ACT-29 AUGUST, 1981

COORDINATED SCIENCE LABORATORY

APPLIED COMPUTATION THEORY GROUP

OPTIMAL DYNAMIC EMBEDDING OF TREES INTO ARRAYS

M.C. LOUI

REPORT R-917 UILU-ENG 81-2248

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) OPTIMAL DYNAMIC EMBEDDING OF TREES INTO ARRAYS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER R-917 : ACT-29 (UIUC-ENG 81-2248)
7. AUTHOR(*s*) Michael C. Loui		8. CONTRACT OR GRANT NUMBER(*s*) N00014-79-C-0424
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign 1101 W. Springfield Avenue Urbana, IL 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE August 1981
		13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*)		15. SECURITY CLASS. (*of this report*) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

tree, array, data structure, multidimensional Turing machine, simulation, embedding

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

An optimal method for dynamically embedding trees into arrays is presented. Every multihead tree machine of time complexity $t(n)$ can be simulated on-line by a multihead d-dimensional machine in time $O(t(n)^{1+1/d}/\log t(n))$. An information-theoretic argument gives the worst-case lower bound $\Omega(t(n)^{1+1/d}/\log t(n))$ on the time required.

DD FORM 1 JAN 73 1473

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

OPTIMAL DYNAMIC EMBEDDING OF TREES INTO ARRAYS

Michael C. Loui

Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

August 1981

Abstract

An optimal method for dynamically embedding trees into arrays is presented. Every multihead tree machine of time complexity $t(n)$ can be simulated on-line by a multihead d-dimensional machine in time $O(t(n)^{1+1/d}/\log t(n))$. An information-theoretic argument gives the worst-case lower bound $\Omega(t(n)^{1+1/d}/\log t(n))$ on the time required.

Key Words: tree, array, data structure, multidimensional Turing machine, simulation, embedding.

Supported by the Joint Services Electronics Program (U.S. Army, U.S. Navy, U.S. Air Force) under Contract N00014-79-C-0424.

1. Introduction

Several researchers [2,5,12,13,14] have investigated static embeddings among data structures. To model the physical implementation of a logical data structure, they fix a correspondence between locations in the guest (logical) structure and locations in the host (physical) structure and analyze the effect of various correspondences on access costs. Lynch's work on accessibility of values in algebras [8] gives lower bounds on these access costs. When few cells of the guest structure are used during a computation, these fixed embeddings waste space in the host structure.

In contrast, we have developed dynamic embeddings that designate representatives in the host only for cells of the guest that are actually used [6,7]. To model storage and retrieval operations in data structures, we have used generalizations of Turing machines: tree machines for trees, multi-dimensional machines for arrays. The access heads of the machines correspond to access pointers into the data structures.

Continuing the study of dynamic embeddings, we present an optimal on-line simulation of a multihead tree machine of time complexity $t(n)$ by a multihead d-dimensional machine in time $O(t(n)^{1+1/d}/\log t(n))$. To establish the lower bound $\Omega(t(n)^{1+1/d}/\log t(n))$ on the time required, we employ the information-theoretic technique developed by Paul, Seiferas, and Simon [9].

Previously, Pippenger and Fischer [10] proved that every tree machine of time complexity $t(n)$ can be simulated on-line by a one-dimensional machine in time $O(t(n)^2/\log t(n))$, an optimal amount of time. For $d \geq 2$, a theorem of Grigoriev [3] implies a simulation by a d-dimensional machine in time $O(t(n)^{1+1/(d-1)})$.

Conversely, Reischuk [11] devised on-line simulations of d-dimensional machines of time complexity $t(n)$ by tree machines in time $O(t(n)(5^d)^{\log^* t(n)})$. It is not known whether Reischuk's simulation is optimal.

2. Definitions

Cook and Aanderaa [1] introduced the bounded activity machine, a generalization of the Turing machine model. A bounded activity machine has a finite-state control, a read-only linear input tape (from which it reads input symbols), a write-only linear output tape (on which it writes output symbols), and a finite number of storage media, each of which has a finite number of access heads. A storage medium is an infinite set of cells, each of which can hold a symbol from a finite storage alphabet. One cell is designated the origin of the storage medium. Each access head is located at a cell of its storage medium and can execute a finite number of shifts to other cells of its storage medium. Cell X is at distance s from cell Y if s is the minimum number of shifts required for an access head on X to travel to Y.

