On One Tape Versus Two Stacks

Ming Li

TR 84-591
January 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

On One Tape versus Two Stacks

Ming Li*
Department of Computer Science
Cornell University
Ithaca, New York 14853

Abstract

We develop a simple method which enables us to prove three new lower bounds (for both worst and average cases) for on-line computations, answering two open problems summarized in [DGPR].

We give a language that requires $\Omega(n^2)$ time for any 1-tape deterministic on-line machine, but it can be accepted by a 2-stack 1-reversal bounded deterministic on-line machine in real time. This provides a tight lower bound, closing the gap between $\Omega(n(logn)^{1/2})$ lower bound by [P2] and the trivial $O(n^2)$ upper bound.

We also prove that 1-tape nondeterministic real time is much stronger than its deterministic version. For 1-tape on-line machines, we give language L (L") which is in nondeterministic linear (real) time but requires $\Omega(n^2)$ ($\Omega(n^{1.5})$) deterministic time.

Finally we give a language which can be accepted by a 2-stack 1-reversal bounded deterministic machine in real time, but it requires $\Omega(n^{1+1/2})$ time for any one tape nondeterministic online machine. This sharply improves an *nlogn* lower bound in [DGPR].

* This work was supported in part by an NSF grant MCS-8301766.

1. Introduction

Obtaining good lower bounds has been one of most important issues, of both practical and theoretical interests, in theoretical computer science. While nontrivial lower bounds for general models still seems beyond our grasp, there has been some success for restricted models, which does offer some insights and proof techniques. But even these proofs appeared to be quite hard.

In this paper we will consider on-line computations which is generally used for investigating the dependency of the computational power on the number of tapes. We call a TM M a k-tape on-line machine if M has a 1-way read only input tape and k working tapes. Without explicit indication, all the machines in this paper will be online machines. An on-line machine M works in *real time* if each time M reads an input symbol it makes only constant number of moves. A 2-stack machine is like an usual pushdown automaton but with two pushdown stores. A k-stack machine is 1–*reversal* bounded if for each stack, once it starts popping, no more pushing is allowed.

One classical question about TM's is that how much power does an additional working tape have. For real time deterministic computations, early in 1963, Rabin [R] proved 2 tapes are better than 1. 11 years later Aanderaa [A] generalized Rabin's result to k+ 1 vs k. In 1982, Duris and Galil [DG] proved, by the crossing sequence technique, that two tapes are better than one for nondeterministic case. There has been no nonlinear lower bounds obtained until recently Paul [P] proved, using Kolmogorov-complexity, on-line simulation of k+ 1 tapes by k tapes requires $\Omega(n(logn)^{1/k+1})$. Later Duris, Galil, Paul, and Reischuk [DGPR] proved, for non-deterministic machines, simulating 2 tapes by 1 tape requires Ω(nloglogn) time. (I think they improved this to logn.) And in that paper ([DGPR]) they raised the open questions (open problems 1 and 7): whether the gap between the upper bound $O(n^2)$ and the lower bound $nlogn^{1/2}$ in deterministic case and the lower bound *nloglogn* in nondeterministic case for 1 tape simulating k(2) tapes can be narrowed. Notice that for 2 tapes simulating k, in deterministic case, we have only logn time loss by [HS] which is pretty close to Paul's lower bound; in nondeterministic case, we do not even lose any time [BGW]. Thus the most interesting and intriguing question left seems to be whether we can narrow the gaps for

(1) 1 tape simulating 2 tapes;

(2) 1 tape simulating 2 stacks;

(3) 1 tape simulating 2 1-reversal bounded stacks.

In this paper we will close or radically narrow these gaps.

In the first theorem we will close the gap between $n(logn)^{1/2}$ and $O(n^2)$ for the deterministic case. We prove:

Theorem 1: There is a language L which can be accepted in real time by a 1-reversal 2-stack deterministic machine, but it requires $\Omega(n^2)$ time for any 1-tape deterministic on-line machine.

Remark 1: Notice that due to a result of Hennie and Stearns [HS] which says 2 tapes simulation of k tapes can be done in time O(nlogn), our result will not apply to the question of k vs k+ 1 for k>1 tapes.

