Information Processing Letters 21 (1985) 147–152
North-Holland

5 September 1985

AN $n^{1.618}$ LOWER BOUND ON THE TIME TO SIMULATE ONE QUEUE OR TWO PUSHDOWN STORES BY ONE TAPE

Paul M.B. VITÁNYI

Department of Computer Science, Centre for Mathematics and Computer Science (C.W.I.), Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

Communicated by K. Mehlhorn
Received 8 November 1984

To simulate two pushdown stores, or one queue, on-line by a one-head tape unit requires $\Omega(n^{1.618})$ time.

Keywords: Multitape Turing machines, pushdown stores, queues, time complexity, lower bounds, on-line simulation by single-head tape units, Kolmogorov complexity

1980 Mathematics Subject Classification: 68C40, 68C25, 68C05, 94B60, 10-00

1982 C.R. Categories: F.1.1, F.1.3, F.2.3

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 [10,13]. Recently it was shown that k-tape Turing machines require nonlinear time to simulate $(k+1)$-tape Turing machines on-line [9]. More in particular, it was shown that the on-line simulation of two pushdown stores by one tape requires $\Omega(n \log^{1/2} n)$ time.[1] In [12] it was shown that the simulation of a queue by a real-time one-tape Turing machine is impossible. In [14] it appeared that to simulate a pushdown store by an *oblivious* one-head tape unit on-line requires $\Omega(n\sqrt{n})$ time. Here, we combine some ideas from [14] with Kolmogorov complexity [4,10] arguments and an adversary demon to demonstrate an $\Omega(n^{1.618})$ lower bound on the simulation time for one queue, or two pushdown stores, by a (nonoblivious) one-tape Turing machine. This considerably narrows the gap with the best known upper bound, viz., each multitape machine can be simulated on-line by a one-head tape unit in square time [3]. Square time is also the best known upper bound for the simulation of one queue or two pushdown stores by a one-head tape unit. In [15] it is shown that square time is optimal for the simulation of one pushdown store or one queue by an *oblivious* one-head tape unit.

Mutually independent work of Maass [7], Li [6], and the present author [16] has in the meantime shown (the greatest common intersection of the results in these three papers) that square time is optimal for the on-line simulation of two tapes by one tape.

[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 $|f(n)| \leqslant cg(n)$ for all $n \geqslant n_0$;
$f(n) \in \Omega(g(n))$ if there are positive constants c and n_0 such that $f(n) \geqslant cg(n)$ for all $n \geqslant n_0$.

Turing machines and simulation

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 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 i*th* output just before polling 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 $i_1, i_2, \ldots, i_k, \ldots$, polled from the input terminal, the output sequences written to the output terminal are the same for A and B, and if $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_k \leqslant \cdots$ are the steps at which B polls or writes a symbol, from or to the terminals, then there are corresponding steps $t'_1 \leqslant t'_2 \leqslant \cdots \leqslant t'_k \leqslant \cdots$ at which A polls or writes the same symbols and $t'_i \leqslant T(t_i)$ for all $i \geqslant 1$. 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.)

2. Kolmogorov complexity

The ideas on descriptional complexity below were developed independently by Kolmogorov [4] and Chaitin [2]. We closely follow the discussion in [10]. Consider the problem of describing a string x over 0's and 1's. Any computable function f from strings over 0's and 1's to such strings, together with a string y such that $f(y) = x$, is such a description. A descriptional complexity K_f of x, relative to f and y, is defined as

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

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

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

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

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

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

Without loss of generality we assume that all tape units below have semi-infinite tapes. That is, the squares of the tapes can be enumerated from left to right by the natural numbers. The 0th square is called the *start* square. Assume further, also without loss of generality, that the tape units write only 0's and 1's in the storage squares which introduces an extra constant delay. The theorem below improves the known lower bounds (before the recent results in [7,6,16]). 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 inter-square boundary, or square, is called a crossing sequence. Early use of the concept is found in [11].