A bounded activity machine operates in a sequence of steps. At each step the machine reads the symbols in the cells on which the input head and access heads are positioned, writes symbols on the storage cells and possibly on the output tape too, and shifts some of its heads. We may assume that the machine can detect when two heads are located at the same cell and hence that the two heads do not attempt to write different symbols in this cell. Initially, all storage cells hold a particular blank symbol, and every access head is positioned on the origin of its storage medium.

The time complexity of the machine is a function t(n) defined for each n to be the maximum over all input strings w of length n of the time (number of steps) that the machine spends on w.

A tree worktape is a storage medium whose cells are organized into a complete infinite binary tree. The root has two children, and all other cells have two children and a parent. An access head can shift from a cell to its parent or to one of its children. The root of the tree worktape is its origin. A tree machine is a bounded activity machine whose storage media are all tree worktapes. Let W be a tree worktape. We fix a natural bijection between strings in $\{0,1\}^*$, called locations, and cells of W; we write W(b) for the cell at location b. Let λ denote the empty string. The root of W is $W(\lambda)$. In general, W(b0) is the left child of W(b), and W(b1) is its right child. Let W[b,r] denote the complete subtree of W of height r rooted at W(b). The leaves of W[b,r] are at distance r from W(b).

A d-dimensional worktape is a storage medium whose cells correspond bijectively with d-tuples of integers, called the coordinates. An access head can shift from a cell to another whose coordinates differ in just one component by $\pm$ 1. The origin of the worktape is the cell whose coordinates are all zero. The location of a cell is a binary string (of a fixed format) that specifies its coordinates. The location of a cell at distance s from the origin has length proportional to log s. The binary encoding of coordinates permits a machine to employ the location of cell X to send an access head from the origin to X in time proportional to s. A box is a set of cells that form a d-dimensional cube. The volume of a box is the number of cells in it. The base cell of a box is the cell whose coordinates are the smallest. The location of a box is the location of its base cell. A d-dimensional machine is a bounded activity machine whose storage media are all d-dimensional worktapes.

Fix integers $d \geq 1$ and $h \geq 1$. Let T be a tree machine of time complexity t(n) with just one worktape W with h access heads. Let Δ be the storage alphabet of T. We may assume that whenever a head of T visits a storage cell, it writes a nonblank symbol in the cell. We devise an on-line simulation of T by a d-dimensional machine S in time $O(t(n)^{1+1/d}/\log t(n))$:

If $s_1 \leq s_2 \leq \ldots$ are the steps at which T shifts its input head or output head, then there are corresponding steps $s'_1 \leq s'_2 \leq \ldots$ at which the input and output heads of S perform the same shifts, reading and writing the same symbols, and $s'_i \leq s_i^{1+1/d} \log s_i$ for every i. (All logarithms are taken to base 2.) Since every tree machine with a total of h access heads on several tree worktapes can be simulated in real time by a tree machine with h heads on one worktape with a larger storage alphabet, it suffices to consider only T.

3. Simulation

On a particular input string of length n, suppose T runs for $N' \leq t(n)$ steps. To simulate this computation, S repeats the simulation described in this section for $N = 1,2,4,8,\ldots$, using N for the length of the computation of T, until $N' \geq N$. When $N > 1$, an output manager prevents repetitious outputs by writing only the output symbols generated during steps $(N/2+1)$ through N of T. In addition, S marks all cells that its worktape heads visit. Before embarking on the simulation with the next value of N, it erases its worktapes via a depth-first traversal of the marked cells. We shall show that for each N, the simulation with that value of N takes time $t_0(N) = O(N^{1+1/d}/\log N)$. It follows that S simulates T on-line in time

$$t_0(1) + t_0(2) + \ldots + t_0(2^{\lceil \log N' \rceil}) = O(t(n)^{1+1/d}/\log t(n)).$$

Choose r so that

$$(\log N)/4d \leq r \leq (\log N)/2d.$$

By definition, $r\,4^r \leq N^{1/d}$.

