Theoretical Computer Science 34 (1984) 157–168
North-Holland

ON THE SIMULATION OF MANY STORAGE HEADS BY ONE*

Paul M.B. VITÁNYI
Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

Abstract. Each multitape Turing machine, of which the storage heads scan $O(\log n)$ distinct squares in each interval of n steps, for all $n \geq 1$, can be real-time simulated by an oblivious one-head tape unit. To simulate but the normal pushdown store, on-line by an oblivious one-head tape unit, requires $\Omega(n\sqrt{n})$ time. (This improves the known $\Omega(n \log n)$ lower bound for the on-line simulation of multitape Turing machines by oblivious one-tape Turing machines.)

Key words. Multitape Turing machines, on-line simulation by single head tape units, upper bounds, lower bounds, real-time simulation by oblivious one-head tape unit, uniform logarithmic space, multicounter machines, augmented counter machines, pushdown stores.

1. Introduction

It is generally the case that additional access pointers in storage enhance computing power. In real-time $(k+1)$-tape Turing machines are more powerful than k-tape Turing machines [1]. Analogous results hold with all heads placed on the same tape [7, 11], head-to-head jumps added [7], and for multihead finite automata with and without head-to-head jumps [5, 10]. Recently it was shown that k-tape Turing machines require nonlinear time to on-line simulate $(k+1)$-tape Turing machines [6]. With respect to upper bounds there are essentially two facts known. Each multitape machine can be on-line simulated by a one-head tape unit in square time [4], and by a two-tape Turing machine in time[1] $O(n \log n)$ [3]. Both of these simulations can be made oblivious (the first one in the obvious way and the second one as in [8]), retaining the same simulation time. In [8] it was furthermore shown that each oblivious multitape Turing machine, simulating a single pushdown store, requires $\Omega(n \log n)$ time. Thus, for on-line simulation of multitape Turing machines by one-head tape units the fastest simulation time is somewhere in between a nonlinear lower bound and a square upper bound, while for on-line simulation by

* This work is registered at the Centre for Mathematics and Computer Science.

[1] We use the mnemonic 'order of magnitude' notations as follows:
$f(n) \in O(g(n))$ if there are positive constants c and n_0 such that $|f(n)| \leq cg(n)$ for all $n \geq n_0$;
$f(n) \in \Omega(g(n))$ if there is a positive constant c such that $f(n) \geq cg(n)$ for infinitely many n;
$f(n) \in \Theta(g(n))$ if $f(n) \in O(g(n)) \cap \Omega(g(n))$;
$f(n) \in o(g(n))$ if $f(n) \in O(g(n)) - \Theta(g(n))$.
Note that there are subtle differences possible about how to define the above notions. Contrary to the customary usage, for our purposes the precise definitions *do* matter, at least with respect to the natural extension to two-variable functions in Section 3.

oblivious one-head tape units the lower bound is $n \log n$ and the upper bound n^2. We improve this situation in two ways. First, in Section 2, we show that for a restricted class of multitape Turing machines, viz. machines of which the storage heads scan $O(\log n)$ distinct squares in each interval of n steps, for all $n \geq 1$, each member can be real-time simulated by an oblivious one-head tape unit belonging to that class. Second, in Section 3, it is demonstrated that each oblivious one-head tape unit, on-line simulating a single pushdown store, requires $\Omega(n\sqrt{n})$ time. This improves the previous best $\Omega(n \log n)$ lower bound on the time to simulate multitape Turing machines on-line by oblivious one-head tape units. In the proof we found it advantageous to define and use the mnemonic order of magnitude symbols in a particular, unusually specific, fashion for two-variable functions.

