Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

Ming Li, P.M.B. Vitanyi

Tape versus queue and stacks:
The lower bounds (Revised)

Computer Science/Department of Algorithmics & Architecture Report CS-R8752 November

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Tape versus Queue and Stacks: The Lower Bounds (Revised)

Ming Li

Harvard University, Aiken Computation Laboratory
Cambridge, Massachusetts

Paul M.B. Vitányi

Centrum voor Wiskunde en Informatica
and
Faculteit Wiskunde en Informatica
Universiteit van Amsterdam
Amsterdam, The Netherlands

Originally submitted in early 1984 as two separate papers.

ABSTRACT

Several new 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, for both the deterministic case and the nondeterministic case. 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
Keywords & Phrases: multitape Turing machine, stack, queue, pushdown stores, determinism, nondeterminism, on-line, off-line, time complexity, lower bounds, simulation by one tape, Kolmogorov complexity
Note: to appear in *Information and Computation*.

The work of the first author was supported in part by the National Science Foundation under Grants DCR-8301766, DCR-8606366, and Office of Naval Research Grant N00014-85-K-0445. The work of the second author was supported in part by the Office of Naval Research under Contract N00014-85-K-0168, by the U.S. Army Research Office under Contract DAAG29-84-K-0058, by the National Science Foundation under Grant DCR-83-02391, and by the Defence Advanced Projects Agency (DARPA) under Contract N00014-83-K-0125. The first author is on leave from the Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210. Part of this work was performed while he was at the Department of Computer Science, Cornell University, Ithaca, NY 14853.

Report CS-R8752
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. Introduction

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

Section_2.1.Simulation of 2 stacks (stack = pushdown store) by 1 tape requires $\Omega(n^2)$ time. This is optimal.

Section_2.2.Simulation of 1 queue by 1 tape requires $\Omega(n^2)$ time. This is optimal.

Nondeterministic case:

Section_3.1.Simulating 2 stacks by 1 tape requires $\Omega(n^{1.5/\sqrt{\log n}})$ time. The corresponding upper bound is $O(n^{1.5\sqrt{\log n}})$ in [10].

Section_3.2.Simulating 1 queue by 1 tape requires $\Omega(n^{4/3}/\log^{2/3} n)$ time. The corresponding upper bound is $O(n^{1.5\sqrt{\log n}})$ in [10].

Section_3.3.Simulating 2 tapes by 1 tape requires $\Omega(n^2/(\log n \ \mathrm{loglog}\, n))$ time. This is a multiplicative factor $\log n$ improvement of the $\Omega(n^2/(\log^2 n \ \mathrm{loglog}\, n))$ lower bound in [11].

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 $n \log n$ [7], and by 1-tape Turing machines in time n^2 [8]. In [15] 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 [13] 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 [14]. Thus, the simulation by two tapes was shown to be nearly optimal; for simulation by one 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 first author at Cornell and the second author at CWI Amsterdam, obtained a square lower bound on the time to simulate two tapes by one tape. All three rely on the excellent notion discovered independently by Solomonoff [12] and Kolmogorov [9] in the early 1960s. 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 [16], a

* A claim for an $\Omega(n^{2-\epsilon})$ lower bound for simulation of two tapes by both one deterministic tape and one nondeterministic tape was first circulated by W. Maass in August 1983. An extended abstract containing this result was submitted to STOC by November 1983. The final STOC paper of May 1984 (submitted February 1984) contained the optimal $\Omega(n^2)$ lower bound for the deterministic simulation of two tapes by one tape. In M. Li: 'On 1 tape versus 2 stacks,' Tech. Rept. TR-84-591, Dept. Comp. Sci., Cornell University, January 1984, the $\Omega(n^2)$ lower bound was obtained for the simulation of two stacks by one deterministic tape. In: P.M.B. Vitányi, 'One queue or two pushdown stores take square time on a one-head tape unit,' Tech. Rept. CS-R8406, Centre for Mathematics and Computer Science, Amsterdam, March 1984, the $\Omega(n^2)$ lower bound was obtained for both the simulation of two stacks and one queue by one deterministic tape. Maass's and Li's result were for off-line computation with one-way input, while Vitányi's result was for on-line computation.

$\Omega(n^{1.5})$ 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 [17] this lower bound was improved to $\Omega(n^2)$, while [18] 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 [11] 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 requires $\Omega(n^2)$ time in the deterministic case and $\Omega(n^2/\log^{\alpha} n)$ time, for some small α, in the nondeterministic case.

In Section 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 [11] which shows the result for two tapes) and another 'adversary' argument for a *single* queue. In Section 3.1-Section 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. Known corresponding upper bounds are $O(n^{1.5}\sqrt{\log n})$ in [10]. Simulating two tapes requires $\Omega(n^2/(\log n \log\log n))$ time, which is a multiplicative factor $\log n$ improvement of [11]. In a successor paper, together with Luc Longpré, we have extended the present work with a comprehensive study stressing queues in comparison to stacks and tapes. (M. Li, L. Longpré and P.M.B. Vitányi, The power of the queue, submitted to SIAM J. on Computing.)

1.2. Storage, Computation Mode and Simulation

Let us briefly review some of the concepts involved in this paper. The machines we consider have *storage* consisting of either linear lists with sequential access (*single-head tape units*) or last-in-first-out storage (*pushdown stores*) or first-in-first-out storage (*queues*), c.f. [8]. *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, respectively (see [8] for details). We are interested in *off-line* computation with *one-way input* (no back-up on the input). 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). 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* [10]. All results below are about this mode of computation.

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 either a 0 or a 1 indicating *rejection* or *acceptance*, respectively. 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.) 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. (For a nondeterministic machine the presence or absence of an end-of-input marker makes no difference since it can 'guess'.)

A machine A simulates a machine B if, when started on the same input string, A accepts if and only if B accepts. A machine A simulates machine B in time $T(n)$:

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

A simulation with $T(n)=n$ is *real-time* and one with $T(n) \in O(n)$ is *linear time*. If B accepts in *real-time*, that is $T_B(n)=n$, then on-line mode and off-line mode coincide. In all our results the *simulated* machine B is real-time, so only the computation mode of the *simulator* A matters. Note, that 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 considered 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 between simulating the tth and the $(t+1)$st steps, for all t. Thus, constant delay with $c=1$ is the same as real-time. Each simulation of constant delay can be speeded up to a real-time simulation by expanding the storage alphabet and the size of the finite control, see [6].