Cover W with overlapping trees W[b,2r+1] such that the length of b is a multiple of $r+1$. Call these trees <u>blocks</u>. Every block has $2^{2r+2} - 1 \leq 4^{r+1}$ cells. Call W[b,r] the <u>upper half</u> of block W[b,2r+1] and the remainder of the block its <u>lower half</u>. By definition, if cell X is at distance at least $r+1$ from the root, then X belongs to exactly two blocks; X is in the upper half of one block and in the lower half of another. The <u>immediate ancestor block</u> of $W[b,2r+1]$ is the block $W[a,2r+1]$ such that W(a) is the ancestor of W(b) at distance $r+1$ from W(b); equivalently, the location a is the initial segment of b for which $|b| = |a| + r + 1$. An <u>immediate descendant block</u> of $W[b,2r+1]$ is a block $W[bc,2r+1]$ such that W(bc) is a descendant of W(b) at distance $r+1$ from W(b); equivalently, $|c| = r+1$.

Define the function τ by

$$\tau(z) = 2^{\lceil \log z \rceil};$$

if z is not a power of 2, then τ maps z to the next larger power of 2. Define k',u, and the function π by

$$k' = 2^{d+1}[(3+4dk)(2h)+24\ dhk^2],$$
$$\pi(m) = \tau((3+4dk)m+3k \log N),$$
$$u = \tau(k'N+(\pi(4^{r+1}))^d),$$

where the constant k is chosen so that the location of every cell at distance at most du from the origin has length at most k log N. Observe that the volume of a box of side $\pi(m)$ is at least $(3+4dk)m+3k \log N$, and that $u = O(N^{1/d})$.

To maintain the simulated contents of blocks, S has a box of side u whose base cell is the origin called the _mass store_. By definition of u, the location of every cell in the mass store has length at most k log N. The mass store contains _pages_. A page P is box whose side is a power of 2 and whose contents are organized into a _path string_, an _Ancestor-pointer_, a _First-pointer_, and a _Current-pointer_. We assume a standard format for pages such that a page of volume $L+3k \log N$ can accommodate a path string of length L in addition to the Ancestor-pointer, First-pointer, and Current-pointer, which together occupy at most 3k log N cells.

We describe how the path string of P represents the contents of a block $B = W[b,2r+1]$. The path string has symbols from three disjoint alphabets: Δ (the storage alphabet of T); three _shift symbols_ for the shifts on the tree worktape; and _location symbols_ to specify locations on the d-dimensional worktape. A symbol occurring in the path string <u>visits</u> cell X of W if a head that starts at W(b) and shifts according to the shift symbols preceding that occurrence arrives at X. The path string <u>represents</u> the contents of the lower half of B if the following two conditions hold. First, for every nonblank cell X in the <u>lower half</u> of B, the path string has <u>exactly one</u> occurrence of a symbol in Δ that visits X, and X contains this symbol. Second, for every binary string x there is at most one nonnull substring of contiguous location symbols that visit W(bx); this substring has length at most k log N, and P has a <u>Descendant-pointer</u> whose value Descendant (P,x) is this string of location symbols.

The Ancestor-pointer, First-pointer, and Current-pointer specify locations of pages in the mass store. Write Ancestor (P), First (P), and Current (P) for the values of these pointers. For convenience, we identify a page with its location. For instance, Current (First (P)) is the value of the Current-pointer of the page at location First (P). As the simulated contents of B change, different pages of S will correspond to B; First (P) is the first page that corresponds to B during the simulation.

For configurations of S we define a <u>correspondence</u> between pages and blocks.

(i) There is exactly one page marked with a special symbol. This page corresponds to $W[\lambda,2r+1]$.

(ii) Let page P correspond to $W[b,2r+1]$. If Descendant (P,c) is nonnull, then Current(First(Descendant (P,c))) corresponds to $W[bc,2r+1]$.

Throughout the simulation, the following will be true for a page P that corresponds to a block B.

Invariant 1. The path string of P represents the contents of the lower half of B.

Invariant 2. If the path string of P visits v distinct cells of B, then the path string has at most 2v shift symbols and at most v Δ symbols. The cells of B visited by the path string are nonblank.

