If $|y|=|x|$ then divide $y$ in $k+1$ equal length substrings, $y=y_0y_1\cdots y_k$. Check if $y_i=x_i$ for all $i>0$. If not, then $y\neq x$; otherwise, arrange the $y_i$'s (including $y_0$) in their correct positions on the input tape. Note that the nondeterminism of $M$ does not matter; we can assume that the program always guesses right so as to follow computation path $P$.

Using the c.s. at point $p$, we run $M$ on this input but only the parts of the computation with storage head $h_2$ left of $p$. Every time $h_2$ meets the c.s., check if the current $ID$ matches the current state of $M$ (including the bit added in the first ● above), and then use the next $ID$ to continue the simulation. $y=x$ iff the simulation ends with everything matching all the way. This description of $x$ takes not more than

$$\frac{nk}{k+1}+6d(\log|M|+\log\frac{n}{d})+O(\log n)\leqslant n-\frac{n}{2k}+12d\log\frac{n}{d}\qquad\text{(by (3.3))}$$

$$\leqslant n-\epsilon\sqrt{(n\log n)}\qquad\text{(by (3.2), (3.3))}$$

bits, for some positive $\epsilon$ ($\epsilon\geqslant 1/2-12/c$) and contradicts the incompressibility of $x$

$(K(x) \geqslant n)$, for large enough $c$ and $n$. □

**Proof of Theorem 3.1.** The language $L$ can be easily accepted by a deterministic two tape machine in real-time. For pushdown stores, we modify $L$ by reversing the string $x_1 x_2 \cdots x_t$ following the # sign. The modified $L$ can be accepted by $\overline{M}$ with two deterministic pushdown stores in linear time as follows: put $x_1$ in stack1, put the next $x_0$ in both stack1 and stack2, put $x_2$ in stack2, put the next $x_0$ in both stack1 and stack2, put $x_3$ in stack1, and so on. When the input head reads #, $\overline{M}$ starts to match in the obvious way. To make this process real time we further modify $L$ by simply putting a $1^{2|x_0|}$ padding after every other reversed $x_i$. Since all these changes do not invalidate our lower bound proof in Lemma 3.1, the proof is complete. □

Combined with Theorem A (below) recently proved in [9], we essentially close the gap for 1-tape versus 2 pushdown stores, nondeterministic case, answering open question 1 of [3].

**Theorem A.** *Two pushdown stores or one queue can be simulated by one nondeterministic tape in* $O(n^{1.5}\sqrt{\log n})$ *time for both on-line and off-line machines.*

*3.2. One Queue Versus One Tape: Nondeterministic Case*

A tight lower bound for one tape simulating one queue in the deterministic case has been obtained in §2.2. Here we obtain an $\Omega(n^{4/3}/\log^{2/3} n)$ lower bound for the nondeterministic case. By [9] $O(n^{1.5}\sqrt{\log n})$ is an upper bound, cf. Theorem A §3.1.

**Theorem 3.2.** *It requires* $\Omega(n^{4/3}/\log^{2/3} n)$ *time to simulate one deterministic queue off-line by one nondeterministic tape with one-way input.*

*Proof Idea.* At first glance, one might think the language $L$ in §3.1 can be used and therefore an $\Omega(n^{1.5}/\sqrt{\log n})$ nearly optimal lower bound can be obtained. Unfortunately, on second thought, one queue probably can not accept $L$ in linear time. But the following observation can be made. As long as the $|x_i|$'s $(0 \leqslant i \leqslant k)$ are chosen such that $\sum_{i=0}^{k} |x_i| \in O(n)$, then a 1-queue machine would be able accept the corresponding subset of $L$ in linear time if it could 'count fast.' That is, make sure that the relative sizes of $x_i$'s are correct. How does a queue count fast? Probably no way. Nonetheless, this leads us to the following language