Turing machines, simulation and obliviousness

We regard machines as *transducers*, that is, as abstract storage devices connected with input- and output terminals. Thus we consider the machine as hidden in a black box, and the presented simulation results concern the input/output behaviour of black boxes and are independent of input/output conventions or whether we want to recognize or to compute. By a k-tape Turing machine we mean an abstract storage device, consisting of a finite control connected with k single-head linear storage tapes, and an input- and an output terminal. A one-tape Turing machine is the same as a one-head tape unit. The transducers effect a transduction from input strings to output strings by producing the ith output just before reading the $(i+1)$st input command. A machine A *simulates* a machine B *on-line* in time $T(n)$ if, for all $n > 0$, the input/output behaviour of B, during the first n steps, is exactly mimicked by A within the first $T(n)$ steps. That is, for each input sequence $i_1, i_2, \ldots, i_k, \ldots$, read from the input terminal, the output sequences written to the output terminal are the same for A and B, and if $t_1 \leq t_2 \leq \cdots \leq t_k \leq \cdots$ are the steps at which B reads or writes a symbol, from or to the terminals, then there are corresponding steps $t'_1 \leq t'_2 \leq \cdots \leq t'_k \leq \cdots$ at which A reads or writes the same symbols and $t'_i \leq T(t_i)$, for all $i \geq 1$. In the sequel we write *simulation* for *on-line simulation.* Simulation in time $T(n) = n$ is called *real-time* simulation; simulation in time $T(n) \in O(n)$ is called *linear time* simulation. A Turing machine is *oblivious* if the movements of the storage tape heads are fixed functions of time, independent of the particular inputs to the machines (see e.g. [8]). There are many reasons why one may want to restrict attention to oblivious computations. For instance, oblivious Turing machine computations translate efficiently to combinational logic networks, while ordinary Turing machine computations do not. We mention two less often cited motives, of a more heuristic nature, for focussing attention on oblivious computations, of which the second one is pure conjecture. Suppose we can simulate some abstract storage device S in time $T(n)$ by an oblivious one-head tape unit M. Then we can also simulate k copies of S, say $S_1, S_2, \ldots, S_k$, interacting through a common finite control, by dividing M's tape into k tracks, modifying M's finite

control, and letting the head on each track do the same job as it formerly did on the total tape. Thus, the resulting oblivious one-head tape unit M', with modified finite control and expanded tape alphabet, uses the same time and space as did M.

Lemma 1.1. *If we can simulate a pushdown store by an oblivious one-head tape unit in time $T(n)$, then we can simulate each multitape Turing machine by an oblivious one-head tape unit in time $T(n)$, using just the same space.*

Proof. Replace each tape of the multitape Turing machine by two pushdown stores and apply the preceding argument to multipushdown store machines. □

Conjecture 1.2. *If we can simulate each multitape Turing machine by a one-head tape unit in time $T(n)$, then we can also simulate each multitape Turing machine by an oblivious one-head tape unit in time $T(n)$.*

The intuitive background for this conjecture is as follows. If we can accommodate the multitude of headmovements of arbitrary many-headed multitape Turing machines by the limited number of trajectories available to the single head of a one-head tape unit, then a single trajectory has to serve such a general multitude of input streams that there is no reason to suppose that the machine needs to take advantage of particular input streams. Thus, since the essential generality of the task at hand is captured in a strategy for one trajectory, the simulating machine ought to be able to follow the strategy and that trajectory for *all* input streams.

2. Uniform space and fast simulation of many heads by a single oblivious one

For on-line computations (viz. the transducer type of computations) it is, perhaps, unreasonable that the workspace accessed in any length input interval may be arbitrarily large, for unbounded storage complexity, if the machine has been computing long enough previously. For example, if a real-time Turing machine M has storage complexity $O(\log n)$, then in the interval of steps, for the processing of the 2^nth input symbol, the machine M can access $\Theta(n)$ distinct storage squares. In the interval of steps, for the processing of the 2^{2^n}th input symbol, the machine M can access $\Theta(2^n)$ distinct squares, and so on. In certain aspects, this seems not what we would expect of a complexity measure for on-line computation. Considering on-line computation the theoretical counterpart of, e.g., interactive computer use, it would entail that, if we forget to log out of a terminal over the week end, then the computer may use far more memory for identical commands than if we did not forget to log out. Thus, we propose a space complexity measure, independent from the *origin* of the time scale, and only depending on the sizes of the *intervals* of steps.