We also prove that nondeterministic 1-tape online TM is much stronger than its deterministic version:

Theorem 2: There is a language L (L") which can be accepted by a 1-tape nondeterministic machine in linear (real) time, but it requires $\Omega(n^2)$ ($\Omega(n^{1.5})$) time for any 1-tape deterministic on-line machine.

Remark 2: The general separation task for deterministic time vs nondeterministic time is well known to be very hard. One recent well known result by [PPST] states: NTIME[n] $\neq$ DTIME[n]. But it is not clear how to restrict this result to 1-tape online machines. [M] and [G] considered the class of TM's with a single tape which serves for both input and working space. (Notice that our 1-tape real time model is not as weak as one might think to be, it can simulate k counters for any k [V]; it can accept the palindrom language which requires n^2 time for above single tape machine even with the power of nondeterminism.) For such TM's, Gill proved that there is an L which can be accepted in nondeterministic *nlogn* time but not in deterministic $n^{1.99}$ time. Maass, using the idea of alternation, proved there is an L which is in nondeterministic linear time but not in deterministic $n^{1.22}$ time.

For nondeterministic case we present the following result which also sharply improves the *nlogn* lower bound in [DGPR],

Theorem 3: There is a language L which is accepted in real time by a 2-stack 1-reversal deterministic machine, but which requires $\Omega(n^{1.5})$ time for any nondeterministic 1-tape online machine.

Remark 3: Theorem 1 and Theorem 3 destroy the possibility that the Hennie and Stearns simulation (even to simulate only two 1-reversal bounded stacks) might be done by 1 tape in either deterministic or nondeterministic case. The former *nlogn* lower bound left this possibility open.

2. Some Kolmogorov complexity

The concept of Kolmogorov-complexity (K-complexity) was originated in [K], and independently by Chaitin and Solomonoff. The K-complexity has been applied to computer science by a number of authors including [H], [Si], [PSS], ... Using K-complexity as a tool of lower bound proof was first introduced in [P], [P1], [P2], [PSS]. In this paper we will apply K-complexity to obtain our lower bounds.

For our convenience, we will use a slightly different definition of K-complexity. The *K−complexity* of a string X (K(X)) is the length of the shortest program (deterministic or nondeterministic) that *accepts only* X. The *K−complexity* relative to string Y of X ($K(X \mid Y)$) is the length of shortest program (det. or nondet.), with Y as its extra information, that accepts only X. X is *random* if K(X)$\geq |X|$.

Remark 2.1: One can easily prove the our acceptor definition with the power of nondeterminism is identical (within a constant) to the traditional deterministic generator definition.

Fact 1: Most strings are random.

Fact 2: If X=uvw, and X is random, then K($v \mid uw$)$\geq |v| - log\ |X|$. i.e. random strings are locally almost random.

Remark 2.2: Suppose $x_1, \ldots, x_k$ are separated blocks of equal length, with gaps in between, spreaded on the input tape. Then we can compress them (deleting the gaps) as follows,

(# of x_i's) $x_1 x_2 \ldots$; (length) $(\bar{p}_1,d_1)(p_2,d_2)\ldots$

Where (p_i,d_i) indicates that the next p_i consecutive x_k's are one group (adjacent to each other), and followed by a gap of $d_i * |x_i|$ long. And p_i's and d_i's are self delimited. We omit the representation details, but one thing should be clear: the gaps can be represented short.

3. Some definitions and a jamming lemma

In this section we first give some definitions, then prove an intuitively straightforward but important lemma.

Throughout the paper variables X, Y, x_i, y_i ... denote strings in Σ^* for $\Sigma=\{0,1\}$.

For a 1-tape on-line machine M, we shall call its input head h_1, and its working tape head h_2. Let x_i be a block of input, R be a region on the working tape. We say that M *maps* x_i *into* R if while h_1 is reading x_i h_2 never goes out of region R; We say *onto* in above if h_2 also travels the *whole* region R while h_1 reads x_i. The *range* of x_i is the region R s.t. M maps x_i onto R. See Fig.0.