Invariant 3. The page Current(First(Ancestor (P))) corresponds to the immediate ancestor block of B. If $B' = W[bc, 2r+1]$ is an immediate descendant block of B and the lower half of B' is nonblank, then Current(First(Descendant(P,c))) corresponds to B'.

Invariant 4. If $B = W[\lambda, 2r+1]$, then for every nonblank cell X in B (not just in the lower half), the path string of P has exactly one occurrence of a symbol in Δ that visits X, and X contains this symbol.

Let $H_1, \ldots, H_h$ be the h access heads of T. To maintain the simulated access head locations, S has several head location tapes named L_{i1}, L_{i2} for $i = 1, \ldots, h$; they are used as linear tapes. On tape L_{i1} the location of H_i is written and marked in consecutive contiguous segments; all segments preceding the last have length $r+1$, and the last has length between 0 and r. Tape L_{i2} is used as a unary counter with values 0 to $2r+1$.

Suppose H_i is located at cell $X = W(bx)$ in a block B rooted at W(b). Let page P correspond to B. The value of the contents of L_{i2} indicates whether H_i is in the upper half or the lower half of B. Using the head location tapes, S can determine whether a symbol in the path string of P visits X. Depending on the value of L_{i2}, S copies the last one or two segments on L_{i1} to an auxiliary tape. While one head scans along the path string, S uses the information on this auxiliary tape to decide in a routine fashion which symbols of the path string visit X. If X is in the lower half of B, then S can retrieve the symbol that X contains and also the value Decendant (B,x).

To record the nonblank symbol δ that H_i writes on X, machine S first determines the ancestor Y closest to X that the path string visits in B. Let Y be at distance $s \geq 0$ from X. Then S produces a new path string that differs from the old one only by the insertion of a string of $2s+1$ consecutive symbols: s shift symbols for the shifts from Y to X, the symbol δ (which visits X), and s shift symbols for the shifts from X to Y. If the original path string visited v distinct cells of B and had at most 2v shift symbols, then the new path string visits $v+s$ cells, including X, and has at most $2v+2s = 2(v+s)$ shift symbols. Throughout the computation of T, the nonblank cells of W form a connected set. Thus, if X and Y are nonblank, then all the $s-1$ cells between Y and X are also nonblank. Consequently, if the old path string visited only nonblank cells of B, then so does the new path string. We conclude that S can maintain Invariant 2.

The simulator S has heads $G_{u1},\ldots,G_{uh}$, $G_{\ell 1},\ldots,G_{\ell h}$ on its mass store. The simulation begins with all these heads on a page P_0 of side $\pi(r)$ that corresponds to $W[\lambda,2r+1]$. Initially, Ancestor (P_0) = First (P_0) = Current $(P_0) = P_0$, and the path string of P_0 is empty.

In general, to simulate head H_i on cell X_i in block $B_i = W[b_i,2r+1]$, head G_{ui} is in page P_i and head $G_{\ell i}$ in page Q_i such that Q_i corresponds to B_i and P_i = Current(First(Ancestor(Q_i))) corresponds to the immediate ancestor block A_i of B_i. If X_i is in the lower half of B_i, then S uses the path string of Q_i read by $G_{\ell i}$ to retrieve the contents of X_i. If the path string of Q_i does not visit X_i, then X_i holds a blank symbol. If X_i is in the upper half of B_i and $b_i \neq \lambda$, then X_i is in the lower half of A_i, and S retrieves the contents of X_i from the path string of P_i read by G_{ui}. (If $b_i = \lambda$, then according to Invariant 4, the path string of Q_i has the contents of X_i even when X_i is in the top half of B_i.) To simulate the effect of one step of T, machine S records the new contents of each X_i in the appropriate path string (of P_i or of Q_i). If T shifts its input head or writes an output symbol at the end of this step, then S does the same. Finally, S updates the head location tapes. When S completes the simulation of this step, Invariant 1 holds: the path string of P_i represents the contents of the lower half of A_i, and the path string of Q_i represents the contents of the lower half of B_i.