$$L_{pad} = \{x_1@x_0@x_2@x_0 \cdots @x_k@x_0\#x_1 1^{|x_0|} \cdots x_k 1^{|x_0|}\# 1^{k|x_0|^2} \mid x_i \in \{0,1\}^* \text{ for } 0 \leqslant i \leqslant k\} \quad ,$$

where the $1^{|x_0|}$'s and $1^{k|x_0|^2}$ are added to ensure that $L_{pad}$ is acceptable by a real-time deterministic 1-queue machine, even when the size of $x_0$ grows too large. We claim that a deterministic 1-queue machine can accept $L_{pad}$ in real-time, but an off-line one-way input nondeterministic 1-tape machine needs $\Omega(n^{4/3}/\log^{2/3} n)$ time in the worst case. The algorithm for accepting $L_{pad}$ by 1 queue is as follows.

(1) Put $x_1 x_0 \cdots x_k x_0$ into the queue.

(2) Match $x_1, \ldots, x_k$ by the input head and the front end of the queue, while deleting all $x_i$'s ($i>0$) and copying the $x_0$'s back to the rear of the queue while reading the $1^{|x_0|}$ paddings.

(3) Match all $x_0$'s bit by bit in $k\,|x_0|^2$ time, while the input head scans the padding. That is, rotate the entire string of $x_0$'s by unstoring from the front and storing to the back of the queue while matching and deleting the first bit of all copies of $x_0$ in the process. Repeat this with $x_0$ minus its first bit, and so on.

The lower bound can be proved in the same way as the one in Theorem 3.1, for the particular choice of parameters: $k=n^{1/3}/\log^{2/3}n$, $|x_0|=(n\log n)^{1/3}$ and $|x_i|=(n\log n)^{2/3}$ for all $i$, $1\leqslant i\leqslant k$. The present lower bound $\Omega(n^{4/3}/\log^{2/3}n)$ is less than the lower bound in Theorem 3.1 as a consequence of the padding. The current choice of parameters yields the optimum lower bound achievable for this case using a proof like in §3.1. We omit the details and refer the reader thither. Informally, the current lower bound is obtained by maximizing $t(n)$ (= lower bound on the running time $T(n)$ of the simulator) under the constraints: $t(n)\in O(n^{1.5}\sqrt{\log n})$, $k\,|x_0|^2\in O(n)$, $t(n)\in O(kn)$, and $(t(n)/n)\log(n^2/t(n))\in O(|x_0|)$. □

This is the first nontrivial lower bound for one tape versus one queue in the nondeterministic case.

## *3.3. Two Tapes Versus One Tape: Nondeterministic Case*

Unlike the results presented above which are independent of [10], Theorem 3.3 is based on and presupposes the approach of [10].

For the nondeterministic off-line one-way input case of one tape versus two tapes, Maass [10] obtained an

$$\Omega(\frac{n^2}{(\log n)^2\log\log n})$$

lower bound. Aiming for the same lower bound, although in a different context, Freivalds (Theorem 2 in [4], without proof) also considered this problem. Both [10,4] independently construct two similar ingenious languages (the language of [4] is less complete).

In [10] a very general language $L_I$ was introduced, but only a simple subset, $\hat{L}$, of it was used. The language $\hat{L}$ can be defined as follows (without loss of generality let $k$ be odd).

$$\begin{aligned}\hat{L} = \{\ & b_0^1 b_1^1 \cdots b_k^1 \\ & b_0^2 b_0^3 b_1^2 b_2^2 b_1^3 b_3^2 \cdots b_{2i}^2 b_i^3 b_{2i+1}^2 \cdots b_{k-1}^2 b_{(k-1)/2}^3 b_k^2 \\ & b_0^4 b_{(k+1)/2}^3 b_1^4 b_2^4 b_{(k+3)/2}^3 b_3^4 \cdots b_{2i \bmod (k+1)}^4 b_i^3 b_{(2i+1) \bmod (k+1)}^4 \cdots b_{k-1}^4 b_k^3 b_k^4 \\ & \mid b_i^1 = b_i^2 = b_i^3 = b_i^4 \text{ for } i = 0, \ldots, k\ \}\end{aligned}$$

The length of each $b_i^j$ (a binary string) may be different. We can also define a delimited version $L^*$ of $\hat{L}$ where every $b_i^j$ in $\hat{L}$ is replaced by $*b_i^j*$ of an uniform length.