A crossing sequence (c.s.) for a point on the working tape of M is a sequence of ID's, where each ID is of form (state of M, h_1's position).

Remark 3.1: Since h_1 is one way, we can represent each ID in a c.s. as follows:

ID_i=(state of M, current h_1's position - h_1's position of ID_{i-1})

Thus if a c.s. has d ID's, and the input length is n, the total space we need is $< d\,|M| + \log k_1 + \ldots + \log k_d$, with $\Sigma k_i = n$, which is $< d\,|M| + d\log n/d$ by a standard calculation (i.e. maximize the function).

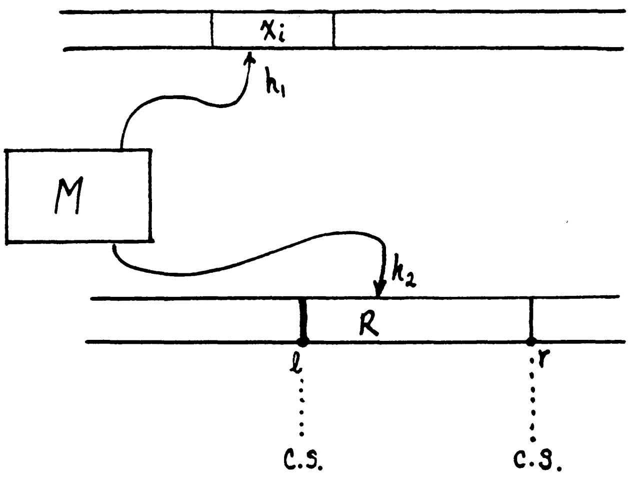

Figure 0

Next, we prove our jamming lemma, which coincides our intuition that a small region with short c.s around it cannot hold a lot information. Formally,

Jamming Lemma: Suppose on input $x_1 x_2 \cdots x_k \#$ ..., where x_i's have equal length, M maps $x_{i_1}, \cdots, x_{i_l}$ into region R when h_1 reaches #. Then the contents of the working tape at this time can be reconstructed by using only $\{x_1, \cdots, x_k\}$ - $\{x_{i_1}, \ldots, x_{i_l}\}$, contents of R, the two c.s.'s around R (see fig.0), plus $2(logp_1 + \ldots + logp_k)$ space as in Remark 2.2 to indicate the positions of the remaining x_i's, where $\Sigma p_i = l$.

Remark: Roughly speaking, if $\Sigma |x_i| > 2(|R| + 2|c.s.| + |M|)$, then the Jamming Lemma implies the either $X = x_1 \ldots x_k$ is not random or we lost some information about X.

Proof: We simulate M as follows: put input $\{x_1, \ldots, x_k\}$ - $\{x_{i_1}, \ldots, x_{i_l}\}$ at their correct positions on the input tape (as indicated by the distances). Then run M with h_2 staying to the left of R. If h_2 reaches R at point l (see fig.0), we interrupt M (In the nondeterministic case we also check the current state and h_1 position.) and take the next ID in the c.s. at point l according to which we relocate h_1 and adjust state of M and then go on running M. After we finish at left of R, we do the same thing at the right. Then finally we put the contents of R into region R. Notice that although there are many empty regions on the input tape corresponding to those unknown x_i's, h_1 will never read these regions because h_2 never goes into R. □

Corollary: If M is nondeterministic, the above Jamming Lemma is still true except that we rephrase 'contents of working tape' as 'legal contents of working tape' which simply means some computation path on the same input would create this working tape contents.

4. A language that requires $\Omega(n^2)$ time for a 1-tape deterministic machine

Define a language

$L = \{x_1 \$ x_2 \$ \cdots \$ x_k \# y_1 \$ \cdots \$ y_l \# (1^{i_1}, 1^{j_1})(1^{i_2}, 1^{j_2}) \ldots (1^{i_s}, 1^{j_s}) \mid x_p = \bar{y}_q$ where $p = i_1 + \ldots + i_t$, $q = j_1 + \ldots + j_t$ for all $t = 1, \ldots, s\}$.

In the following we will be generous on the constants for an easier proof. Anybody who does not want details may change all C's in the following to $logn$ to get a bound $n^{2-\epsilon}$ for any $\epsilon > 0$.