1.3. Kolmogorov Complexity

Any of the usual definitions of Kolmogorov complexity [2, 9, 12, 13] 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. The *descriptional* complexity K_f of x, *relative* to f and y, is defined by

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

where $|x|$ is the (nonnegative integer) *length* of string x. For the *universal* computable partial function f_0 we know that, for all f, there is a constant c_f such that for all strings x, y, $K_{f_0}(x \mid y) \leq K_f(x \mid y)+c_f$. So the *canonical relative descriptional* complexity $K(x \mid y)$ can be set equal to $K_{f_0}(x \mid y)$. Define the *descriptional* complexity of x as $K(x)=K(x \mid \varepsilon)$, where ε denotes the empty string ($|\varepsilon|=0$). Since there are 2^n binary strings of length n, but only 2^n-1 possible shorter descriptions d, it follows that $K(x) \geq |x|$ for some binary string x of each length. We call such strings *incompressible*. It also follows similarly that, for any length n and any binary string y, there is a binary string x of length n such that $K(x \mid y) \geq |x|$.

A string $x=uvw$ can be specified by v, $|x|$, $|u|$ and the bits of uw. Thus,

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

Hence, if $K(x) \geq |x|$ then we obtain

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

1.4. Descriptions and Self-Delimiting Strings

In the previous Section we formalized the concept of a greatest lower bound on the length of a description. Now we look at feasibility. Throughout the present paper the variables x, y, x_i, y_i ... will denote strings in $\{0,1\}^*$. Let x be a binary string of length n with $K(x) \geq n$. 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 requires $O(1)$ 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 $\mathrm{bin}(n) \in \{0,1\}^*$ be the binary representation of n without leading zeros. For each string w, the string $\bar{w}$ is obtained by doubling each letter in w. Let $w' = \overline{\mathrm{bin}(|w|)}01w$. 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 requires $\log n + 2\log\log n + 2$ bits and the self-delimiting version of a binary string w requires $|w| + 2\log |w| + 2$ bits. All logarithms are base 2 unless otherwise noted. For convenience, we denote the length $|\mathrm{bin}(n)| = \lceil \log(n+1) \rceil + 1$ of a natural number n by "$\log n$".

Remark 1.1. 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

$$C, m, p_1, d_1, p_2, d_2, \ldots, p_m, d_m \tag{1.1}$$

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

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

The actual values of the parameters in (1.1) and $\hat{x}$ are coded self-delimiting. Then, by the convexity of the logarithm function, the total number of bits needed to describe the above information is no more than (loglog≤log):

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

1.5. Crossing Sequences

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

Remark 1.2. Since h_1 is nondecreasing, we can represent the ith 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-many IDs and the length of the input is n, then (by Section 1.4 and loglog≤log):

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

with $\sum_{i=1}^{d} k_i = n$. Maximizing the function above it follows that

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

1.6. The Jamming Lemma

Definition 1.1. 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 1.3. 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 $x_{j_1} \cdots x_{j_{k-1}}$ 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. 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. •

Remark 1.4. 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 Section 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:

$$L = \{x_1@x_2@ \cdots @x_k\#y_1@ \cdots @y_l\#(1^{i_1},1^{j_1})(1^{i_2},1^{j_2})...(1^{i_t},1^{j_t}) : \tag{2.1}$$

$$x_p=y_q \ \ \& \ \ (p = i_1+...+i_t,\ q = j_1+...+j_t) \ \ \& \ \ 1\le t\le s \} .$$

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

Proof. (I). Assume, by way of contradiction, that an 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 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., $K(x)\ge n$). Let x consist of the concatenation of $k=n/C$ substrings, $x_1,x_2,\ldots,x_k$, each substring C bits long. Let

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

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 1.1) a contiguous tape segment of size at

least n/C^3 then M requires $\Omega(n^2)$ time, which is a contradiction. Therefore,

(a) There is a multiset X of $k/2$ x_i's (each x_i can occur more than once in X), and a set of tape segments on the storage tape (each tape segment a block of contiguous tape cells of length $\leq n/C^3$), such that each x_i in X is mapped *into* (see Definition 1.1) a tape segment from that set.

(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 $T(n) \in \Omega(n^2)$.

Case 1 (jammed). Assume there are k/C blocks $x_i \in X$ 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 1.1),
- 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 1.2), and
- the contents at time $t_\#$ of tape segment $R_l'RR_r'$ in at most $3n/C^2 + O(\log n)$ bits (by Section 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 (2.1) of L we have that M accepts if and only if $y=x$.

This description of x requires not more than

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

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

Case 2 (not jammed). Assume that

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

Fix a tape segment of length n/C^2 into which median x_c is mapped. 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, $1\le m\le k/6$,

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

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 force M to "check" (2.2) by completing the input with suffix

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

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 *only* x by a 'cut and paste' argument, and show that it yields too short a description of x.

Using the description of

- this discussion (including the text of the program P below) 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 Section 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 1.1),
- $\leq(6n/C)(\log|M|+\log C)+O(1)$ bits for the c.s. of length n/C at p (by Remark 1.2), and
- $\leq k(\log 3+2)+O(\log n)$ bits for indices of the $k/3$ indices out of k of the y_i's in (2.3) (by Remark 1.1).
- $\leq (6k/24)(\log 24+2)+O(\log n)<2n/C$ bits for indices a_i and b_i $(1\leq i\leq k/24)$ in (2.4) (by Remark 1.1).

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

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 to 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 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 $z\neq x$. 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 in prefix (2.3) 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. By construction the outlined program P accepts the string $z=x$. Suppose P also accepts $z' \neq x$. Then the described computation of M accepts for candidate z'. We can cut and paste the two computations of M with candidate strings z and z' using the computation with z left of p and the computation with z' right of p. Then string (2.3)(2.4) composed from x and z' according to (2.2) is accepted by M. Since z and z' must differ in blocks corresponding to the blocks of x in S_x this string is not in L as defined in (2.1): contradiction.