The language $B$ constructed in [4] is similar (but less complete). Here is the construction of [4]. Let $B'$ consist of all strings

$$a(1)b(1)a(2)b(2) \cdots a(2n)b(2n)2a(2n)b(2n)b(2n-1)a(2n-1)$$
$$b(2n-2)b(2n-3) \cdots a(n+1)b(2)b(1)$$

in $\{0,1\}^*\{2\}\{0,1\}^*$, $n \geqslant 0$. The set $B$ is defined to be the set of all strings $0x$ or $1y$, where $x \in B'$ and $y \in \overline{B'}$. ($\overline{B'}$ is the complement of $B'$.) In [4] it is stated that a 1-tape nondeterministic on-line TM requires $\Omega(n^2)$ time to accept $B$. However, in [9] it is proved that this is not the case (Theorems B and C below).

**Theorem B.** *$\hat{L}$ ($L^*$ and $B$) can be accepted in $O(n^2 \log\log n / \sqrt{\log n})$ time by a 1-tape nondeterministic on-line machine.*

**Theorem C.** *Language $B$ can be accepted by a 1-tape nondeterministic on-line machine in time $O(n^{1.5}\sqrt{\log n})$.*

In the rest of this §, trying to meet the upper bound of Theorem B, we improve the lower bound of [10] to

$$\Omega(\frac{n^2}{\log n \, \log\log n}) \ .$$

Since the following theorem is based on the approach in that paper, we assume the reader is familiar with the details of [10] and only point out where and how the improvement is made*.

**Theorem 3.3.** *It requires $\Omega(n^2 / (\log n \, \log\log n))$ time to simulate two deterministic tapes off-line by one nondeterministic tape with one-way input.*

We show that the language $L^*$ (and $\hat{L}$) requires $\Omega(n^2 / (\log n \, \log\log n))$ time for off-line one-way input nondeterministic one-tape machines. In [10] Maass proved an important combinatorial lemma (Theorem 3.1 in that reference) which is generalized as follows.

**Lemma 3.3.1.** *Let $S$ be a sequence of numbers from $\{0, \ldots, k-1\}$, where $k = 2^l$ for some $l$. Assume that every number $b \in \{0, \ldots, k-1\}$ is somewhere in $S$ adjacent to the number $2b \pmod k$ and $2b \pmod k + 1$. Then for every partition of $\{0, \ldots, k-1\}$ into two sets $G$ and $R$ such that $S = G \cup R$ and $|G|, |R| > k/4$ there are at least $k/(c \log k)$ (for some fixed $c$) elements of $G$ that occur somewhere in $S$ adjacent to a number from $R$.*

The proof of this lemma is a simple reworking of the proof in [10]. A $k / \sqrt{\log k}$ upper bound corresponding to the lower bound in this lemma is contained in [9].

Notice that any sequence $S$ in $L^*$ satisfies the requirements in Lemma 3.3.1. Let $n$ be the length of a incompressible string that is divided into $k = n / \log\log n$ blocks. From these $k$ blocks we construct a sequence $S$ in $L^*$. A new idea is to find *many*, instead of just *one* as in [10] 'deserts' on the storage tape.

* Zvi Galil, Ravi Kannan and Endre Szemeredi have announced a $\Omega(n^2 / (m^4 \log m))$ lower bound, with $m = \log^* n$, on the time to simulate 2 tapes by 1 nondeterministic off-line tape with one-way input. ($m = \log^* n$ if the $m$ times iterated logarithm $\log^{(m)} n = \log\log \cdots \log n \leqslant 1$ and $\log^{(m-1)} n > 1$.)