Definition 2.1. Let M be a multitape Turing machine, and let for any unbounded input sequence ω the interval of steps by M, executed in processing ω, from the $(m+1)$st through $(m+i)$th step, be denoted by $I^{\omega}_{m,i}$, for all $m \geq 0$ and $i \geq 1$. For any unbounded input sequence ω, and any $m \geq 0$ and $i > 0$, we denote by $U^{\omega}_{m,i}$ the number of distinct squares, summed over all of M's storage tapes, which are visited by some storage tape head during $I^{\omega}_{m,i}$. The *uniform space complexity* $U(n)$ of M is defined as

$$U(n) = \sup\{U^{\omega}_{m,n} \mid m \geq 0 \ \& \ \omega \text{ an unbounded input sequence}\}.$$

Thus, finite automata correspond to Turing machines with uniform space complexity $O(1)$. A little reflection learns us the relation between uniform space complexity, on the one hand, and ordinary space complexity and time complexity on the other. If the (ordinary) space complexity of some Turing machine is $S(n)$ and if the time complexity is $T(n)$, then its uniform space complexity $U(n) \in \Omega(S(T^{-1}(n))) \cap O(n)$. Recall that finite automata have the exceptional property that a storage facility, consisting of a collection of k finite automata, interacting through a common finite control, can be replaced by a single finite automaton without slowing down the computation. Below we show that additional tapes, or nonobliviousness, likewise do not increase the power of an (oblivious) one-head tape unit in uniform logarithmic space.

Lemma 2.2. *Each multitape Turing machine of uniform space complexity* $O(\log n)$ *can be real-time simulated by an oblivious one-head tape unit of also uniform space complexity* $O(\log n)$.

Proof. The proof uses a complicated tape manipulation technique developed in [12], to simulate multicounter machines by oblivious one-head tape units in real-time. Without going into details, an oblivious one-head tape unit M can be constructed such that, for each $i \geq 1$, the pair of squares, or rather square contents or *cells*, originally in positions i, $i+1$, is scanned at least once in each time interval of c^i steps, for some small constant c, say c is about 3. Moreover, the head recognizes such pairs when they are scanned (knows they are cells i, $i+1$ for some $i \geq 1$) and has always cells 1, 2 under scan. The tape unit M works by, in each step, interchanging cells residing on the currently simultaneously scanned tapesquares. (M's fat head scans a few adjacent squares simultaneously.) In this process, the identity of the underlying squares is not important; the identity of the cells (index i above), however, is fixed wherever they end up. The oblivious one-head tape unit M has uniform space complexity $\Theta(\log n)$. By Lemma 1.1 we only have to show that any pushdown store P of uniform space complexity $O(\log n)$ can be real time simulated by the described oblivious one-head tape unit. So, let P be a pushdown store which does not change its stack height by more than $O(\log i)$ elements in each interval of steps $I^{\omega}_{m,i}$, for all $m \geq 0$, $i \geq 2$ and any ω. In the simulating M, each cell (square contents)

can contain an ordered segment of P's stack consisting of 0, d or $2d$ elements, and the first cell can contain an initial segment of P's stack of in between 0 and $2d$ elements. Each cell i ($i > 1$) strives for an occupancy of stack elements as follows. If it contains $2d$ elements when cells i, $i+1$ are scanned, then the last d elements are shifted to cell $i+1$. If it contains 0 elements when cells i, $i+1$ are scanned, and cell $i+1$ contains d or $2d$ elements, then the first d elements are shifted from cell $i+1$ to cell i. Cell 1, being distinguished, shifts d elements out if it contains $2d$ elements, and shifts d elements in if it contains $d-1$ (or less) elements, to and from cell 2. According to the current input, elements are added/deleted from the segment in cell 1 in each step. Thus, a segment of d stack elements can be shifted from the 1st cell to the ith cell, or vice versa, in $\sum_{j=1}^{i-1} c^j \leq c^i$ ($c \geq 2$) steps. Thus, for all $i \geq 1$, in c^i steps id elements can be pushed or popped. Starting with an empty stack, it is tedeous, but not difficult, to prove that at all times $t \geq 0$, for any input,

(i) no cell contains more than $2d$ stack elements;

(ii) if any cell contains stack elements, then cell 1 contains stack elements,

provided the stack height does not change more than id elements in I^{ω}_{m,c^i}, for all m, i, ω. Choosing d appropriately, which is possible since the stack height varies $O(\log i)$ elements in each interval $I^{\omega}_{m,i}$, for all m, i, ω, (i) and (ii) show that the arrangement can real-time implement a uniform $O(\log n)$ space pushdown store. □