The description of x requires not more than

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

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

Case 1 and Case 2 complete the proof that $T(n) \in \Omega(n^2)$.

(II). Obviously, L can be accepted in linear time by a 2-tape machine. For two deterministic pushdown stores we define a language L_{push} which is essentially L in (2.1).

$$L_{push} = \{x_k @ \cdots @ x_1 \# y_l @ \cdots @ y_1 \# (1^{i_1}, 1^{j_1})(1^{i_2}, 1^{j_2}) \cdots (1^{i_s}, 1^{j_s}) :$$

$$x_p = y_q \ \ \& \ \ (p = i_1 + \ldots + i_t, q = j_1 + \ldots + j_t) \ \ \& \ \ 1 \leq t \leq s \} .$$

L_{push} can be accepted on-line in linear time by a deterministic 2-pushdown store machine in the obvious way. By padding the 'tail' of the strings in L_{push} we obtain a language L_{push}^{pad} which can be accepted in real-time by a deterministic 2-pushdown machine and for which the lower bound proof (I) works as well. (Replace

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

in the definition of L_{push} 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_{push}^{pad}.)

By (I) and (II) the proof of Theorem 2.1. is complete. •

2.2. One Queue Versus One Tape: Deterministic Case

We present a tight lower time bound for deterministic simulation of one queue by one off-line tape with one-way input.

Remark 2.1. Only in this Section 2.2, $g(n) \in \Omega(f(n))$ means "there is a positive constant δ such that $g(n) \geq \delta f(n)$ infinitely often". Everywhere else the results hold for the stronger variant of Ω: "there exist a positive constant δ and a positive integer n_0 such that $g(n) \geq \delta f(n)$ for all $n \geq n_0$".

The witness language L_q involves a process that pushes symbols at the rear of a queue which occasionally "rolls" symbols from the front of the queue to the rear of the queue, in which case the new symbol is marked. This process is performed real-time by a queue but requires

$\Omega(n^2)$ time by a tape. As a first approach, let $a_1a_2\cdots a_n$ be an n-length sequence of bits to be pushed consecutively in a queue Q, $a_i\in\{0,1\}$, and let sequence $b_1b_2\cdots b_n$, $b_i\in\{0,1,\varepsilon\}$, of length m ($m\le n$), be the sequence of bits which are consecutively popped from the queue. The ith element of $w=(a_1,b_1)\cdots(a_n,b_n)$ consists of the pair (a_i,b_i), meaning that a_i is appended to the rear of Q and b_i (possibly ε) is deleted from the front of the queue. Q accepts w if it accepts every proper prefix of w and either $b_n=\varepsilon$ or b_n equals the current front element. For technical reasons in the proof below, we have to complicate this scheme. On one hand, $a_1a_2\cdots a_n$ which has been pushed in Q, needs to remain stored in Q forever. On the other hand, to force Q to operate correctly we need to be able to pop it. In the final approach, to combine both requirements, each pair (a_i,b_i) causes Q not only to push a_i and to pop b_i (possibly ε), but also to push b_i anew. Below we show that a scheme of unbarred and barred a_i's, related to whether or not the associated 'pop' b_i's are ε or not, makes it possible to retrieve the complete sequence of a_i's, in the order they have been pushed originally, from the queue contents at each instant.

Formally, the *witness* language L_q over $\Sigma=\{0,1\}\times\{0,1,\bar{0},\bar{1},\varepsilon\}$ is defined as the language accepted by a queue Q as follows:

- Initially, Q contains the empty word ε.
- For all $i\ge1$, input '(a_i,b_i)' to Q is interpreted by Q as '*if* $b_i=\varepsilon$ *then* append a_i to the rear *else* append $\bar{a}_i$ to the rear; delete b_i up front; append b_i to the rear.' (Here 'action1;action2;action3' denotes the sequential execution of action1, action2 and action3.)
- Q accepts $(0,\varepsilon)$ and $(1,\varepsilon)$. A word $(a_1,b_1)\cdots(a_n,b_n)$ is accepted if $(a_1,b_1)\cdots(a_{n-1},b_{n-1})$ is accepted and either $b_n=\varepsilon$ or b_n equals the front element of Q after processing $(a_1,b_1)\cdots(a_{n-1},b_{n-1})$. All other words are rejected.

Lemma 2.1. *Let* $\le_{prefix}$ *mean 'is a prefix of.' If*

$$(a_1,b_1)(a_2,b_2)\cdots(a_n,b_n)\in L_q \tag{2.5}$$

then for all i, $1\le i\le n$,

$$b_1b_2\cdots b_i \le_{prefix} \hat{a}_1b_1\hat{a}_2b_2\cdots\hat{a}_ib_i\ ,$$

where for any pair $(a,b)\in\Sigma$ *we define* $\hat{a}$ *by*

$$\hat{a}=a \text{ if } b=\varepsilon$$

$$\hat{a}=\bar{a} \text{ if } b\neq\varepsilon\ .$$

Proof. Since the left hand side of the inequality describes the sequence popped from the queue, and the right hand side describes the sequence pushed on the queue. •

The properties of words of form (2.5) we need in the sequel are expressed in the following three lemmas.

Lemma 2.2. *For a word of the form (2.5),* $|\hat{a}_1b_1\hat{a}_2b_2\cdots\hat{a}_ib_i|-|b_1b_2\cdots b_i|=i$ *for all* i, $1\le i\le n$.

Proof. Obvious. •

Lemma 2.3. *For a word of the form (2.5) we can reconstruct* $a_1a_2\cdots a_n$ *from the* n-

length suffix of $\hat{a}_1 b_1 \hat{a}_2 b_2 \cdots \hat{a}_n b_n$.