If H_i shifts from $W(b_i)$ to its parent, then S performs an upward reorientation on heads G_{ui} and $G_{\ell i}$. Let M = Current(First(Ancestor(P_i))). The upward reorientation consists of sending $G_{\ell i}$ to P_i and G_{ui} to M, which corresponds to the immediate ancestor block of A_i, in whose lower half the parent of $W(b_i)$ is located.

If H_i shifts to a child of a leaf $W(b_i cx)$ of B_i, where $|c| = r+1$, then S performs a downward reorientation of G_{ui} and $G_{\ell i}$ by sending G_{ui} to Q_i ang $G_{\ell i}$ to R = Current(First(Descendant(Q_i,c))), provided that Q_i has a Descendant(Q_i,c) pointer. If R is not defined, then the bottom half of $W[b_i c, 2r+1]$ is completely blank (by Invariant 3), although r cells in its top half (on a path from $W(b_i c)$ to $W(b_i cx)$) must be nonblank. The storage allocation procedure ALLOCATE (described in Section 5) produces a new completely blank page R of side $\pi(r)$ in the mass store. Next, on this new page S sets First (R) $\leftarrow$ R, Current (R) $\leftarrow$ R, Ancestor (R) $\leftarrow$ Q. Also, S sets Descendant $(Q_i, c) \leftarrow R$ and sends $G_{\ell i}$ to R. After this initialization, R corresponds to $W[b_i c, 2r+1]$.

Now suppose that when S adds further symbols to the path string of a page P to record the contents of a cell in the corresponding block B, P is not large enough to contain the updated path string. With a call to ALLOCATE, S finds a new unused box P' in the mass store whose side is a power of 2 such that P' is just large enough to hold the new path string (as well as the Ancestor-pointer, First-pointer, and Current-pointer). Then S writes the updated path string into P' and sets First (P') $\leftarrow$ First (P), Current (First (P)) $\leftarrow$ P', and Ancestor (P') $\leftarrow$ Ancestor (P). Page P' now corresponds to B.

4. Analysis of the Simulation

First, we establish upper bounds on the volumes of pages. Throughout this section we may assume that at least r cells of $W[\lambda,2r+1]$ are nonblank.

Lemma 1. In some configuration of S during the simulation, let page P correspond to simulated block $B = W[b,2r+1]$ with m nonblank cells. Then the length of the path string of P is at most $(3+4dk)m$, and the side of P is at most $\pi(m)$.

Proof. Invariant 2 guarantees that the path string of P has at most 2m shift symbols and at most m symbols in Δ. If P has a Descendant (P,c) pointer, then by Invariant 3, the lower half of block $W[bc,2r+1]$ is nonblank; since the nonblank cells of W always form a connected set, at least r cells of $W[bc,r]$ in the lower half of B are nonblank. The number of location symbols in Descendant (P,c) is at most $k \log N \leq 4\,dkr$. Consequently, the total number of location symbols in the path string of P is bounded above by 4 dk times the number of nonblank cells in the lower half of B, which is at most m. We have deduced that the length of the path string is at most $2m+m+4\,dkm = (3+4\,dk)m$.

The Ancestor-pointer, First-pointer, and Current-pointer together occupy at most $3\,k \log N$ cells. Therefore, the volume of P is at most

$$(3+4\,dk)m + 3\,k \log N \leq (\pi(m))^d,$$

and the side of P is at most $\pi(m)$. □

Lemma 2. Throughout the simulation, the total volume of pages in the mass store is at most $k'N$.

Proof. At an arbitrary configuration of the simulation, let $P_1, P_2, \ldots$ be the pages that correspond to blocks on W. Let P_j correspond to block B_j, and let B_j have m_j nonblank cells. Since every cell of W belongs to at most two blocks and since the access heads of T can visit at most hN cells of W,

$$\Sigma_j\, m_j \le 2\ hN.$$

Because a page is allocated only when the corresponding block has at least r nonblank cells, every $m_j \ge r$, and hence there are at most 2 hN/r pages P_j. The mass store holds smaller pages that correspond to B_j in previous configurations of S. The volumes of these smaller pages are distinct powers of 2. It follows that the total volume of pages that have ever corresponded to B_j is at most twice the volume of P_j. Ergo, the total volume of all pages in the mass store is

