Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.M.B. Vitányi

Square time is optimal for simulation of one pushdown store by an oblivious one-head tape unit

Department of Computer Science Report CS-R8402 January

SQUARE TIME IS OPTIMAL FOR SIMULATION OF ONE PUSHDOWN STORE BY AN OBLIVIOUS ONE-HEAD TAPE UNIT

PAUL M.B. VITANYI

Centre for Mathematics and Computer Science, Amsterdam

To simulate a pushdown store or queue on-line by an oblivious one-head tape unit takes at least square time. Since each multitape Turing machine can be trivially simulated by an oblivious one-head tape unit in square time this result is optimal.

1980 MATHEMATICS SUBJECT CLASSIFICATION: 68C40, 68C25, 68C05, 94B60, 10-00.

1982 CR CATEGORIES: F.1.1, F.1.3, F.2.3.

KEYWORDS & PHRASES: multitape Turing machines, pushdown stores, queues, time complexity, square lower bounds, on-line simulation by oblivious single-head tape units, Kolmogorov complexity.

NOTE: This report will be submitted for publication elsewhere.

Report CS-R8402
Centre for Mathematics and Computer Science,
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. Introduction

Each multitape Turing machine can be simulated by an *oblivious* one-head tape unit in square time. Such a simulation is essentially the same as the obvious one by nonoblivious one-head tape units in [HU]. For simulation by an oblivious one-head tape unit we can derive a lower bound which matches this upper bound on the simulation time. So in this case the obvious simulation is also optimal. The best previous lower bound on this simulation time was $n^{1.618}$ in [Vi1]. 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, see for instance [Vi2]. We also assume that the steps at which the input is polled are the same for all input streams. It is obvious that we can simulate one device BLAH by an *oblivious* one-head tape unit in time $T(n)$ iff we can simulate a multiBLAH machine that way [Vi2]. Thus it suffices to show the lower bound on the time to simulate a queue or a pushdown store. The proof uses Kolmogorov Complexity as [Vi1] and some references in that paper. We use "simulation" in the strong sense of "on-line simulation" throughout. A one-head tape unit is used as synonym for a Turing machine with one single-head storage tape.

2. Kolmogorov Complexity

The ideas on descriptional complexity below were developed independently by Kolmogorov [Ko] and Chaitin [Ch]. We follow [PSS]. Consider the problem of describing a vector $\bar{x}$ of strings x_i over 0's and 1's. The string entries of the vector can be separated by ¢'s so that the vector is a string too. That is, $\bar{x} \in \{0, 1, ¢\}$. Any computable function f from vectors of strings over 0's and 1's to such vectors, together with a vector $\bar{y}$, such that $f(\bar{y}) = \bar{x}$, is such a description. A descriptional complexity K_f of $\bar{x}$, relative to f and $\bar{y}$, is defined by

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

For the *universal* computable partial function f_0 we have that, for all f with appropriate constant c_f, for all vectors $\bar{x},\bar{y}$, $K_{f_0}(\bar{x} \mid \bar{y}) \leq K_f(\bar{x} \mid \bar{y}) + c_f$. So the canonical relative descriptional complexity $K(\bar{x},\bar{y})$ can be set equal to $K_{f_0}(\bar{x} \mid \bar{y})$. Define the *descriptional complexity* of $\bar{x}$ as $K(\bar{x}) = K(\bar{x} \mid \epsilon)$. ($\epsilon$ denotes the empty string.) Since there are 2^n binary strings of length n, but only $2^n - 1$ possible shorter descriptions d, it follows that $K(x) \geq \mid x \mid$ for some binary string x of each length. We call such strings *incompressible*. It also follows that $K(x \mid y) \geq \mid x \mid$ for some binary string x of each length. Since similarly $K(x) \geq (1-\delta) \mid x \mid$ for $2^{\delta \mid x \mid}$ strings over $\{0,1\}$, which thus cannot be compressed to less than $(1-\delta) \mid x \mid$ bits, such "nearly" incompressible strings are abundant. Note that a string $x = uvw$ can be specified by v, $\mid x \mid$, $\mid u \mid$ and the bits of uw. Thus,

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

so that with $K(x) \geq (1-\delta) \mid x \mid$ we obtain

$$K(v) \geq \mid v \mid - \delta \mid x \mid - O(\log \mid x \mid) \; .$$

3. The square lower bound

Without loss of generality, we assume that the 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 and relax the *real-time* requirement to *constant delay*. A computation is of *constant delay* if there is a fixed constant c such that there are at most c computation steps in between processing the nth and the $(n+1)$th inputsymbol, for all n. Thus, constant delay with $c = 1$ is the same as real-time, and it is not difficult to see that each computation of constant delay

can be sped up to a real-time computation by expanding the storage alphabet and the size of the finite control.