Proof. Let the n-length suffix be $x_1 x_2 \cdots x_n$ with $x_i \in \{0,1,\bar{0},\bar{1}\}$ $(1 \le i \le n)$. By (2.5) one of the following two cases must hold (note that the combination $x_{n-1} \in \{0,1\}$ *and* $x_n \in \{\bar{0},\bar{1}\}$ is impossible):

(a) Assume $x_n, x_{n-1} \in \{0,1\}$. Then $a_n = x_n$ and $b_n = \varepsilon$ by (2.5). Consequently, $x_1 x_2 \cdots x_{n-1}$ is the $(n-1)$-length suffix of $\hat{a}_1 b_1 \hat{a}_2 b_2 \cdots \hat{a}_{n-1} b_{n-1}$ by definition of $x_1 \cdots x_n$.

(b) Assume $x_{n-1} \in \{\bar{0},\bar{1}\}$. Then $\bar{a}_n = x_{n-1}$ and $b_n = x_n$ by definition of $x_1 \cdots x_n$. Consequently, $x_n x_1 x_2 \cdots x_{n-2}$ is the $(n-1)$-length suffix of $\hat{a}_1 b_1 \hat{a}_2 b_2 \cdots \hat{a}_{n-1} b_{n-1}$ by (2.5). (Because b_n is the last *popped* symbol which has been appended to the rear of the queue, it is the last symbol to have been popped from the front of the queue. Therefore, to repush the queue contents just before (a_n, b_n) is processed, we delete suffix $a_n b_n$ from $x_1 x_2 \cdots x_n$ and prefix the remaining string with b_n.)

Iterating this reasoning n times we recover all of $a_1 a_2 \cdots a_n$. This proves the lemma. •

Lemma 2.4. *For a word of the form (2.5) with* $|b_1 \cdots b_n| = m$, *we can reconstruct* $a_1 a_2 \cdots a_{m/2}$ *from* $b_1 \cdots b_n$.

Proof. Let

$$b_1 b_2 \cdots b_n = x_1 x_2 \cdots x_m, \quad x_i \in \{0,1,\bar{0},\bar{1}\} \ (1 \le i \le m).$$

By (2.5), $x_1 \cdots x_m$ is a prefix of $\hat{a}_1 b_1 \cdots \hat{a}_n b_n$, so $\hat{a}_1 = x_1$.

(a) If $x_1 \in \{0,1\}$ then $a_1 = x_1$ and $b_1 = \varepsilon$. Consequently, $x_2 \cdots x_m$ is the $(m-1)$-length prefix of $\hat{a}_2 b_2 \cdots \hat{a}_n b_n$.

(b) If $x_1 \in \{\bar{0},\bar{1}\}$ then $\bar{a}_1 = x_1$ and $b_1 = x_2$. Consequently, $x_3 \cdots x_m$ is the $(m-2)$-length prefix of $\hat{a}_2 b_2 \cdots \hat{a}_n b_n$.

Iterating this reasoning $m/2$ times we recover all of $a_1 a_2 \cdots a_{m/2}$. This proves the lemma. •

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

Proof. With the description of L_q we have already indicated how a queue recognizes this language in real-time. Thus, we only need to show the lower bound. 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 too short a description. Without loss of generality, it can be assumed that 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. The input is polled by head h_1 and the storage tape head is h_2. The positions at time t are denoted by $h_1(t)$ and $h_2(t)$. By t_i we denote the time when the ith input command is polled, i.e., $h_1(t_i) = i$ and $h_1(t_i - 1) = i - 1$. 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. Below we show that $T(m) > m^2/4C^3$, for some m, $\sqrt{n}/C \le m \le n$, which contradicts the assumption and proves the theorem.

First, choose an incompressible binary string $x = x_1 \cdots x_n$ of length n (i.e., $K(x) \ge n$). We consider the behavior of M on a fixed input string $z = (x_1, y_1) \cdots (x_n, y_n)$, which is uniquely

determined by x as follows. Define the y_i's ($1 \le i \le n$) inductively by:

(1) M starts its computation with $y_1 = \varepsilon$. So first (x_1, y_1) is polled.

(2) Let t_i be the time at which M polls (x_i, y_i), after accepting $(x_1, y_1) \cdots (x_{i-1}, y_{i-1})$. If $h_2(t_i) \in [0, n/4)$ then $y_i = \varepsilon$, else $y_i \neq \varepsilon$. In the latter case y_i is determined uniquely from $(x_1, y_1) \cdots (x_{i-1}, y_{i-1})$ by using the relation $y_1 y_2 \cdots y_i \le_{prefix} \hat{x}_1 y_1 \cdots \hat{x}_{i-1} y_{i-1}$, that is, using (2.5) and the fact that $y_1 = \varepsilon$ by (1).

Proof idea: We consider two cases. In the first case we assume that many (x_i, y_i)'s 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 the assumption, x can be compressed (by the Jamming Lemma). In the second case, we assume there is no such 'jammed' tape segment. We consider a prefix of z such that half of M's polls with h_1 on this prefix take place with h_2 on $[n/4, \infty)$, such that symbols are deleted from the front of the simulated queue which were appended to the rear in polls with h_2 on $[0, n/4)$. Then many x_i's are pushed at the rear of the queue with h_2 far left of the $n/4$th tape square and popped by matching y_j's from the front of the queue, with h_2 far right of the $n/4$th tape square. Either M spends too much time running back and forth to match these symbols, or we can compress x, yielding a second contradiction. After fixing the input length n, we focus the discussion on input prefixes of length $m \le n$, for technical reasons which will become apparent below.