**Lemma 3.3.2 (Many Deserts Lemma).** *For some constant C, and for large enough n, there are* $I = (\log n) / C$ *tape segments* $D_1, D_2, \ldots, D_I$ *on the storage tape such that,*

(1) *for all* $i \neq j$, $D_i \cap D_j = \emptyset$;

(2) *for each* $i$, $|D_i| = n / (c^{12} \log n)$, *where* $c \geqslant 2$ *is the constant in Lemma 3.3.1;*

(3) *for each* $i$, *at least* $k / 4 = n / (4 \log\log n)$ *blocks are mapped to each side of* $D_i$.

**Proof sketch.** Again we only give the ideas behind the proof. Divide the whole storage tape into tape segments of length $n / (c^{13} \log n)$. By the Jamming Lemma, no tape segment can hold more than $n / (c^{11} \log n)$ blocks. By a standard counting argument, we can find tape segments $D_1, D_2, \ldots, D_{(\log n)/C}$ for some constant $C$ in the 'middle' of the storage tape such that (1), (2), and (3) above are satisfied. □

**Proof sketch of Theorem 3.3.** To prove Theorem 3.3, we apply the proof of [10] for each desert $D_i$ in Lemma 3.3.2. Instead of using Theorem 3.1 of [10] we use Lemma 3.3.1 above. Notice that since each $D_i$ is 'short', the total number of blocks mapped outside $D_i$ is more than $k - (k / (c^9 \log k))$. Therefore Lemma 3.3.1 can be applied. Now for each tape segment $D_i$, $M$ has to spend $\Omega(n^2 / (\log^2 n \log\log n))$ time. Summing the amounts of time $M$ spends on each of the $\Omega(\log n)$ tape segments, yields the $\Omega(n^2 / (\log n \log\log n))$ lower bound. □

## *3.4. Nondeterministic On-Line Simulation*

Nondeterministic computation is *off-line*: it is assumed that a fixed input is contained on a separate input tape, and is the same for all legal computation paths of the computing device. In §3.3 we saw a nearly square lower bound on the time for one nondeterministic tape unit, off-line and with one-way input, to accept a particular language which can be real-time accepted by a deterministic Turing machine with two single-head storage tapes. The adversary technique used in §2.2 prompts the consideration of on-line nondeterminism in between determinism and standard off-line nondeterminism. Define an *on-line* nondeterministic device as a device in a black box with input terminals and output terminals. In each step, the device in the black box can make a fixed constant number of guesses or nondeterministic choices. A *legal* computation path consists of an allowed sequence of such guesses with an accompanying sequence of polled input commands. That is, the sequence of commands polled by the machine in the sequence of polling states on that legal computation path. The distinction with conventional nondeterminism is contained in the fact that this sequence of input commands need not be the same for all computation paths. That is, when the machine enters a polling state *any* input command can be polled at the input terminal. We can think of the device as splitting itself in distinct copies at each choice. The input terminals of the different copies can produce different input commands at the next poll.

*Definition.* A language $L$ is accepted *on-line* by a nondeterministic machine $M$ in time $T(n)$ if for each word $w$ in $L$, there is a legal computation path $P$ in the computation tree such that $|P| \leqslant T(|w|)$ and each prefix $w'$ of $w$ which belongs to $L$ is accepted on a prefix $P'$ of $P$ with $|P'| \leq T(|w'|)$.