Theorem 1. *A one-head tape unit for simulating one queue on-line requires* $\Omega(n^{1.618})$ *time.*

Proof. Consider a queue Q which can *store* 0's and 1's polled from the input terminal by appending them to the tail of the currently stored string. We can *retrieve* the contents of Q, one bit at a time, by writing the current front bit to the output terminal. A queue is a *first-in-first-out* storage device. Let M be an actual one-head tape unit simulating a virtual queue Q in time T(n). Intuitively, we have detached Q from its input terminal and output terminal and have replaced it by M which is programmed to behave as if it were Q. Thus, below we can distinguish between M as the embodiment of Q, containing a binary string as polled through the input terminal insofar as the front bits have not yet been retrieved through the output terminal, and the actual encoding of Q's contents on M's storage tape. It is known that $T(n) \in O(n^2)$. Without loss of generality, M has a semi-infinite tape and writes only 0's and 1's. We will argue that M loses a lot of time to either reach or transport earlier stored data, while simulating the virtual queue Q. Consider a sufficiently long incompressible string $x \in \{0,1\}^*$ of length n. Below we use the operational behaviour of M, while simulating Q, to obtain a description of x. To store a substring of x of length at least m, M needs to use $m - O(\log n) - c_M$ work tape squares, with c_M a fixed constant depending only on M, by the incompressiblity of x. Divide the tape in four segments $[0, n/6)$, $[n/6, 2n/6)$, $[2n/6, 3n/6)$ and $[3n/6, \infty)$ (see Fig. 1). An *adversary demon* supplies the sequence of input commands to the input terminal. The adversary demon chooses input commands (for the simulated queue Q) by observing the current actual state of the simulator M. It issues input commands for storing and retrieving the consecutive bits of x in and from the simulated queue Q, as follows:

- If M's storage head resides on $[0, 2n/6)$ when the input terminal is polled, then the next bit of x is input and appended to the tail of the simulated queue Q.
- If M's head resides on $[2n/6, \infty)$ when the input terminal is polled, and the simulated queue Q is nonempty, then the input command is "retrieve the current front bit of the simulated queue Q". If M's head resides on $[2n/6, \infty)$ when the input terminal is polled, and the simulated queue Q is empty, then the next bit of x is input and appended to the tail of the virtual queue Q.
- Having used all of x, the demon retrieves by the subsequent consecutive input commands, one bit at a time, the entire current contents of the simulated queue Q.

First we observe that, given x, the demon must have used all of x within issuing a 2n-length sequence of input commands according to the above strategy. Second, for some $\alpha \leqslant 2$ yet to be choosen, let $T(n) \leqslant cn^\alpha$ for large enough n. Then, by the adversary demon's strategy, the simulator M has to store a prefix u of x with $n/4 \geqslant |u| \geqslant (n/6c)^{1/\alpha}$ while its storage head does not leave the tape segment $[0, n/6)$. Note that $K(u) \geqslant |u| - O(\log n)$. Informally, there are two exhaustive cases of what can happen to the simulator M, faithfully imitating Q, under the adversary input strategy outlined. Both cases lead to the claimed lower bound. *Case* 1: All of u is retrieved by input commands polled with M's storage head on the tapesegment $[2n/6, \infty)$ before all of x has been used by the adversary demon (that is, within 2n input commands). Intuitively therefore, the tapesegment $[n/6, 2n/6)$ in between the locations where M's storage head resides while polling the bits of u and the locations of M's storage head while polling the commands to retrieve the bits of u, needs to be traversed $\Omega(|u|)$ times. This necessitates long crossing sequences for each square of M's tapesegment $[n/6, 2n/6)$. *Case* 2: In the first 2n input commands at least $|x| - |u|$ bits of x are polled while M's head resides on the tapesegment $[0, 2n/6)$ and not retrieved again. Therefore, they need to be kept stored. The available storage space on $[0, 2n/6)$ gets so jammed that part of the encoding of the simulated queue Q's contents on

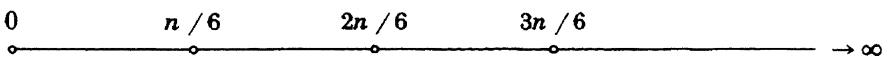

Fig. 1. Division of M's tape in consecutive segments.

this segment of M's tape has to be exported to squares outside this tapesegment. This necessitates long crossing sequences for each square of M's tapesegment [2n/6, 3n/6). In both cases we argue that if $T(n) \notin \Omega(n^{1.618})$, then we could describe an incompressible string in significantly less bits than its length, since it can be reconstructed from a short description of M's behaviour in simulating Q, thus obtaining a contradiction. For convenience in the proof below, we equip M with an extra register in which it stores the last bit it has written to the output terminal.

Case 1. The initially stored prefix u is retrieved one symbol at a time, before the demon has used all of x (i.e., within 2n inputcommands), by commands polled while M's head is on [2n/6, ∞).

In our description of x = uv we given the u in terms of M's operation and v literally as suffix.