Case 1 (Jammed). Let m be any integer such that $\sqrt{n}/C \le m \le n$. Fix m, and consider the m-length prefix $z(m)$ of z. By (2.5), if z is in L_q then so is each prefix of z, so in particular $z(m) \in L_q$. Assume, by way of contradiction, that in the accepting computation on $z(m)$ at least $2m/C$ polls occur, with h_2 on a particular (m/C)-length tape segment $R = [a, a+m/C)$. 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 $T(m) > m^2/4C^3$ time, which is a contradiction. (If $0 \le a < m/4C$ then set $|R_l| = a$, so that $R_l R R_r \subset [0, \infty)$. Choose $p_l = 0$ and note that the length of the associated c.s. can be set to 0.) We show that a short program can be constructed which accepts only x. Let u be the string consisting of the bits of $x_1 \cdots x_m$ polled with h_2 outside tape segment $[p_l, p_r]$, concatenated in the order in which they occur in x. Using the description of:

- this discussion (including the recovery algorithm below) and of simulator M in $O(1)$ bits,
- the values of n, m, C, a and the locations of p_l, p_r in $O(\log n)$ bits,
- two c.s.'s at p_l, p_r in $\le (12m/C^2)(\log C^2 + \log |M|) + O(1)$ bits (by Remark 1.2),
- the self-delimiting version of u in not more than $m - 2m/C + O(\log n)$ bits (by Section 1.4),
- the bits $x_{m+1} \cdots x_n$ in $n - m + O(\log n)$ bits (by Section 1.4),
- the final contents of $[p_l, p_r]$ at time t_{m+1}, the state of M at time t_{m+1} and $h_2(t_{m+1})$, in not more than $3m/(2C) + O(\log n)$ bits (by Section 1.4),

we can construct a program to check if a string $x' \in \{0,1\}^*$ equals x. Check $|x'| = n$ and $x'_{m+1} \cdots x'_n = x_{m+1} \cdots x_n$. Reconstruct the contents of M's storage tape at time t_{m+1}, after processing $z(m) = (x_1, y_1) \cdots (x_m, y_m)$, where t_{m+1} is the time when h_1 polls (x_{m+1}, y_{m+1}). Do the

reconstruction by running M on all n-length candidate strings, one after the other, like in the proof of the Jamming Lemma. Unlike the latter, we also run M in between p_l and p_r. Note, that y is determined uniquely by x, n and M's operation, as in the definition of z above. Hence, each candidate for x determines a unique candidate for z. If for some candidate string z' everything in M's computation matches the description above, then, as in the Jamming Lemma, we have reconstructed the tape contents of M at time t_{m+1} after processing z. Simulate M from time t_{m+1} onwards on an input suffix

$$(0,y_{m+1})(0,y_{m+2})\cdots(0,y_{2m}) \tag{2.6}$$

with $|y_{m+1}y_{m+2}\cdots y_{2m}|=m$, and such that M accepts for the chosen y_i's $(m+1\le i\le 2m)$. It is easy to see from (2.5), that there is such a suffix (2.6) for which M accepts if $x'_1x'_2\cdots x'_m=x_1x_2\cdots x_m$. In that case $x'=x$, and by (2.5) and Lemma 2.2, $y_{m+1}y_{m+2}\cdots y_{2m}$ equals the m-length suffix of $\hat{x}_1y_1\cdots\hat{x}_my_m$. By Lemma 2.3, we can retrieve $x_1x_2\cdots x_m$ from this suffix. Suppose, there is a $x'\neq x$ such that

$$z'(m)=(x'_1,y'_1)(x'_2,y'_2)\cdots(x'_m,y'_m) \tag{2.7}$$

matches the description above, and $z'(m)$ drives M into the same configuration at time t'_{m+1} of M's $(m+1)$th poll in its computation, as the configuration into which $z(m)$ drives M at time t_{m+1}. Consequently, the concatenation of (2.7) and (2.6) is also accepted by M. Note, that x' differs from x only in the first m bits, and more in particular in those bits polled with h_2 positioned in tape segment $[p_l,p_r]$. We can cut and paste the computations based on $z'(m)$ inside $[p_l,p_r]$ and based on $z(m)$ outside $[p_l,p_r]$, and still have M accept. The 'cut and paste' computation is accepting up to the $(m+1)$th poll because both computations satisfy the description above, and afterwards because the two computations are identical from the $(m+1)$st poll onwards. Let the resulting string be composed in the obvious way from $x_1\cdots x_m$ and $x'_1\cdots x'_m$ be $\chi(m)=\chi_1\cdots\chi_m$ with $\chi_i\in\{x_i,x'_i\}$ $(1\le i\le m)$. Above we saw that we can retrieve $x_1x_2\cdots x_m$ from $y_{m+1}\cdots y_{2m}$, by Lemma 2.2 and Lemma 2.3. However, this contradicts the acceptance by M of the cut and paste computation based on $z(m)$ and $z'(m)$, because that entails the retrieval of $\chi(m)\neq x_1x_2\cdots x_m$ from $y_{m+1}\cdots y_{2m}$ by (2.5), Lemma 2.2 and Lemma 2.3.

This description of x requires no more than

$$O(\log n)+\frac{12m\,(2\log C+\log|M|)}{C^2}+n-\frac{m}{2C}\le n-\frac{m}{4C}$$

bits, for large enough C and n. However, this contradicts the incompressibility of x since $K(x)\ge n$ and $m\ge\sqrt{n}\,/C$.

Case 2 (Not jammed). We now fix a particular value m as determined by M's computation on z.

(a) By assumption (with $n'=n$), we have that $\le n/2$ polls occur on $[0,n/4)$ and $\ge n/2$ polls occur on $[n/4,\infty)$.

(b) Since $T(n)\notin\Omega(n^2)$, we have $h_2(t_i)\in[0,n/4)$, for all i $(1\le i\le\sqrt{n}\,)$.

Let $l(t)$ and $r(t)$ be the number of polls for (x_i,y_i)'s, with $h_2(t_i)\in[0,n/4)$ and $h_2(t_i)\in[n/4,\infty)$ $(1\le t_i\le t)$, respectively. By (a) and (b) there is an integer m such that $l(t)>r(t)$, for $1\le t<t_m$,

$l(t_m)=r(t_m)$ and $\sqrt{n}/C \le m \le n$. This m is the *break even* length where the number of polls left and right of position $n/4$ on the tape is equal for the first time. Let $z(m)$ be the m-length prefix of z. By (2.5), if $z \in L_q$ then $z(m) \in L_q$. Assume, by way of contradiction, that in the accepting computation of M on $z(m)$ at most $2m/C$ polls occur with h_2 on any particular (m/C)-length tape segment $R=[a, a+m/C)$.