$$\Sigma_j\ 2(\pi(m_j))^d \le 2\cdot 2^d\ \Sigma_j((3+4\ dk)m_j + 3k\ \log N) \le 2^{d+1}[(3+4\ dk)(2hN) + (2hN/r)(3k\ \log N)] \le k'N$$

by Lemma 1 and definition of r and k'. □

The time used by S to simulate one step of T is proportional to the length of the h path strings that it handles. Lemma 1 implies that every path string has length at most $O(4^r)$. Thus, to simulate N individual steps, S spends time $O(N\ 4^r) = O(N^{1+1/d}/\log N)$ updating path strings and head location tapes.

After an upward or a downward reorientation of G_{ui} and $G_{\ell i}$, the simulated head H_i is at distance r or r + 1 from both the root and the leaves of a block. Consequently, this head can induce at most N/r reorientations. Each upward reorientation takes time O(log N) to retrieve the location of another page and time $O(u) = O(N^{1/d})$ to move the heads across the mass store. For a downward reorientation, S may spend, in addition, time $O((\log N)^2)$ for a call to ALLOCATE (as we show in Section 5). Therefore, the total time for reorientations is at most

$$(N/r)O((\log N)^2 + N^{1/d}) = O(N^{1+1/d}/\log N).$$

When S copies the contents of a page P to a larger page P', it spends time $O(u) = O(N^{1/d})$ to move the heads across the mass store, time $O((\log N)^2)$ for a call to ALLOCATE, and time $O(4^r + \log N)$ to copy the path string and pointer values. Since every page has volume at least r, Lemma 2 implies that S makes at most $O(N/r)$ allocations of pages. Thus, S spends time

$$O((N/r)(N^{1/d} + 4^r + (\log N)^2)) = O(N^{1+1/d}/\log N)$$

finding and initializing new pages.

In summary, the simulator uses time $O(N^{1+1/d}/\log N)$ to simulate N individual steps, time $O(N^{1+1/d}/\log N)$ to reorient heads G_{ui} and $G_{\ell i}$, and time $O(N^{1+1/d}/\log N)$ to prepare new larger pages in the mass store.

<u>Theorem 1</u>. Every multihead tree machine of time complexity $t(n)$ can be simulated on-line by a multihead d-dimensional machine in time $O(t(n)^{1+1/d}/\log t(n))$.

5. Storage Allocation

Machine S has a free storage list, a list of locations of blank boxes in the mass store. For $q = 1,2,\ldots,u/2,u$, this list has locations of at most 2^d-1 boxes of side q. Initially, the free storage list holds the location of the mass store itself, a single box of side u.

Procedure ALLOCATE employs a buddy system [4] to allocate a blank box with a desired side in the mass store. To obtain a blank box of side p, a power of 2, ALLOCATE finds the location of a box of side p on the free storage list. If the free storage list has no boxes of side p, then let q^* be the smallest power of 2 for which the free storage list has a box of side q^*. (We shall show that when ALLOCATE is called during the simulation, q^* must exist.) For $q = q^*,q^*/2,\ldots,4r,2r$ in order, select the location of a box Q_q of side q and delete this location from the list; add to the list the locations of the 2^d disjoint boxes of side $q/2$ whose union is Q_q. Finally, let y be the location of a box of side p on the free storage list, delete y from the list, and return the value y. The time taken by ALLOCATE is $O((\log u)(k \log N)) = O((\log N)^2)$.

Let $q_1 \leq q_2 \leq \ldots \leq q_s$ be the sides of boxes whose locations are on the free storage list when ALLOCATE is called to produce a blank box of side $\pi(m)$, where $m \leq 4^{r+1}$. Lemma 2 and the definition of u imply that

$$q_s^d + \ldots + q_1^d \geq u^d - k'N \geq (\pi(4^{r+1}))^d \ .$$

Since the free storage list has at most 2^d+1 boxes of each distinct side,

$$q_s^d + \ldots + q_1^d \le (2^d - 1)q_s^d + (2^d - 1)(q_s/2)^d + \ldots + (2^d - 1)(1) < (2q_s)^d .$$

Thus, $(2q_s)^d > (\pi(4^{r+1}))^d$, hence $q_s \ge \pi(4^{r+1})$ because q_s is a power of 2. It follows that ALLOCATE can find a box of side $\pi(m)$ in the mess store.