Theorem. *The fastest simulation of a pushdown store by an oblivious one-head tape unit takes* $\Theta(n^2)$ *time.*

Proof. The only thing we have to prove is the square lower bound for some device, real-time simulatable by one or more pushdown stores. So let us consider a queue Q. Let M be an oblivious one-head tape unit simulating the virtual queue Q in time $T(n)$. Without loss of generality M has a semi-infinite tape and writes only 0's and 1's. An adversary demon supplies the sequence of input commands. The adversary demon first determines the *rightmost* intersquare boundary B such that left of the boundary there are less than n "read input"'s in the first $2n$ "read input"'s. If there happen at least $n/4$ "read input"'s on the cell immediately right of B then $T(n) \in \Omega(n^2)$, cf. Case 1 below. Therefore, there must be at least $3n/4$ "read input"'s left of B, and at least n right of B. The adversary demon supplies the sequence of input commands. Divide the tape into three segments $[0, B)$, $[B, C)$ and $[C, \infty)$, with the length of the middle segment $[B, C)$ being $n/16$ squares. See Figure 1.

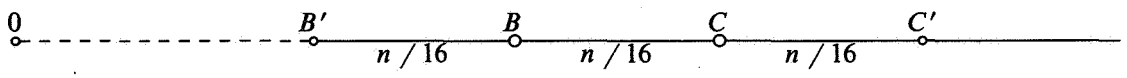

Figure 1.

Case 1. Suppose that at least $n/4$ "read input"'s fall in the middle segment $[B, C)$. Consider a sufficiently long incompressible string $x \in \{0,1\}^*$ of length $2n$. The first $2n$ input commands, supplied by the adversary demon, are "store next unread bit of x". In the description of x below we give most of x in concatenated literal form in a suffix v and the concatenated remainder w of x in terms of M's operation. (Therefore x is a *shuffle* of v and w.)

- 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 *any* pair of squares (p, q) with p in $[B', B)$ and q in $[C, C')$, where the lengths of both $[B', B)$ and $[C, C')$ are $n/16$. This takes $O(\log n)$ bits. See Figure 1.
- The crossing sequence at that pair (p, q) of squares, as described below.
- The *final* contents of the tape segment $[p, q]$, after $2n$ "read input"'s of M have been executed, processing all of x.
- The concatenated literal remainder v of x in not more than $7n/4$ bits.

For *any* pair (p, q) of such squares, the crossing sequence associated with that pair contains for each crossing the state of M and whether we enter/leave $[p, q]$ from/to left or right in O(1) bits. Associated with each *entrance* of $[p, q]$ we give the number of times the input is polled up to the corresponding *leave* of $[p, q]$; summed over all of the crossing sequence this does not take more than $O(l \log(n/l))$ bits, where l is the length of the crossing sequence. To recover x, start reading the literal representation v until the head of M enters $[p, q]$. Try all continuations which lead in the correct number of inputs to the correct exit of $[p, q]$ and continue with the literal representation v, and so on. Finally, after having processed $2n$ bits, which includes all of v, and matching the final contents of $[p, q]$, the resulting machine i.d. must store x, which can be retrieved by $2n$ pop commands. Let the minimal length of any crossing sequence for a pair (p, q) be $m(n)$. Then the

description of x takes not more than:

$$O(1) + O(\log n) + O(m(n)\log(n / m(n))) + 3n / 16 + 7n / 4$$

bits. Since this amount must be at least $K(x) = 2n$, it follows that $m(n)\log(n / m(n)) \in \Omega(n)$ and therefore $m(n) \in \Omega(n)$. Summing the lengths of the crossing sequences of a set S of all pairs (p, q), such that if $(p_1, q_1), (p_2, q_2) \in S$ then $p_1 \neq p_2$ and $q_1 \neq q_2$, must give a lower bound on the running time. Therefore $T(2n) \geq (n / 16)m(n)$. Hence, $T(n) \in \Omega(n^2)$.

Case 2. Not more than $n / 4$ input commands are polled on $[B, C)$. Therefore, out of the first $2n$ input commands, more than $3n / 4$ input commands are polled in $[0, B)$ and more than $3n / 4$ input commands in $[C, \infty)$. Let y be an incompressible binary string of length $3n / 4$. The input is now supplied by the adversary demon as follows.