Lemma 1. Any deterministic one-tape on-line TM accepting L requires $\Omega(n^2)$ time.

Proof : To the contrary, suppose an one-tape on-line deterministic machine M accepts L in $o(n^2)$. Let h_1 and h_2 be the heads on M's one-way input tape and working tape, respectively. We will derive a contradiction by showing that a random string has a short program.

Let us assume that each tape square of M's working tape can be coded by C_0 binary bits. We fix a large constant C s.t. $C>100C_0|M|$, $2nlogC/C<n/48$, and $(logC)n/C^2<n/8C$.

First we find a string X of length n s.t. $K(X) \geq |X|$. Then equally break X into $k=n/C$ parts, each $l=C$ long, and call them $x_1, x_2, \cdots, x_k$. Consider the partial input $x_1\$x_2\$ \cdots \$x_k\#$ ... to M. If for >k/2 x_i's, while h_1 is in the range of $\$x_i\$$, h_2 moves around on the working tape for a range $>n/C^3$, then M requires $O(n^2)$ time, a contradiction. So we can assume that for > k/2 x_i's, while h_1 is in the range of $\$x_i\$$, h_2 moves only in a range of size $< n/C^3$ on the working tape. We order these x_i's by their first h_2 positions. And let x_c be the median.

Now we consider 2 cases. In the first case we assume that many x_i's are jammed(mapped) into a small region, that is, when h_1 is reading them, h_2 is always in this small region. We shall show that then X is not random by the Jamming Lemma; In the second case, we assume there is no jamming region, x_i's are spread evenly on the working tape. Then we will arrange the y_i's s.t. there will be many pairs $x_j = y_i$, so that x_j and y_i are mapped far apart. Then we argue that it takes long time for M to match x_j and y_i.

Case 1 (jammed): There exist k/C x_i's and a fixed region R of length n/C^2 on the working tape s.t. M maps these x_i's into R.

We will show that a short program can be constructed which accepts only X. Consider two $|R|$ long regions to the left and right of R on the working tape, call them R_l and R_r, resp. In each of them we find positions p_l and p_r, which has shortest crossing sequence (c.s.) in that region. (see figure 1.) If the shortest c.s. in either region is $> n/C^3$, then M uses $\geq O(n^2)$ time , a contradiction. So we assume they are both shorter than n/C^3. Let region R_l' (R_r') be the portion of R_l (R_r) right (left) to p_l (p_r) as shown in Fig. 1.