6. Lower Bound

We define a tree machine T' and demonstrate that every d-dimensional machine that simulates T' on-line requires time $\Omega(N^{1+1/d}/\log N)$.

Machine T' has just one access head on one tree worktape and operates in real time. Its input alphabet is a set of commands of the form $\langle e,\sigma\rangle$, where $e \in \{0,1,?\}$ and σ is a shift for a tree worktape. Suppose T' is in a configuration in which the cell X at which the access head is located contains e'. On input $\langle e,\sigma\rangle$, machine T' writes e' on its output tape, and the access head writes e on X if $e \in \{0,1\}$, but writes e' on X (its current contents) if $e = ?$. At the end of the step the access head executes the shift σ.

Let d-dimensional machine S' simulate T' on-line. To establish the lower bound $\Omega(N^{1+1/d}/\log N)$ on the time required by S', we formalize the following volumetric argument. The worktape head of T' can access one of a set of N cells within log N steps, whereas the heads of S' require $\Omega(N^{1/d})$ steps to access one of a set of N cells in the worst case. If the contents of a set of N cells of the worktape of T' are sufficiently random (in a sense made precise below), then there is a sequence of $\Omega(N/\log N)$ "hard questions" about the contents of these cells, each question having length log N, such that S' requires time $\Omega(N^{1/d})$ to answer each question.

Let # be a new symbol. For strings x,y in $\{0,1,\#\}^*$, let $K(x|y)$ be the Kolmogoroff complexity of x given y with respect to a fixed universal function U. Formally, $K(x|y)$ is the length of the shortest binary string b such that $U(b \# y) = x$. Intuitively, b is a binary description of x, given y. Write $K(x)$ for $K(x|\lambda)$, where λ is the empty string. The following elementary properties of K are well known: There is a fixed constant c such that for all x and y,

$$K(x) \le 2|x| + c,$$
$$K(x) \le K(x|y) + K(y) + c.$$

Call a binary string x for which $K(x) \ge |x|$ <u>incompressible</u>. Since there are 2^n binary strings of length n but only 2^n-1 possible shorter binary descriptions, there exists for every n at least one incompressible binary string of length n.

<u>Lemma 3</u>. Let $h \ge 1$ and let x be an incompressible string of length $N > 8(c+h)$. For every set of h strings $\{y_1,\dots,y_h\}$ of length at most $N/4h$ each, $K(x|y_1\# \dots \#y_h) > N/4$.

<u>Proof</u>. If not, then

$$K(x) \le K(x|y_1\# \dots \#y_h) + K(y_1\# \dots \#y_h) \le N/4 + (2h)((N/4h)+1) + 2c < N.$$

Contradiction. □

<u>Theorem 2</u>. Let d-dimensional machine S' with head-to-head jumps on one worktape simulate T' on-line in time $t'(N)$. Then $t'(N) = \Omega(N^{1+1/d}/\log N)$.

<u>Proof</u>. For every N that is a sufficiently large power of 2, we construct a string of N input commands on which S' requires time $\Omega(N^{1+1/d}/\log N)$. The input string has a filling part Q_0 of length N/2 followed by a query part of length N/2.

Let W be the worktape of T'. The filling part compels the head of T' to write on the $(N/4) - 1$ cells of $W[\lambda,\log(N/8)]$ such that a depth-first traversal of the contents of $W[\lambda,\log(N/8)]$ gives an incompressible string x of length $(N/4) - 1$.

The query part is a sequence of $N/(4 \log N)$ <u>questions</u> $Q_1, Q_2, \ldots$. Each question is a string of $2 \log N$ commands of the form $\langle ?, \sigma \rangle$ that drives the head of T' from the root $W(\lambda)$ to a cell of $W[\lambda, \log(N/8)]$ and back to the root. Note that the contents of $W[\lambda, \log(N/8)]$ remain unchanged when T' processes a question. We choose the questions so that S' spends time $\Omega(N^{1/d})$ to process each Q_j.