Claim 1. As a consequence of this definition of m and (1) and (2), it follows that $r(t_m)=|y_1 \cdots y_m|=m/2$ for input prefix

$$z(m)=(x_1,y_1)\cdots(x_m,y_m).$$

Since each prefix of z satisfies (2.5), we can retrieve $x_1 \cdots x_{m/4}$ from prefix $y_1 \cdots y_m$ of $\hat{x}_1 y_1 \cdots \hat{x}_m y_m$ by Lemma 2.4.

Claim 2. By definition, all y_i's in $y_1 \cdots y_m$, which are different from ε, are polled on $[n/4,\infty)$. Since $l(t_{m/4})>r(t_{m/4})$, at most $m/8$ of the x_i's in $x_1 \cdots x_{m/4}$ are polled on $[n/4,\infty)$.

In the computation on the m-length prefix $z(m)$ of z, choose the point p with the shortest c.s. in $[n/4-m/C, n/4)$. This c.s. is shorter than m/C^2; otherwise, the running time $T(m)>m^2/C^3$, which is a contradiction.

Using the description of:

- this discussion (including the text of the program to retrieve x below) and simulator M in $O(1)$ bits,
- the values of n, m, and the position of p in $O(\log n)$ bits,
- the c.s. at p in $\le (6m/C^2)(2\log C+\log|M|)+O(1)$ bits (by Remark 1.2),
- the string u of concatenated bits of $x_1 \cdots x_{m/4}$, polled with h_2 on $[p,\infty)$ in $\le m/8+2m/C+O(\log n)$ bits (by Section 1.4), that is, $\le 2m/C$ bits polled on $[p,n/4)$ by assumption and $\le m/8$ bits polled on $[n/4,\infty)$ by Claim 2,
- the string $x_{(m/4)+1} \cdots x_n$ in $n-m/4+O(\log n)$ bits (by Section 1.4),

we can construct a program to check if a string $x' \in \{0,1\}^*$ equals x. Check $|x'|=n$ and $x'_{(m/4)+1} \cdots x'_n = x_{(m/4)+1} \cdots x_n$. Let u' be the result of deleting the bits in x' in the same positions as the ones used to obtain u from x. These positions are determined by the crossing sequence at p. Check $u'=u$. If the test is negative then $x' \neq x$, else x' can only differ from x on positions where $x_1 \cdots x_{m/4}$'s bits are polled with h_2 on $[0,p)$. Run M on $z'(m)$, that is, the input constructed according to (1), (2), using the m-length prefix $x'_1 x'_2 \cdots x'_m$ of a candidate x'. Whenever h_2 crosses p we interrupt M and check if the current *ID* in the computation is consistent with the corresponding *ID* in the c.s. at p.

By construction everything matches up to the end of processing input $z'(m)$, and M accepts, if $x'=x$. Assume that $x' \neq x$ matches the description as well. Therefore, $x'_1 x'_2 \cdots x'_{m/4} \neq x_1 x_2 \cdots x_{m/4}$ and $x'_i = x_i$ for all i $(m/4+1 \le i \le n)$. Let the input $z'(m)$, based on $x'_1 x'_2 \cdots x'_m$ and constructed according to (1),(2), be

$$z'(m)=(x'_1,y'_1)(x'_2,y'_2)\cdots(x'_m,y'_m).$$

Let the input based on $x_1 x_2 \cdots x_m$, constructed according to (1), (2), be

$$z(m) = (x_1,y_1)(x_2,y_2)\cdots(x_m,y_m)\,.$$

By assumption, x' and x differ only on the first $m/4$ bits, and then only on the bits that are polled left of p. Let the final accepting position of h_2 for M's computation on $z(m)$ be right of p. (If it is left of p interchange z and z' below.) Cut and paste the computations on $z(m)$ and $z'(m)$ such that M runs on input $z'(m)$ with h_2 left of position p, and M runs on input $z(m)$ with h_2 right of position p. Let $\zeta(m)$ be the input composed in this way from $z'(m)$ and $z(m)$. By construction, the computation on $\zeta(m)$ is also an accepting computation of M. Consequently, $\zeta(m)$ satisfies (2.5). Denote

$$\zeta(m) = (a_1,b_1)(a_2,b_2)\cdots(a_m,b_m)$$

with (a_i,b_i) is either (x_i,y_i) or (x'_i,y'_i) $(1\le i\le m)$. By construction $a_1\cdots a_m = x'_1\cdots x'_m$. We now consider the concatenated sequences of "popped" symbols (the second coordinates of the symbol pairs) of $\zeta(m)$ and $z(m)$, and show that these sequences are equal. I.e., we prove (2.8) below, and this suffices to derive a contradiction. So let us proceed. For the accepting computations under consideration, since $p<n/4$, all second coordinates (the popped part) of symbols polled left of p equal ε. The symbols with second coordinate unequal ε are by definition precisely the symbols polled right of $n/4$, and therefore also polled right of p. Therefore, the concatenation of the second coordinates of all polled symbols equals the concatenation of the second coordinates of just the symbols polled right of $n/4$. Because both $z(m)$ and $\zeta(m)$ match the description above, the ith symbol of $z(m)$ is polled left of position p if and only if the ith symbol of $\zeta(m)$ is polled left of position p, for all i, $1\le i\le m$. So all these symbols have second coordinates ε. The symbols polled right of p in the computation on $\zeta(m)$ equal the corresponding symbols of $z(m)$ by construction. Namely, we have constructed $\zeta(m)$ from $z'(m)$ and $z(m)$, by $(a_i,b_i)=(x'_i,y'_i)$ if (a_i,b_i) is polled left of p, and $(a_i,b_i)=(x_i,y_i)$ if (a_i,b_i) is polled right of p. The symbols polled right of p, include all second coordinates $\neq\varepsilon$. Therefore, the concatenation of the second coordinates of $\zeta(m)$ equals the concatenation of the second coordinates of $z(m)$:

$$b_1b_2\cdots b_m = y_1y_2\cdots y_m \tag{2.8}$$

I.e., everything popped right of p happens to be everything popped at all in both the computations of $\zeta(m)$ and $z(m)$, and, moreover, the popped sequences are equal as well. Because $x_1x_2\cdots x_{m/4}$ is retrieved from $y_1\cdots y_m$ by Claim 1, we retrieve $x_1x_2\cdots x_{m/4}$ from $b_1b_2\cdots b_m$ as well, by (2.8). But since $\zeta(m)$ is accepted by M and thus satisfies (2.5), we now have $a_1\cdots a_{m/4} = x_1\cdots x_{m/4}$. This implies that $\zeta(m)=z(m)$, since

- they coincide on the symbols polled *right* of p by definition of $\zeta(m)$,
- they coincide on the symbols polled *left* of p on the first coordinate because $x_1\cdots x_{m/4}=a_1\cdots a_{m/4}$ as we have proved, and $x_{m/4+1}..x_m=a_{m/4+1}\cdots a_m$ by description, and on the second coordinates since all of these are ε by definition of $z(m), z'(m), \zeta(m)$.