The next question is which computations, or problems, are in uniform logarithmic space. In [12] it is shown that each multicounter computation is of this space complexity. Uniform log space is, however, more extensive. Recall that multicounter machines consist of a set of counters numbered say, 1,2, ..., k, which can execute one-step arithmetic/boolean instructions as "add [subtract] 1 from counter i" and "test counter i for 0", $1 \leq i \leq k$. Several other one-step instructions can be synthesized, by using concealed auxiliary counters, such as tests for equality amongst counters (by maintaining all the differences on extra counters). Instructions for which it is known [2] that they cannot be so synthesized as one-step instructions are "set counter i to 0" or "set counter i to the value of counter j". Call multicounter machines, with arbitrary integer initial counter contents allowed, and with those one-step instructions added, *augmented counter machines* [ACMs]. The following lemma can be proved [13].

Lemma 2.3. *Each augmented counter machine can be real-time simulated by a uniform log space oblivious one-head tape unit.*

None the less, uniform log space computations are *not very* powerful if we impose time restrictions.

Lemma 2.4. *There are (ordinary) log space/real-time Turing machines such that the fastest uniform log space Turing machines simulating them use exponential time.*

Proof. Take as an example the on-line recognition of[2]

$$L = \{@w_1@w_2@ \cdots @w_{2^i}@ \cdots | \, w_j \text{ is a palindrome in } \{0,1\}^*, j \geq 1,$$

$$\& \; \# w_{2^i} = \# w_{2^i+1} = \cdots = \# w_{2^{i+1}-1} = 2(i+1), \; i \geq 0\}.$$

*Recognition in real-time/*log *n space.* On one track of its single storage tape the recognizing machine maintains a binary count of the number of received @'s. During the update of the @ count, it can write the first half of the, simultaneously read, current word w_{current} on a second track and, while proceeding back to the origin, compare it with the second half. Due to the particular choice of the lengths of the consecutive palindromes in a word of L, the recognizing machine can be oblivious. Thus L is recognized by a real-time oblivious one-head tape unit in logarithmic space.

Recognition in uniform log *n space.* To on-line check whether w_{2^i} is a palindrome, a candidate machine must access $\Omega(i)$ storage tape squares. By definition it takes $\Omega(2^i)$ steps to do so. □

Obviously, any real-time multitape Turing machine computation can be simulated on a uniform log n space Turing machine in exponential time. Thus, by Lemmas 1.1, 2.2–2.4 we have the following theorem.

Theorem 2.5. (i) *In real-time, multicounter machines are less powerful than uniform* log *n space Turing machines.*

(ii) *In real-time, uniform* log *n space multitape Turing machines are equally powerful as uniform* log *n space oblivious one-head tape units.*

(iii) *There are* log *n space/real-time one-tape Turing machines which cannot be simulated by uniform* log *n space multitape Turing machines in less than exponential time.*

What is the most extensive uniform space complexity class for which nonobliviousness or extra heads do not increase the power of the device under the real-time restriction? We do not know yet. However, we can give an upper bound on the uniform space complexity allowing *linear* time simulation, of multitape Turing machines, by oblivious one-head tape units. Viz., each real-time multitape Turing machine, which can be linear time simulated by an oblivious multitape Turing machine, has uniform space complexity $o(n/\log n)$. This can easily been proven similar to the overlap argument in [8] (used to prove that an oblivious Turing machine, simulating a single pushdown store, needs $\Omega(n \log n)$ time). Recall that the *overlap*, in an input segment of length i, is the maximum number of distinct storage locations which are visited both during the first part of the input segment and during the remainder of the input segment, for all partitions of the input segment in two consecutive pieces. To simulate a real-time Turing machine with uniform

[2] Below we use $\# x$ to denote the *length* of a string x and $\# X$ to denote the *cardinality* of a set X.

space complexity $U(n)$, in linear time by an oblivious Turing machine, the overlap in each input segment of length i needs to be $\Omega(U(i))$, for all $i>0$. Setting $n=2^{\log n}$ and summing all independent overlaps as in [8], the resulting total must be majorized by the simulation time $T(n)$ of the oblivious simulator. The argument leads in fact to the more general trade-off:

$$\sum_{i=1}^{\log n} 2^{-i}U(2^i) \in O(T(n)/n),$$

which for $T(n) \in O(n)$ yields

$$\sum_{i=1}^{\log n} 2^{-i}U(2^i) < c$$

for some constant $c \geqq 0$, from which the above statement follows.