Let S' have h access heads. For $j \geq 0$, consider the configuration of S' after it has processed Q_j. Let $B_1, \ldots, B_h$ be the boxes of side $(N/(32\ c'h))^{1/d}$ centered at the heads in this configuration, where the constant c' depends on S' and is specified later. These boxes hold all the cells accessed by S' during the next $(N/(32\ c'h))^{1/d}/2$ steps. We claim that for some Q_{j+1}, some head of S' must exit $B_1 \cup \ldots \cup B_h$ when S' processes Q_{j+1}. Otherwise, let y_i be a binary encoding of the contents of B_i and z_i be a binary encoding of the relative position of access head i in B_i. Evidently, if c' is sufficiently large, then both $|y_i| \leq c'|B_i|$ and $|z_i| \leq c'|B_i|$ for every i. From the string $y_1 \# \ldots \# y_h \# z_1 \# \ldots \# z_h$, only a small constant amount of additional information (essentially a binary description of this discussion) is necessary to generate x because S' can process every question Q_{j+1} with the heads remaining in $B_1 \cup \ldots \cup B_h$. We deduce that $K(y_1 \# \ldots \# y_h \# z_1 \# \ldots \# z_h) = O(1)$, contravening Lemma 4.

Therefore, since some head spends time $(N/(32\ c'h))^{1/d}/2$ to exit $B_1 \cup \ldots \cup B_h$ when S' processes question Q_{j+1}, the time spent by S' on the query part alone is at least

$$(N/(4 \log N))(N/(32\ c'h))^{1/d}/2 = \Omega(N^{1+1/d}/\log N). \qquad \square$$

References

[1] S. A. Cook and S. O. Aanderaa. On the minimum computation time of functions. Trans. Am. Math. Soc. 142 (1969) 291-314.

[2] R. A. De Millo, S. C. Eisenstat, and R. J. Lipton. Space-time tradeoffs in structured programming: An improved combinatorial embedding theorem. J. ACM 27 (1980) 123-127.

[3] D. Ju. Grigoriev. Imbedding theorems for Turing machines of different dimensions and Kolmogorov's algorithms. Soviet Math. Dokl. 18 (1977) 588-592.

[4] D. E. Knuth. The Art of Computer Programming, vol. 1: Fundamental Algorithms. Addison-Wesley, 1968.

[5] R. J. Lipton, S. C. Eisenstat, and R. A. De Millo. Space and time hierarchies for classes of control structures and data structures. J. ACM 23 (1976) 720-732.

[6] M. C. Loui. Simulations among multidimensional Turing machines, Tech. Rep. TR-242, Lab. for Comp. Sci., M.I.T., Aug. 1980. To appear in Proc. 22nd Ann. Symp. on Foundations of Computer Science, IEEE, 1981. To appear in Theor. Comp. Sci.

[7] M. C. Loui. Minimizing access pointers into trees and arrays. Rep. R-910, ACT-27, Coordinated Sci. Lab., Univ. of Illinois, June 1981.

[8] N. A. Lynch. Accessibility of values as a determinant of relative complexity in algebras. To appear in J. Comp. Sys. Sci.

[9] W. J. Paul, J. I Seiferas, and J. Simon. An information-theoretic approach to time bounds for on-line computation. Proc. 12th Ann. ACM Symp. on Theory of Computing, 1980, pp. 357-367. To appear in J. Comp. Sys. Sci.

[10] N. Pippenger and M. J. Fischer. Relations among complexity measures. J. ACM 26 (1979) 361-381.

[11] R. Reischuk. A fast implementation of a multidimensional storage into a tree storage. Proc. 7th Intern. Colloq. on Automata, Languages and Programming, Springer-Verlag, 1980, pp. 531-542. To appear in Theor. Comp. Sci.

[12] A. L. Rosenberg. Data encodings and their costs. Acta Informatica 9 (1978) 273-292.

[13] A. L. Rosenberg. Encoding data structures in trees. J. ACM 26 (1979) 668-689.

[14] A. L. Rosenberg and L. Snyder. Bounds on the costs of data encodings. Math. Sys. Th. 12 (1978) 9-39.