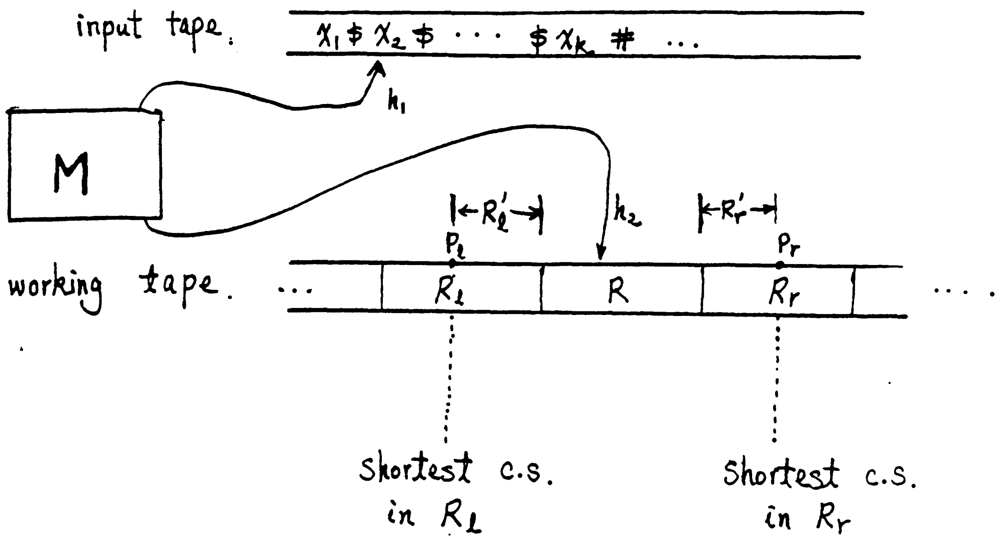

Figure 1

Now, by the Jamming Lemma, with the information of other (without the jammed ones) $k-k/C$ x_i's, the two c.s.'s, and $R_l'RR_r'$, we can tell if a string Y is X by runnig M as follows: Construct the working tape of M at the time h_1 reads to first # sign; Check if $|Y|=|X|$; Then divide Y into k equal pieces and form $y_1\$ \cdots \$y_k$. (we need a coding here but it is trivial.) As the second part of the input we put $y_1\$...\$y_k$ #(1,1)(1,1)...(1,1). (k groups) If M answered Yes, then we know Y=X.

We have to represent the c.s.'s and k-k/C x_i's in a small space. For the c.s.'s, by Remark 3.1, we can use $n/C^3(|M|+logC^3)$ space to write them. For the x_i's, use the representation of Remark 2.2, we need no more than $n-n/C+n/4C$ space (by a standard maximization calculation and the choice of C).

Now the length of this program is only the sum of: $n-n/C+n/4C$ for $k-k/C$ x_i's; $3n/C^2$ for the regions R_l', R, and R_r'; $n/C^3(|M|+3logC)$ for the c.s.'s; and O(logn) for some counters; total less than n for large n's. Thus K(X)< $|X|$, a contradiction.

Case 2 (not jammed): For each fixed region R, with $|R|=n/C^2$, there are only $< k/C$ x_i's such that while they are read by h_1, h_2 is always in R.

Now we fix a region of n/C^2 long which contains x_c. Call it R_c. By a counting argument, there must be r_1=k/6 x_i's, say, $x_{i_1}, \cdots, x_{i_{r_1}}$, (name this set S_r) s.t. when they are read by h_1, h_2 is always to the right of R_c; And same thing is true for the left, name those x_i's $x_{j_1}, \cdots, x_{j_{l_1}}$ and name this set S_l, $l_1=k/6$. Let us re-order them and assume $i_1<i_2<...<i_{r_1}$, and $j_1<j_2<...<j_{l_1}$.

Now we arrange y_i's. Let y_1 be x_{i_1}, y_2 be x_{j_1}, y_3 be x_{i_2}, y_4 be x_{j_2}, and so on. See fig. 2.