3. Improved lower bound on the time to simulate multitape Turing machines by oblivious one-head tape units

In view of Lemma 1.1, any lower bound on the time to simulate multitape Turing machines, by oblivious one-head tape units, also holds for the simulation of a single pushdown store by the latter (and obviously, vice versa). The following theorem improves the known lower bounds. In the proof we make extensive use of *crossing sequences*. For a one-head tape unit we assume that, when it makes a move, it first overprints the symbol scanned and changes state, then moves the head. Thus, for any pair of adjacent tape squares we can list the sequence of states in which the unit *crosses* from one to another. The first crossing must always be from left to right; after that, crossings alternate in direction. The sequence of states so related to an intersquare boundary, or square, is called a crossing sequence. The concept seems to originate from [9].

Theorem 3.1. *Any oblivious one-head tape unit simulating the typical pushdown store requires* $\Omega(n\sqrt{n})$ *time.*

Proof. Let M be the fastest oblivious one-head tape unit for simulating a typical pushdown store in, say, time $T(n)$. It is known that $T(n) \in \Omega(n \log n) \cap O(n^2)$. Let $I_{m,i}$ denote the interval of steps by machine M to process the $(m+1)$st through $(m+i)$th *input command.* (Do not confuse these intervals with the I-intervals of the previous section. The subscripts refer to the sequence of input commands, instead of the sequence of steps, and since M is oblivious a superscript referring to particular input sequences is unnecessary, and therefore suppressed. For simplicity we assume that, in an oblivious machine, the steps at which it reads or writes a symbol are the same for all input streams. However, it is not necessary to assume this subtility in order to derive the desired results.) Let M have n_1 states and n_2 storage tape symbols.

Assume that M's tape is one-way infinite. Consider the set of all input streams of length n. Let $[0, c_Q n)$ be some initial $c_Q n$-length[3] tape segment, let $[c_S n, \infty)$ be a final tape segment, consisting of all of the tape but an initial $c_S n$-length tape segment, and let $[c_Q n, c_S n)$ be the segment in between, for some constants $c_S > c_Q > 0$, yet to be chosen. See the diagram below.

$[0, c_Q n)$	$[c_Q n, c_S n)$	$[c_S n, \infty)$

The idea is to show that there must be an inputsegment $I_{m,\sqrt{m}}$, for which the storage head starts out on the final tape segment $[c_S n, \infty)$, and will not traverse more than $(c_S - c_Q)n$ tapesquares, thus staying out of the initial tapesegment $[0, c_Q n)$. Consequently, the information which has originally been recorded on the initial tapesegment $[0, c_Q n/2)$, must have been transported over the intervening tapesegment $[c_Q n/2, c_Q n)$, in order to be accessed during $I_{m,\sqrt{m}}$. This then entails long crossing sequences for each square in the intervening tape segment, the sum of which yields the claimed lower bound on the running time. Below we shall use three functions S, P and Q,

$$S, P, Q : \mathbb{R} \times \mathbb{N} \to \mathcal{P}(\mathbb{N}),$$

with $\mathbb{R}$ the set of positive rationals, $\mathbb{N}$ the set of natural numbers and $\mathcal{P}(\mathbb{N})$ the powerset of the set of natural numbers:

$S(c, n) \stackrel{\text{def}}{=} \{j \mid 1 \leq j \leq n$ and the tapesegment visited in $I_{j,1}$ is contained in $[0, cn)\}$,

$P(c, n) \stackrel{\text{def}}{=} \{j \mid 1 \leq j \leq n$ and the tapesegment visited in $I_{j,\sqrt{j}}$ has length at least $cj\}$,

$Q(c, n) \stackrel{\text{def}}{=} \{j \mid 1 \leq j \leq n$ and the tapesegment visited in $I_{j,\sqrt{j}}$ is contained in $[cn, \infty)\}$.