- A description of this discussion in O(1) bits.
- A description of M in O(1) bits.
- The value of n in O(log n) bits.
- The location of a square of [n/6, 2n/6) in O(log n) bits.
- The crossing sequence at that square.
- v in $n - |u|$ literal bits.

For a square in [n/6, 2n/6), the crossing sequence associated with that square contains, for each crossing, the time of crossing in O(log n) bits and the state of M in O(1) bits.

To recover u, exhaustively test *every* binary string of length |x|, using the adversary demon's strategy, for consistency with the description above. By definition, x is consistent with the description. Suppose, by way of contradiction, that some other candidate $x' \neq x$ were consistent with it. So $x' = u'v$ and therefore $u' \neq u$. The sequence of bits, of the |u|-length prefix of the string entered through the input terminal, can be extracted from M's special register at the instants just before M polls the command to retrieve a next bit. The retrieve commands are all polled on the tapesegment [2n/6, ∞). The description determines the sequence of instantaneous descriptions of this tapesegment, and the state of M's finite control when the storage head resides on this tapesegment. Consequently, the retrieved string is uniquely fixed by the description above, and if $u' \neq u$, then either u' is retrieved while u was stored, or vice versa, contradicting simulation of the queue Q by M.

Let the minimal number of crossings in a crossing sequence on [n/6, 2n/6) be m. Then the description of x takes

$$O(1) + O(\log n) + O(m \log n) + n - |u|$$

bits. Consequently,

$$|u| \geq (n/6c)^{1/\alpha} \,\&\, |u| \in O(m \log n).$$

Thus, $m \in \Omega(n^{1/\alpha}/\log n)$ and summing the crossing sequences of [n/6, 2n/6) yields

$$T(2n) \in \Omega\left(\frac{n^{1+1/\alpha}}{\log n}\right).$$

Since, by assumption, $T(n) \in O(n^{\alpha})$, we obtain $\alpha \geq (1 + \sqrt{5})/2$ and therefore

$$T(n) \in \Omega(n^{1.618}). \tag{1}$$

Case 2. Suppose that not all of u has been retrieved before the demon has used all of x. This implies that M has polled less than |u| "retrieve current front bit of the simulated queue Q" input commands with its head residing on [2n/6, ∞). Therefore, M must have polled |x| input commands "append next bit of x to the tail of the simulated queue Q" with its head on [0, 2n/6). In our description of x = uv below we give u literally as a prefix and v in terms of M's operation in simulating Q under the demon's strategy.

- u in |u| literal bits.
- A description of this discussion in O(1) bits.
- A description of M in O(1) bits.
- The value of n in O(log n) bits.
- The location of a square of [2n/6, 3n/6) in O(log n) bits. Let this be square p.
- The *final* tape contents of the segment [0, p) in at most n/2 bits.
- The crossing sequence at square p.

For a square of [2n/6, 3n/6), the crossing sequence associated with that square consists of the times of crossing together with the states of M.

To recover v, exhaustively test uv' for *every* binary string v' of length |v|, for consistency with the description above. By definition, v is consistent. Assume, by way of contradiction, that some other word $v' \neq v$ is consistent with the description. The description determines M's final

instantaneous description when the adversary demon has used all of x. The prefix u of x is given explicitly. From this final instantaneous description of M we can, by polling at most x consecutive "retrieve front bit of simulated queue Q" commands, retrieve the suffix of u (insofar not yet retrieved) followed by a unique $|v|$-length string. Consequently, M would erroneously retrieve the same word in the first $|x|$ unstore commands for both $x = uv$ and $x = uv'$ under the strategy of the adversary demon. This contradicts that M simulates Q.

Let the shortest crossing sequence on $[2n/6, 3n/6)$ consist of m crossings. Then the above description of x takes at most

$$|u| + O(1) + O(\log n) + n/2 + O(m \log n)$$

bits. Since x is incompressible, it follows that $m \in \Omega(n/\log n)$, and summing the crossing sequences of $[2n/6, 3n/6]$ yields

$$T(2n) \in \Omega\left(\frac{n^2}{\log n}\right), \tag{2}$$

which, together with inequality (1) in Case 1, proves the theorem. □

Since four one-head tape units can simulate a queue in real-time [5], the lower bound on the time to simulate many storage tape units by one is set to $\Omega(n^{1.618})$. The lower bound also holds for the simulation of two pushdown stores.