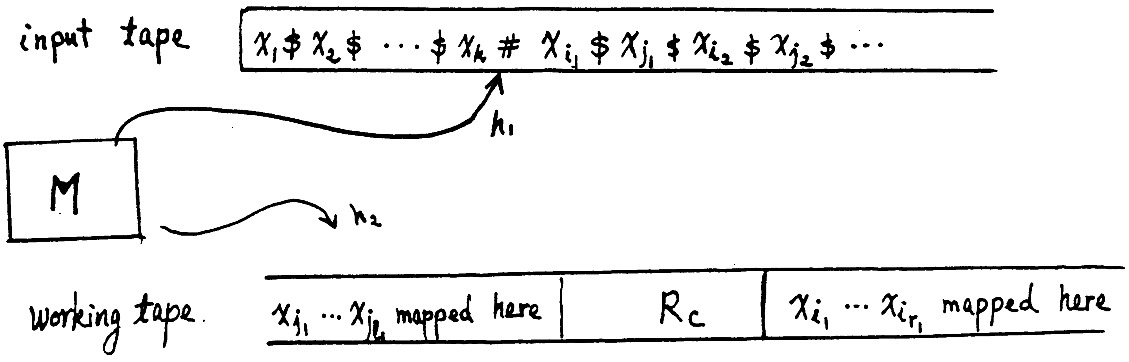

Figure 2

Let p=2min$\{r_1, l_1\}$. We feed $y_1\$ \cdots \$y_p\$$ as our partial input to M following $x_1\$ \cdots \$x_k\#$. If there does not exist k/12 pairs y_i and y_{i+1} such that while h_1 is reading them, h_2 travels only in a range of size less than $n/4C^2$, then M uses n^2 time, a contradiction. Otherwise if we have such k/12 pairs, then by our order of y_i's and a simple counting, there will be a region R ($\subset R_c$) of length $n/4C^2$ and there exist k/24 x_i's from one of S_r and S_l mapped to one side of R and their corresponding y_j's (with $y_j=x_i$) mapped to the other side of R. Let the set of these x_i's be S_x, and the set of these y_i's be S_y. We know when h_1 reads S_x, h_2 is at one side of R; and when h_1 reads S_y, h_2 is at the other side of R. And $|S_x|=|S_y|=k/24$. Let the indices in S_x be $a<b<c<...$, and the indices in S_y be $u<v<w<....$ By our previous arrangement we know x_a should match to y_u, x_b to

y_v, and so on. Now we can complete our input to M with

(1) $\#(1^a,1^u)(1^{b-a},1^{v-u})(1^{c-b-a},1^{w-v-u})$ ...

After M finishes its computation, we find a position p in R which has shortest c.s.. Let $C_1 = n/(|S_x| * l)$. If this c.s. is longer than n/C_1C^3, then M uses time $O(n^2)$, a contradiction. If it is less than n/C_1C^3, then in the following we again construct a short program to accept X.

For input Y, we first compare it with $\{x_1, \dots, x_k\}$-S_x. To compare x_i's in S_x, we run M: we leave the places for x_i's in S_x arbitrary, and put y_l's arranged as above (all y_i's of S_y, the un-compared part, will be in it.) followed by (1) above. Then using the c.s. at point p we run M s.t. h_2 will stay in S_y's side and it will never pass the point p (if h_2 meets p, we check if the c.s.'s match and if they do we take the next ID in c.s. to continue to run M). If M accepts, we accept; If M does not accept or it messes things up (e.g., when M runs to p, it is not in the same state or h_1 position as it is indicated in the c.s..), then we reject. Notice that it is possible M accepts or rejects on the other side (S_x's side) of point p, in which case if all ID's match our accepting c.s. we know M accepts since once h_2 passes point p it does not read S_y and all other y_i's are 'good' ones (we checked them already). The length of the program is the sum of: $n - C * |S_x| + C|S_x|/4$ for $n - |S_x|$ x_i's not in S_x by using the representation of Remark 2.2; $(n/C_1C^3)(|M| + logC_1C^3)$ for the n/C_1C^3 long c.s. by Remark 3.1; less than $(2n/C)logC$ (by a simple calculation), which is $< C|S_x|/4$ by the definition of C, for indices of y_l's (Notice that odd and even y_i's can be specified by relative distances, resp, since we ordered x_{i_k}'s, and x_{j_k}'s.) in above arrangement; $|S_x|log(n/|S_x|) < C|S_x|/8$ for the indices in (1); and several $O(logn)$ long counters; total less than n for large n's. This again implies $K(X) < |X|$, a contradiction. □

By an essentially same argument (Divide X into $n^{1/2}$ blocks.), we have,

Corollary: It requires $n^{1.5}$ time for any 1-tape deterministic machine to accept the following language,

$L'' = \{x_1\$ \dots \$x_k \# y_1 \$ \dots \$ y_l \# (1^i, 1^j) \mid x_i = y_j\}$.

5. A language that requires $\Omega(n^{1.5})$ time for a 1-tape nondeterministic machine

Define L'$=\{x_1\$x_0\$x_2\$x_0 \cdots \$x_k\$x_0\#x_0\$x_1x_2 \cdots x_k \mid x_i$ are strings of 0,1 $\}$.

In the following proof, we will concentrate on explaining the ideas of the proof, the details can be found in the proof of Lemma 1.

Lemma 2: It requires $\Omega(n^{1.5})$ time to accept L' by any 1-tape nondeterministic online machine.

Proof: Suppose an 1-tape M accept L' in time $o(n^{1.5})$ with head h_1 on 1-way input tape and h_2 on the working tape.