It follows straightaway from the above definitions that for

$$0 < c_Q < c_S - c_P \tag{1}$$

we have

$$Q(c_Q, n) \supseteq \{1, 2, \ldots, n\} - S(c_S, n) - P(c_P, n). \tag{2}$$

We wish to use the mnemonic devices $O(n)$, $\Omega(n)$, $\Theta(n)$ and $o(n)$ to classify the cardinalities of the above two-variable functions. We therefore define for $\# X(c, n)$, with $X = S, P$ or Q,

$$\# X(c, n) \in O(n) \quad \text{if there is a } \Delta > 0 \text{ such that, for each } c > 0, \text{ there is an } n_c > 0 \text{ so that } \# X(c, n) \leq \Delta n, \text{ for all } n \geq n_c,$$

[3] To avoid a cumbersome notation we tacitly assume, while indicating a value in a discrete universe of integers by a noninteger value, that the *ceiling* of the noninteger value is intended. Thus, by $[x, y]$ we mean the tapesegment consisting of the $\lceil x \rceil$th through $\lceil y \rceil$th tape square. Likewise, by $\sqrt{x}$ we mean $\lceil \sqrt{x} \rceil$.

$\# X(c, n) \in \Omega(n)$ if there is a $\delta > 0$ such that, for each $c > 0$, $\# X(c, n) \geq \delta n$, for infinitely many n,

$\# X(c, n) \in \Theta(n)$ if $\# X(c, n) \in O(n) \cap \Omega(n)$,

$\# X(c, n) \in o(n)$ if $\# X(c, n) \in O(n) - \Theta(n)$.

Claim 1. *If* $T(n) \in o(n^2)$, *then* $\# S(c, n) \in o(n)$.

Proof. Assume $T(n) \in o(n^2)$ and $\# S(c, n) \in \Theta(n)$. (By definition $\# S(c, n) \in O(n)$.) Thus, there is a $\delta > 0$ such that, for each $c > 0$, we have $\# S(c, n) \geq \delta n$ for infinitely many n. In the remainder of the proof of Claim 1, when we use n, we tacitly assume that, for the concerned constant c, we only consider those values for n, of which there are infinitely many by contradictory assumption, such that $\# S(c, n) \geq \delta n$. Thus, for a set of $2^{\delta n}$ distinct input sequences, of length n, the distinguishing input commands are all received and processed while the head is on the $[0, cn)$ tape segment, and the input commands received with the head on the final $[cn, \infty)$ tape segment are identical. To distinguish between all of these input command sequences afterwards, we need to utilize at least xn tapesquares of storage with

$$n_2^{xn} + n_1 \geq 2^{\delta n}. \tag{3}$$

Assume that x is minimal with respect to (3). Since δ is assumed to be independent of c, we are at this stage still free to choose c to suit our argument. So set c to $x/3$. Consider the crossing sequences associated with the squares of the tapesegment $[cn, 2cn)$ (see diagram below).

$[0, cn)$	$[cn, 2cn)$	$[2cn, \infty)$

Suppose, that some square on the stapesegment $[cn, 2cn)$ has a crossing sequence of length $o(n)$. Then, to distinguish the $2^{\delta n}$ input sequences of length n, which differ only while the head is on the tapesegment $[0, cn)$, we must have

$$2^{\delta n} \leq n_1^{o(n)} + n_2^{2cn} = n_2^{2xn/3 + O(1)}, \tag{4}$$

for n large enough. The assumed minimality of x in (3) is, for large enough n, contradicted by inequality (4), so we conclude that all squares on the tape segment $[cn, 2cn)$ have crossing sequences of length $\Omega(n)$. The time $T(n)$ must by definition majorize the sum of the lengths of the crossing sequences, so

$$T(n) \geq cn \cdot \Omega(n) \in \Omega(n^2). \tag{5}$$

Since (5) contradicts the assumed $T(n) \in o(n^2)$, we conclude that $\# S(c, n) \in o(n)$, and the claim is proven. □

Claim 2. *If* $T(n) \in o(n\sqrt{n})$, *then* $\# P(c, n) \in o(n)$.