But if $\zeta(m)=z(m)$, then $x'=x$, which is a contradiction.

The description of x requires no more than

$$n-\frac{m}{8}+\frac{6m\,(2\log C+\log|M|)}{C^2}+\frac{2m}{C}+O(\log n)\le n-\frac{m}{16}$$

bits, for large enough C and n. However, this contradicts the incompressibility of x since $K(x) \geq n$ and $m \geq \sqrt{n}/C$.

Since $m \geq \sqrt{n}/C$, Cases 1 and 2 complete the proof of $T(n) \in \Omega(n^2)$. •

3. Lower Bounds for Nondeterministic Simulation

3.1. Two Pushdown Stores Versus One Tape: Nondeterministic Case

In this Section , 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

$$L = \{ x_1@x_0@x_2@x_0 \cdots @x_t@x_0\#x_1x_2 \cdots x_t\# : x_i \in \{0,1\}^* \text{ for } i=0, \ldots, t \}$$

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 .*

Proof. (I). Assume, by way of contradiction, that an 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 it can be assumed that 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 $(K(x) \geq n)$. Partition x as $x_0x_1 \cdots x_k$, where each x_i is of length $n/(k+1)$ and

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

Let

$$y = x_1@x_0@x_2@x_0 \cdots @x_k@x_0\#x_1x_2 \cdots x_k\# \tag{3.1}$$

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 $\Omega(n^{1.5}/\sqrt{\log n})$.

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 $\Omega(n^{1.5}/\sqrt{\log n})$, which is a 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 the following two cases.

Case 1 (jammed). Assume that 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 such that $|R_l| = |R| = |R_r|$. 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, where

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

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, which is a contradiction.

We can reconstruct the contents of the storage tape at time $t_\#$ by the Jamming Lemma. Using the description of:

- this discussion (including the text of the program below) and of simulator M in $O(1)$ bits,
- the values of n and k and the positions of l and r in $O(\log n)$ bits,
- 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 1.1),
- two c.s.'s that require at most $12d(\log |M| + \log(n/d)) + O(1)$ bits (by Remark 1.2),
- 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 Section 1.4),

we can construct a program to check if a string $z \in \{0,1\}^*$ equals x. For each z such that $|z| = |x|$, divide $z = z_0 z_1 \cdots z_k$ into the same number of equal length substrings as x. Check if $z_0 = x_0$. Check the x_i's which are given literally against the corresponding z_i's. Assume that $\#z_1 \cdots z_k\#$ is the input suffix in (3.1) and continue to simulate M from $t_\#$ on, with the storage tape reconstructed 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 $z = x$.

This description of x requires no more than

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

bits for large enough c and n and some positive γ, $\gamma < 1$, by (3.2) and (3.3). This contradicts the incompressibility of x ($K(x) \geq n$). One might worry about nondeterminism here, but note that nondeterminism does not matter. We simply try all possibilities.

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* and *into* which these two pairs are mapped is at least n/c. Let R_0 be a tape segment in between and $|R_0| \geq n/c$. 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 $\Omega(n^{1.5}/\sqrt{\log n})$, which is a contradiction. We use this shortest c.s. to reconstruct x_0 below. 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, and the location of p in $O(\log n)$ bits,
- a literal description of $x_1 x_2 \cdots x_k$ in $nk/(k+1) + O(\log n)$ bits (by Section 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 1.2),

we can construct a program to check if a string z equals x by running M as follows. Check if $|z| \neq n$. If $|z| = n$ then divide z in $k+1$ equal length substrings, $z = z_0 z_1 \cdots z_k$. Check if $z_i = x_i$ for all $i > 0$. If not, then $z \neq x$; otherwise, arrange the z_i's (including z_0) in their correct positions on the input tape. Note that the nondeterminism of M does not matter; the program can try all possibilities to find 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 and then use the next ID to continue the simulation. By construction, for $z=x$ the simulation ends with everything matching all the way. Suppose there is a $z' \neq x$ which passes all tests as well. Then we can cut and paste the two computations of M using candidates z and z' at point p, obtaining an accepting computation as well. However, z and z' must differ in the block corresponding to x_0, say z_0 and z'_0. By assumption, therefore, M accepts a string containing both $x_i@z_0$ and $x_j@z'_0$ with z_0 and z'_0 in the positions reserved for x_0's: contradiction.

The description of x requires no more than

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

$$\leq n - \gamma\sqrt{(n \log n)} \quad \text{(by (3.2), (3.3))}$$

bits, for some positive γ ($\gamma \geq 1/2 - 12/c$) and contradicts the incompressibility of x ($K(x) \geq n$), for large enough c and n. Case 1 and Case 2 prove (I).

(II). 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 both in stack1 and in stack2, put x_2 in stack2, put the next x_0 both in stack1 and in 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 these changes do not invalidate the lower bound proof in (I), the proof of the Theorem follows. •

Combined with Theorem A (below) recently proved in [10], 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 Section 2.2. Here we obtain an $\Omega(n^{4/3}/\log^{2/3} n)$ lower bound for the nondeterministic case. By [10] $O(n^{1.5\sqrt{\log n}})$ is an upper bound, cf. Theorem A in Section 3.1.