We fix a large constant C as in Lemma 1;

Fix a random string X of length n. Equally partition X into $x_0x_1 \cdots x_k$, where $k=n^{1/2}/C$ Now, consider input $Y=x_1\$x_0\$x_2\$x_0 \cdots \$x_k\$x_0\#x_0\$x_1x_2 \cdots x_k$ to M. $|Y|<3n$. M should accept this input Y. Let us fix a *shortest* accepting path P of M on input Y. Now we will show that path P is long.

Consider the k pairs $x_i\$x_0$ in Y. If for half of them while h_1 is reading a pair, h_2 moves in a range larger than n/C^3, then M uses time $O(n^{1.5})$, a contradiction. Thus in the following we will assume that for more than half of above pairs, M maps each of them into some region of size $< n/C^3$. Let S be the set of such pairs. When M reads to # sign, we consider 2 cases:

Case 1: (jammed) Assume that there do not exist two pairs in S s.t. while h_1 reads the two x_0's, the nearest h_2 positions of them are at least n/C^2 apart, i.e. no two x_0's are mapped n/C^2 apart. In this case, a simple counting shows that there are $|S|$ (for all the pairs in S) x_i's mapped into a region R of size $3n/C^2$. (i.e with all their h_2 positions in R.) But then by a similar argument as in Case 1 of Lemma 1, we can construct a short program for X, showing K(X)$<|X|$, a contradiction. One may worry about the nondeterminism here, but notice that the nondeterminism is also defined in the K-complexity, we can simply simulate M nondeterministically in above, making sure that the c.s.'s are matched.

Case 2: (not jammed) There exist 2 pairs, say $x_i\$x_0$ and $x_j\$x_0$, in S s.t. when h_1 reads the 2 x_0's, their nearest h_2 positions are at least n/C^2 apart. Now M goes on to read its input $\#x_0\$x_1 \cdots x_k$. We further partition x_0 into $x_{0,1}x_{0,2} \cdots x_{0,l}$, with $l = n^{1/2}/C^2$. If for half of the $x_{0,i}$'s, while h_1 reads each of them, h_2 goes distance n/C^3, then the total time is $O(n^{1.5})$, a contadiction. So we know that there are $n^{1/2}/2C^2$ $x_{0,i}$'s s.t. while h_1 reads each of them, h_2 goes $< n/C^3$ distance. But by a simple counting, there will be a region R_0 of length $n/4C^2$ s.t. there are a set S_0 of $n^{1/2}/4C^2$ $x_{0,i}$'s mapped to one side of R_0, and one of above two x_0's is mapped to the other side of R_0. As before we search for a shortest c.s in R_0. If the shortest c.s. $> n^{1/2}/C^3$, then M runs too long. Otherwise we can reconstruct the $x_{0,i}$'s in S_0 as in Case 2 of Lemma 1 with the help of the short c.s., other x_i's, for i>0, and $\{x_{0,1},...,x_{0,l}\}$-S_0, contradicting to the randomness of X. We leave the detail construction to the interested readers. □

6. Proof of the Theorems

Proof of Theorem 1: Although accepting L (in section 3) in real time by a 2-stack (or even 2-tape) deterministic machine might be impossible, we can define a new padded language L_p from L, a string

$$x_1\$...\$x_k\#y_1\$...\$y_k\#(1^{i_1},1^{j_1},1^{k_1})(1^{i_2},1^{j_2},1^{k_2})...(1^{i_t},1^{j_t},1^{k_t})$$

is in L_p if: we start from the #'s after x_k and y_k, go back i_1 and j_1 steps, respectively. Then match the next (x,y) pair for k_1 long. If not match or not correct length, reject. Then go back i_2 and j_2 on x and y, resp., and match next k_2 long pair, ... If by the end everything matches exactly we accept this string for L_p. Thus L_p can be accepted in real time by a 2-stack 1-reversal deterministic machine. Actually the program is already given above. And we observe that the same lower bound proof (Lemma 1) for L works as well for L_p. (Notice that the length of each substring there is fixed: C.) Therefore the theorem is proved. □

Proof of Theorem 2: It is not hard to see that L in Lemma 1 of Section 4 is accepted by a 1-tape nondeterministic machine in linear time. And L" in the corollary of Lemma 1 can be easily accepted in real time by a 1-tape nondeterministic machine. □