Proof. Assume $T(n) \in o(n\sqrt{n})$ and $\# P(c, n) \in \Theta(n)$. (By definition $P(c, n) \in O(n)$.) Thus, there is a $\delta > 0$ such that, for each $c > 0$, we have $\# P(c, n) \geq \delta n$, for infinitely many n. In the sequel of the proof of this claim we assume that the used values for n are, for each particular $c > 0$, chosen such that $\# P(c, n) \geq \delta n$. For each index $j \in P(c, n)$ we have

$$T(j+\sqrt{j}) - T(j) \geq cj. \tag{6}$$

Therefore, for each such j, the average value $A_{j'}$ of the lengths $\# I_{j',1}$'s, of intervals $I_{j',1}$'s, for the set $\{j' \mid j \leq j' \leq j+\sqrt{j}\}$, is given by

$$A_{j'} = \frac{\# I_{j,\sqrt{j}}}{\sqrt{j}} \geq c\sqrt{j} \geq \frac{c\sqrt{j'}}{2}. \tag{7}$$

By assumption there are δn distinct elements in $P(c, n)$, so a fortiori also δn distinct indices j, $0 \leq j \leq n$, with an average value of $\# I_{j,1}$ of at least $c\sqrt{j}/2$, by (6) and (7). Denote the set of these indices by J_n. Then,

$$T(n) = \sum_{j=0}^{n} I_{j,1} \geq \tfrac{1}{2} \sum_{j \in J_n} c\sqrt{j} \geq \tfrac{1}{2} \sum_{j=0}^{\delta n} c\sqrt{j} \in \Omega(n\sqrt{n}). \tag{8}$$

By (8) we contradict the assumption, and consequently prove the claim. □

Claim 3. *For each $\varepsilon > 0$ there are positive constants δ and n_δ such that $\# S(\delta, n) \leq \varepsilon n$ (respectively, $\# P(\delta, n) \leq \varepsilon n$) for all $n \geq n_\delta$.*

Proof. The proof follows by Claims 1 and 2, according to the definition of $o(n)$. □

Proof of Theorem 3.1 (*continued*). Assume, by way of contradiction, that $T(n) \in o(n\sqrt{n})$. Then, choosing c_S, c_P and c_Q according to (1), by (2) it follows from Claims 1, 2 and 3 that

$$\# Q(c_Q, n) \geq n - \# S(c_S, n) - \# P(c_P, n) \geq (1-\varepsilon)n, \tag{9}$$

for arbitrary small constant $\varepsilon > 0$, depending on c_S and c_P, for n large enough. Since all indices in $Q(c_Q, n)$ are distinct, and $m(c_Q, n)$ is the largest index in $Q(c_Q, n)$, it therefore follows that

$$\forall_{\varepsilon > 0} \; \varepsilon \; \exists_{\delta > 0} \; \delta \, [m(\delta, n) \geq (1-\varepsilon)n], \tag{10}$$

by the choice of c_S, c_P and c_Q, for n large enough. Now consider, for some $\delta > 0$ for which $\varepsilon < \frac{1}{2}$ in (10), the input ensemble

$$\{push\ 0,\ push\ 1\}^{\sqrt{n}} \{skip\}^{m(\delta,n)-\sqrt{n}} \{pop\}^{\sqrt{n}}. \tag{11}$$

Since $T(n) \in o(n\sqrt{n})$, we have $T(\sqrt{n}) \in o(n^{3/4})$, and therefore, for all large enough n, the tape head never leaves the initial tapesegment $[0, \delta n/2)$ during the interval $[0, \sqrt{n}]$. Thus, for the input ensemble (11), we must pop an arbitrary sequence of 0's

and 1's of length $\sqrt{n}$, originally recorded on the initial $\delta n/2$ length tapesegment $[0, \delta n/2)$, while never leaving the final tapesegment $[\delta n, \infty)$ (see the diagram below):