In this definition no assumptions are made about the sequence of polled inputs being identical along all computation paths. When machine $M$ polls for input it may receive *any* input command from the input terminal. Consequently, for this notion of on-line nondeterministic computation the adversary strategy of §2.2 can be followed on each separate computation path of an on-line nondeterministic single-head tape unit as on the single computation path of the deterministic version above. Let $P_1, P_2$ be two deterministic pushdown stores. Let the input commands be of the form "op($d$, $P_i$)" with *operation* op $\in$ {pop, push}, the bit $d$ concerned is $d \in \{0,1\}$ and the stack $P_i$ concerned is given by $i \in \{1,2\}$. A sequence of input commands is a *valid* sequence if at the execution of each 'pop($d$, $P_i$)' command in the sequence the top bit of $P_i$'s stack is $d$. Define the obvious *language* consisting of all valid sequences of input commands. (This is actually the *shuffle* of two disjoint copies of the Dyck set on two types of brackets: $D_2^{(1)}$ with brackets $\{0_1, 1_1, \overline{0_1}, \overline{1_1}\}$ and $D_2^{(2)}$ with brackets $\{0_2, 1_2, \overline{0_2}, \overline{1_2}\}$) The analog for the single *queue* is $L_q$ in §2.2. One can obtain, according to the discussion and the definitions above, by an adversary argument as in §2.2:

**Theorem 3.4** *There is a language which is accepted in real-time by two pushdown stores (respectively by one queue) but which requires $\Omega(n^2)$ time for acceptance by one on-line nondeterministic single-head tape unit.*

Acknowledgements

We thank Chanderjit Bajaj, Juris Hartmanis, Luc Longpre, Walter Savitch, Joel Seiferas and Yaacov Yesha for helpful discussions on various topics contained in this paper.

References

[1] Aanderaa, S.O., "On k-tape versus (k-1)-tape real-time computation," pp. 75-96 in Complexity of Computation, ed. R.M. Karp, American Mathematical Society, Providence, R.I. (1974).

[2] Chaitin, G.J., "Algorithmic Information Theory," *IBM J. Res. Dev.*, vol. 21, pp.350-359, 1977.

[3] Duris, P., Z. Galil, W. Paul, and R. Reischuk, "Two nonlinear lower bounds for on-line computations," *Information and Control*, vol. 60, pp.1-11, 1984.

[4] Freivalds, R., "Probabilistic machines can use less running time," *Information Processing*, vol. 77, pp.839-842, 1977.

[5] Hartmanis, J. and R.E. Stearns, "On the computational complexity of algorithms," *Trans. Amer. Math. Soc.*, vol. 117, pp.285-306, 1969.

[6] Hennie, F.C. and R.E. Stearns, "Two tape simulation of multitape Turing machines," *J. Ass. Comp. Mach.*, vol. 4, pp.533-546, 1966.

[7] Hopcroft, J.E. and J.D. Ullman, *Formal Languages and their Relations to Automata.* Addison-Wesley, 1969.

[8] Kolmogorov, A.N., "Three approaches to the quantitative definition of information," *Problems in Information Transmission*, vol. 1, no. 1, pp.1-7, 1965.

[9] Li, M., "Simulating two pushdowns by one tape in O(n**1.5 (log n)**0.5) time", 26th Annual IEEE Symposium on the Foundations of Computer Science, 1985.

[10] Maass, W., "Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines," *Trans. Amer. Math. Soc.*, To appear.

[11] Paul, W.J., J.I. Seiferas, and J. Simon, "An information theoretic approach to time bounds for on-line computation," *J. Computer and System Sciences*, vol. 23, pp.108-126, 1981.

[12] Paul, W.J., "On-line simulation of k+1 tapes by k tapes requires nonlinear time," *Information and Control*, pp.1-8, 1982.

[13] Rabin, M.O., "Real-time computation," *Israel J. of Mathematics*, vol. 1, pp.203-211, 1963.

[14] Vitányi, P.M.B., "On the simulation of many storage heads by one," *Theoretical Computer Science*, vol. 34, pp.157-168, 1984.

[15] Vitányi, P.M.B., "Square time is optimal for the simulation of a pushdown store by an oblivious one-head tape unit," *Information Processing Letters*, vol. 21, pp.87-91, 1985.

[16] Vitányi, P.M.B., "An N**1.618 lower bound on the time to simulate one queue or two pushdown stores by one tape," *Information Processing Letters*, vol. 21, 1985.