- The input commands left of B consist of storing the next unread bit of y.
- The input commands right of C consist of unstoring Q. In case Q is empty the "skip" instruction which does not change anything is polled.
- The input commands polled on $[B, C)$ are "skip" instructions, which do not change anything.

This in case that at least one-half of the initial $3n / 4$ store commands received on $[0, B)$ store bits of y which are subsequently unstored by a corresponding one-half of the $3n / 4$ unstore commands received on $[C, \infty)$. If this is not the case then the roles of $[0, B)$ and $[C, \infty)$ in the above input strategy of the demon and in the sequel of the argument are reversed. The argument then proceeds symmetrical. That these two possibilities are exhaustive is argumented below.

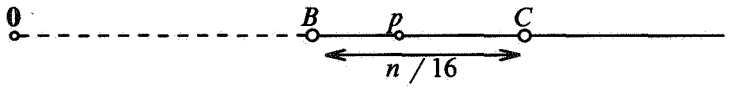

Figure 2.

It is immediately clear that given y the demon must have used all of y within $2n$ input commands. Also within $2n$ input commands, at least $3n / 8$ bits of y stored on $[0, B)$ have been retrieved on $[C, \infty)$, or *vice versa* under the alternative strategy of the demon. (If we look at the subsums of the initial i elements of a $2m$ length sequence consisting of m elements "$+1$" and m elements "-1" then either the number of i's such that the sum from the 1th up to the ith element is greater than 0 exceeds $m / 2$ or the number of i's such that the sum from the 1th up to the ith element is smaller than 0 exceeds $m / 2$. From this it easily follows that in each particular sequence of m stores and m unstores either there must be at least $m / 2$ stored items which are subsequently unstored or this holds for that sequence with the role of store and unstore everywhere interchanged.) In the description of y below we give part of y literally in a suffix v and part w of y in terms of M's operation. Now y is a *concatenation* of v and w, that is $y = wv$

- 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 *any* square p on $[B, C)$ in $O(\log n)$ bits.
- The crossing sequence at p.

- The literal suffix v of y in not more than $3n / 8$ bits.

The crossing sequence associated with p contains for each crossing the state of M and with each entrance of $[0,p)$ the number of times the input is polled up to the corresponding leave of $[0,p)$. Similar to Case 1, if $l(p)$ is the length of the crossing sequence at p then the crossing sequence can be denoted in not more than $O(l(p)\log(n / l(p)))$ bits. Let the minimum length of such a crossing sequence in $[B, C)$ be $m(n)$. Then the description of y takes not more than

$$O(1) + O(\log n) + O(m(n)\log(n / m(n))) + 3n / 8$$

bits. To recover y, try all binary strings of appropriate length which lead to the correct exits of $[0,p)$. Inputs on $[C, \infty)$ constitute all of the unstores and are supposed to yield the consecutive bits of y. To that purpose, attach a special register to M's finite control which remembers the last answer to a query; this for the cases where the answer to the query is output *left* of p on $[0,p)$ (or *right* of p on $[p, \infty)$ under the alternative strategy of the demon). This slightly enlarges M's finite control, but leaves the preceding discussion intact. Thus, by extracting the consecutive bits from this register just before each query we must obtain at least the first $3n / 8$ bits of y. The $3n / 8$-length suffix v is given literally in the description. Since $K(y) = 3n / 4$ we have $m(n)\log(n / m(n)) \in \Omega(n)$. Therefore $m(n) \in \Omega(n)$. Minorizing the running time $T(2n)$ by summing the lengths of the crossing sequences over all squares of $[B, C)$ to at least $m(n)n / 16$ we obtain $T(2n) \in \Omega(n^2)$. □

REFERENCES

Ch Chaitin, G.J., Algorithmic Information Theory, *IBM J. Res. Dev. 21* (1977) 350 - 359.

HU Hopcroft, J.E., and J.D. Ullman, *Formal languages and their relations to automata.* Addison-Wesley, 1969.

Ko Kolmogorov, A.N., Three approaches to the quantitative definition of information, *Problems in Information Transmission, Vol. 1, No. 1,* (1965) 1 - 7.

PSS Paul, W.J., J.I. Seiferas & J. Simon, An information-theoretic approach to time bounds for on-line computation, 12th ACM Symposium on Theory of Computing, 1980, 357 - 367.

Vi1 Vitányi, P.M.B., An $N^{1.618}$ lower bound on the time to simulate one queue or two pushdown stores by one tape. Technical Report IW 245, Mathematisch Centrum, Amsterdam, dec. 1983.

Vi2 Vitányi, P.M.B., An optimal simulation of counter machines, *SIAM J. Comput.,* in press.