$[0, \delta n/2)$	$[\delta n/2, \delta n)$	$[\delta n, \infty)$

Again using a crossing sequence argument, we obtain the contradictory $T(n) \in \Omega(n\sqrt{n})$, and hence the theorem. For suppose that any square in the tapesegment $[\delta n/2, \delta n)$ had a crossing sequence of length not exceeding $e\sqrt{n}$ during $I_{0,n}$, for a small constant $e > 0$ yet to be chosen. Then for the input ensemble (11), the number of distinguishable final contents of the tapesegment $[\delta n, \infty)$, subsequent to the processing of n-length input sequences, is bounded above by $n_1^{e\sqrt{n}}$. Yet there are $2^{\sqrt{n}}$ distinct initial sequences to be popped, without leaving the final tapesegment $[\delta n, \infty)$. Choosing the constant e such that

$$2^{\sqrt{n}} > n_1^{e\sqrt{n}}, \tag{12}$$

for n large enough, implies that not all distinct pushed 0–1 sequences of the input ensemble (11) can be distinguished in the popping phase, thus fooling the machine. Consequently we must assume that the crossing sequences of all squares of the tapesegment $[\delta n/2, \delta n)$ have length greater than $e\sqrt{n}$ for some fixed constant $e > 0$ such that inequality (12) does *not* hold. Since the time $T(n)$ majorizes the sum of the lengths of the crossing sequences, we have

$$T(n) \geq \tfrac{1}{2}\delta n \cdot e\sqrt{n} \geq \frac{\delta n\sqrt{n}}{2 \log n_1}, \tag{13}$$

for some $\delta > 0$. Since the contradictory assumption $T(n) \in o(n\sqrt{n})$ leads to (13), the theorem is proven. □

Note added in proof

Wolfgang Maass (16th ACM-STOC, 1984), Ming Li (Manuscript, Comput. Sci. Department, Cornell Univ., 1984) and the present author (C.W.I., Tech. Rept. CS-R8406, March 1984) have independently improved the lower bound, to simulate two one-head tape units by one (nonoblivious) one-head tape unit, to $\Omega(n^2)$. The actual results are stronger than this in a variety of ways.

References

[1] S.O. Aanderaa, On k-tape versus $(k+1)$-tape real-time computation, in: R.M. Karp, ed., *Complexity of Computation, SIAM-AMS Proceedings Vol. 7:* (Amer. Math. Soc., Providence, RI, 1974) 75–96.

[2] P.C. Fischer, A.R. Meyer and A.L. Rosenberg, Counter machines and counter languages, *Math. Systems Theory* **2** (1968) 265–283.

[3] F.C. Hennie and R.E. Stearns, Two-tape simulation of multitape Turing machines, *J. ACM* **13** (1966) 533–546.
[4] J.E. Hopcroft and J.D. Ullman, *Formal Languages and their Relations to Automata* (Addison-Wesley, Reading, MA, 1969).
[5] L. Janiga, Real-time computations of two-way multihead finite automata in: L. Budach, ed., *Fundamentals of Computation Theory (FCT '79)* (Akademie Verlag, Berlin, DDR, 1979) 214–218.
[6] W. Paul, On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, *Inform. Control* **53** (1982) 1–8.
[7] W. Paul, J. Seiferas and J. Simon, An information-theoretic approach to time bounds for on-line computation, *J. Comput. System Sci.* **23** (1981) 108–126.
[8] N. Pippenger and M.J. Fischer, Relations among complexity measures, *J. ACM* **26** (1979) 361–381.
[9] M.O. Rabin, Real-time computation, *Israel J. Math.* **1** (1963) 203–211.
[10] W.J. Savitch and P.M.B. Vitányi, On the power of real-time two-way multihead finite automata with jumps, *Inform. Process. Lett.* **19** (1984) 31–35.
[11] P.M.B. Vitányi, On the power of real-time Turing machines under varying specifications, *7th Coll. on Automata, Languages and Programming* (ICALP '80), Lecture Notes in Computer Science **85** (Springer, Berlin, 1980) 658–671.
[12] P.M.B. Vitányi, An optimal simulation of counter machines, *SIAM J. Comput.* **14** (1985) to appear.
[13] P.M.B. Vitányi, An optimal simulation of counter machines: The ACM case, *SIAM J. Comput.* **14** (1985) to appear.