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

Proof Idea. At first glance, one might think the language L in Section 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 ($0 \leq i \leq k$) 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

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

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 $x_1x_0 \cdots x_kx_0$ 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, with $1 \le i \le k$. 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 Section 3.1. We omit the details. Intuitively, the lower bound is obtained by maximizing $t(n)$ (= lower bound on the running time $T(n)$ of the simulator) under the constraints:

- $t(n) \in O(n^{1.5\sqrt{\log n}})$ (by Theorem A),
- $k|x_0|^2 \in O(n)$ (the length of the padding must be less than length of input),
- $t(n) \in O(kn)$ (not more than $k/2$ pairs $x_i@x_0@$ can be mapped onto tape segments of length n/c for fixed constant c), and
- $(t(n)/n)\log(n^2/t(n)) \in O(|x_0|)$ (crossing sequences of length of order $t(n)/n$ can be described in at most $O(|x_0|)$ bits). •

3.3. Two Tapes Versus One Tape: Nondeterministic Case

Unlike the results presented above which are independent of [11], Theorem 3.3 is based on and presupposes the approach of [11].

For the nondeterministic off-line one-way input case of one tape versus two tapes, Maass [11] obtained an

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

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 [4, 11] independently construct two similar ingenious languages (the language of [4] is less complete).

In [11] a general language L_I was introduced, but only a simple subset, $\hat{L}$, of it was used. This language $\hat{L}$ consists of all words of the form uvw such that

- the u_i's are binary strings, and
- suffix vw is obtained from $uu=u_1 \cdots u_k u_{k+1} \cdots u_{2k}$, with $u_{k+i}=u_i$, by inserting u_i in between u_{2i-1} and u_{2i} $(1 \le i \le k)$.

The length of each u_i may be different. We can also define a delimited version L^* of $\hat{L}$ where every u_i in $\hat{L}$ is replaced by $*u_i*$ of a 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 $\{0,1\}^*\{2\}\{0,1\}^*$, $n \ge 0$. 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 [10] it is proved that this is not the case. (Theorems B and C below. Theorem C is really a corollary of Theorem A above.)

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 Section, trying to meet the upper bound of Theorem B, we improve the lower bound of [11] to

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

Since the following theorem is based on the approach in that paper, we assume the reader is familiar with the details of [11] and only point out where and how the improvement is obtained*.

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 [11] Maass proved an important combinatorial lemma (Theorem 3.1 in that reference) which is generalized as follows.

Lemma 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.*

---

* Zvi Galil, Ravi Kannan and Endre Szemeredi have obtained a still better $\Omega(n^2/\log^{(k)} n)$ lower bound (for all k) on the time to simulate 2 tapes by 1 nondeterministic off-line tape with one-way input [5]. ($\log^{(k)}$=loglog $\cdots$ log is the k times iterated logarithm.)

The proof of this lemma is a simple reworking of the proof in [11]. A $k/\sqrt{\log k}$ upper bound corresponding to the lower bound in this lemma is contained in [10].

It can be shown that any sequence S in L^* satisfies the requirements in Lemma 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 [11] 'deserts' on the storage tape.

Lemma 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 $c\geq 2$ is the constant in Lemma 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. Divide the whole storage tape into tape segments of length $n/(c^{13}\log n)$. By the Jamming Lemma, no tape segment can have more than $n/(c^{11}\log n)$ blocks mapped into it. 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 [11] for each desert D_i in Lemma 3.2. Instead of using Theorem 3.1 of [11] we use Lemma 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 3.1 can be applied. Now M spends $\Omega(n^2/(\log^2 n \log\log n))$ time on each tape segment D_i. There are $\Omega(\log n)$ such tape segments, and summing the amounts of time M spends on each of them yields the $\Omega(n^2/(\log n \log\log n))$ lower bound. •

Acknowledgements

Chanderjit Bajaj, Juris Hartmanis, Evangelos Kranakis, Luc Longpre, Joel Seiferas, Yaacov Yesha and the referee supplied helpful comments. We are grateful to Zvi Galil for carefully checking the manuscript, and pointing out a gap in the earlier version of the proof of Theorem 2.2.

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.* **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* **60**, pp. 1-11 (1984).
4. Freivalds, R., "Probabilistic machines can use less running time," *Information Processing* **77**, pp. 839-842 (1977).
5. Galil, Z., R. Kannan, and E. Szemeredi, "On nontrivial separators for k-page graphs and simulations by nondeterministic one-tape Turing machines," in *Proceedings 18th Annual ACM Symposium on Theory of Computing* (1986).

6. Hartmanis, J. and R.E. Stearns, "On the computational complexity of algorithms," *Trans. Amer. Math. Soc.* **117**, pp. 285-306 (1969).

7. Hennie, F.C. and R.E. Stearns, "Two tape simulation of multitape Turing machines," *J. Assoc. Comp. Mach.* **4**, pp. 533-546 (1966).

8. Hopcroft, J.E. and J.D. Ullman, *Formal Languages and their Relations to Automata*, Addison-Wesley (1969).

9. Kolmogorov, A.N., "Three approaches to the quantitative definition of information," *Problems in Information Transmission* **1**(1), pp. 1-7 (1965).

10. Li, M., "Simulating two pushdowns by one tape in O(n**1.5 (log n)**0.5) time," pp. 56-64 in *Proceedings 26th Annual IEEE Symposium on the Foundations of Computer Science* (1985).

11. Maass, W., "Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines," *Trans. Amer. Math. Soc.* **292**, pp. 675-693, (Preliminary Version "Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines," pp 401-408 in Proceedings 16th ACM Symposium on Theory of Computing, 1984.) (1985).

12. Morkoc, H. and P.M. Solomon, "The HEMT: a superfast transistor," *IEEE Spectrum*, pp. 28 - 35 (February, 1984).

13. 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* **23**, pp. 108-126 (1981).

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

15. Rabin, M.O., "Real-time computation," *Israel J. of Mathematics* **1**, pp. 203-211 (1963).

16. Vitányi, P.M.B., "On the simulation of many storage heads by one," *Theoretical Computer Science* **34**, pp. 157-168 (1984).

17. 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* **21**, pp. 87-91 (1985).

18. 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* **21**, pp. 147-152 (1985).