Theorem 2. *A one-head tape unit for simulating two pushdown stores on-line requires* $\Omega(n^{1.618})$ *time.*

Proofsketch. The proof is essentially the same as the proof of Theorem 1. Let M be an actual one-head tape unit simulating two virtual pushdown stores P_1 and P_2 in time T(n). Without loss of generality, M has a semi-infinite tape and writes only 0's and 1's. Consider a sufficiently long incompressible string $x \in \{0,1\}^*$ of length n. Divide the tape of M in four segments as done previously (see Fig. 1). An adversary demon supplies the sequence of input commands.

- Initially, while the simulator M's storage head has not yet ranged outside the tapesegment $[0, 2n/6)$, the polled input commands push the next bit of x on the simulated pushdown store P_1. The simulator M's storage head must range outside $[0, 2n/6)$ by the incompressibility of x. Ever after it has first done so, the input commands polled with M's storage head residing on $[0, 2n/6)$ push the next bit of x on the other simulated pushdown store P_2.
- If M's storage head resides on $[2n/6, \infty)$ at poll time and the simulated pushdown store P_1 is nonempty, then the polled input command pops the top of the simulated pushdown store P_1. If M's head resides on $[2n/6, \infty)$ at poll time and the simulated pushdown store P_1 is empty, then the polled input command pushes the next bit of x on the simulated pushdown store P_2.
- Having used all of x, the demon pops consecutively all of the remaining contents of the simulated pushdown stores P_1 and P_2.

It is immediately clear that given x the demon must have used all of x within a 2n-length sequence of input commands. Choose T(n) and u as in the previous proof. The same reasoning in Cases 1 and 2 is mutatis mutandis applicable. □

Corrollary. *The result implies trivially than an* ***oblivious*** *one-head tape unit for simulating one pushdown store on-line requires* $\Omega(n^{1.618})$ *time.*

References

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

[2] G.J. Chaitin, Algorithmic information theory, IBM J. Res. Develop. 21 (1977) 350–359.

[3] J.E. Hopcroft and J.D. Ullman, Formal Languages and Their Relations to Automata (Addison-Wesley, Reading, MA, 1969).

[4] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Problems in Information Transmission 1 (1) (1965) 1–7.

[5] B.L. Leong and J.I. Seiferas, New real-time simulations of multihead tape units, J. Assoc. Comput. Mach. 28 (1981) 166–181.

[6] M(ing) Li, On one tape versus two stacks, Unpublished manuscript, Computer Science Dept., Cornell Univ., 1984.

[7] W. Maass, Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines, 16th ACM Symp. on Theory of Computing (1984) 401–408.

[8] W.J. Paul, On heads versus tapes, 22nd IEEE Symp. on Foundations of Computer Science (1981) 68–73; also in: Theoret. Comput. Sci. 28 (1,2) (1984) 1–12.

[9] W.J. Paul, On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, 23rd IEEE Symp. on Foundations of Computer Science (1982) 53–56; also in : Inform. and Control 53 (1982) 1–8.

[10] W.J. Paul, J. Seiferas and J. Simon, An information-theoretic approach to time bounds for on-line computation, 12th ACM Symp. on Theory of Computing (1980) 357–367; also in: J. Comput. System. Sci. 23 (1981) 108–126.

[11] M.O. Rabin, Real-time computation, Israel J. Math. 1 (1963) 203–211.

[12] M.K. Valiev, Certain estimates of the time of computations on Turing machines with an input, Cybernetica 6 (1972) 734–741; translated from: Kibernetica 6 (1970) 309–317.

[13] P.M.B. Vitányi, On the power of real-time Turing machines under varying specifications, 7th Internat. Coll. on Automata, Languages and Programming, Lecture Notes in Computer Science 85 (Springer, Berlin, 1980) 658–671.

[14] P.M.B. Vitányi, On the simulation of many storage heads by one, 10th Internat. Coll. on Automata, Languages and Programming, Lecture Notes in Computer Science 154 (Springer, Berlin, 1983) 687–694; also in: Theoret. Comput. Sci. 34 (1,2) (1984) 157–168.

[15] P.M.B. Vitányi, Square time is optimal for simulation of one pushdown store or one queue by an oblivious one-head tape unit, Inform. Process. Lett. 21 (2) (1985) 87–91.

[16] P.M.B. Vitányi, One queue or two pushdown stores take square time on a one-head tape unit, Tech. Rept. CS-R8406, Computer Science Dept., Centre for Mathematics and Computer Science (C.W.I.), Amsterdam, 1984.