Proof of Theorem 3: The language L' in Section 4 can be easily accepted by a two tape machine. For 1-reversal bounded 2-stack, we modify L': reverse $x_1x_2...x_k$ following $\#x_0\$$, and reverse this last x_0 in $\#x_0\$$. The modified L' can be accepted be 2 deterministic 1-reversal stacks in linear time as follows: put x_1 in stack1, put next x_0 in stack1 and in stack2, put x_2 in stack2, put next x_0 in stack1 and stack2, and x_3 in stack1, When we read to #, we start matching in an obvious way. To make this process real time we further modify L' by simply putting a $1^{2|x_0|}$ padding after every other reversed x_i. Since all these changes do not hurt our lower bound in Lemma 2, the proof is completed. □

Remark: As a by-product of K-complexity, observe that all these bounds we have obtained are not only for the worst cases, but for average cases as well, because there are many random strings.

The K-complexity approach suffers a serious problem when we talk about higher time complexity, e.g., can we replace our n by a general t(n) for some reasonably constructible function t?

7. Acknowledgement

I am very grateful to Professor J. Hartmanis, C. Bajaj, J. Kadin, L. Longpre, and Y. Yesha for their patience in reading and listening to my proofs and for several very helpful discussions and suggestions.

Addendum: It was brought to the author's attention that recently the related work has been done by W. Maass at Berkeley [M1]. I want to thank Professor P. Fejer, B. Harper, and Professor W. Maass for bringing me the reference [M1].

8. References

[A] S.O. Aanderaa, On k-tape versus (k-1)-tape real time computation, in complexity of computation. R. Karp Ed. (1974) pp 75-96.

[BGW] R. Book, S. Greibach, and B. Wegbreit, Time- and tape-bound Turing acceptors and AFLs, JCSS 4,6 (Dec. 1970) pp. 606-621

[DGPR] P. Duris, Z. Galil, W. Paul, and R. Reischuk, Two nonlinear lower bounds, Proc. 15th ACM STOC (1983), 127-132.

[DG] P. Duris and Z. Galil, Two tapes are better than one for nondeterministic machines, Proc. 14th ACM STOC (1982), pp. 1-7

[G] J. Gill, Computational complexity of probabilistic Turing machines, SIAM J Comp. 6 (1977) pp 675-695

[H] J. Hartmanis, Generalized Kolmogorov complexity and the structure of feasible computations, 24th IEEE FOCS (1983) pp 439-445

[HS] F.C. Hennie and R.E. Stearns, Two tape simulation of multitape Turing machines, J.ACM, 4 (1966) pp533-546.

[HU] J. Hopcroft and J. Ullman, Introduction to automata theory, languages, and computation, Addison-Wesley (1979).

[K] A. Kolmogorov, Three approaches to the quantitative definition of information, Problems of Information Transmission, 1-1, 1-7, Jan-Mar (1965)

[M] W. Maass and A Schorr, Speedup of 1-tape Turing machines by bounded alternation, manuscript 1983

[M1] W. Maass, Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines, Extended abstract. to appear in recent 16th STOC (1984).

[P] W. Paul, Kolmogorov complexity and lower bounds, 2nd International conference on fundamentals of computation theory. (1979)

[P1] W. Paul, On heads versus tapes, Proc. 22nd IEEE FOCS (1981) pp 68-73.

[P2] W. Paul, On-line simulation of k+ 1 tapes by k tapes requires nonlinear time, Proc. 23rd IEEE FOCS (1982) pp 53-56

[PPST] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter, On determinism versus nondeterminism and related problems, Proc. 24th IEEE FOCS (1983) pp. 429-438

[PSS] W. Paul, J. Seiferas, and J. Simon, An information-theoretic approach to time bounds for on-line computations, 12th ACM STOC (1980) pp357-367

[R] M. O. Rabin, Real time computation, Israel J. of Math, Vol. 1, No 4

[Si] M. Sipser, A complexity theoretic approach to randomness, 24th IEEE FOCS (1983) pp 439-445

[V] P. Vitanyi, Real-time simulations of multicounters by oblivious one tape Turing machines, 14th ACM STOC, (1982) pp. 27-36