Acta Informatica 9, 273–292 (1978)

Acta Informatica

© by Springer-Verlag 1978

Data Encodings and Their Costs

Arnold L. Rosenberg

Mathematical Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

Summary. This paper is devoted to developing and studying a precise notion of the "encoding" of a "logical data structure" in a "physical storage structure," that is motivated by considerations of computational efficiency. The development builds upon the notion of an encoding of one graph in another. The cost of such an encoding is then defined so as to reflect the structural compatibility of the two graphs, the (externally specified) costs of "implementing" the host graph, and the (externally specified) set of intended "usage patterns" of the guest graph. The stability of the constructed framework is demonstrated in terms of a number of results; the faithfulness of the formalism is argued in terms of a number of examples from the literature; and the tractability of the model is hinted at by several results and by further references to the literature.

1. Introduction

It is not uncommon for an algorithm that is optimally efficient in an idealized environment to have to be completely retailored in order to conform to the requirements of a given programming language or computing system. Perhaps the most painful (and ubiquitous) phase of this adaptation process is the re-outfitting of the algorithm with data structures that is often referred to as the translation from "logical data structures" to "physical storage structures." Since the transformed algorithm is (barring errors) behaviorally equivalent to the original, there must be compatibility between the "guest" data structures of the original algorithm and the "hast" data structures of the transformed one. Of no less significance is the extent to which the source and target data structures match in terms of computational efficiency: one obviously has no desire to optimize an algorithm in an idealized environment, only to have the gains in efficiency evaporate in the course of implementation. One's chances of devising an efficient algorithm and transforming it into an efficient program are clearly enhanced if one understands the issues influencing the compatibility of the guest and host

0001-5903/78/0009/0273/$04.00

data structures. It is our aim in this paper to propose and study a notion of data encoding that tries to capture at least certain aspects of this compatibility.

In order to motivate the various features of the framework of our investigation, let us discuss briefly a number of data encodings described in the literature.

(1.1) In Section 2.3.2 of [12], Knuth describes a technique for encoding an arbitrary tree in a binary tree. The main benefit of such an encoding is that the two-way "switches" at the nodes of a binary tree are rather easier to store in a two-pointer-per-word computer such as MIX than are more-than-two-way switches.
(1.2) Of course not all data structures are trees. Accordingly, Pfaltz [17] has characterized those graphs that can be encoded in outdegree-two graphs using the obvious generalization of encoding (1.1). (Of course any graph can be encoded in an outdegree-two graph using other encoding techniques.)
(1.3) In Section 6.2.3 of [13], Knuth discusses a variety of encodings of ordered sets in trees, that enhance the efficiency of various operations on the sets. Notable among the storage structures discussed are so-called B-trees, a paging-oriented variety of balanced search tree whose structure is designed to permit retrieval/insertion/deletion of stored keys without visiting too many nodes of the tree so as to keep low the danger of page faults engendered by edge traversals.
(1.4) Standish [25, Section V] describes encodings of both queues and finite sets in the leaves of trees, as examples of the kind of automatically/semi-automatically generated computer-palatable data representation that needs to be studied in order to render feasible the "abstract" specification of data types.
(1.5) DeMillo, Eisenstat, and Lipton [2] describe an efficient encoding of arrays in the leaves of trees; Harper [8, 9] characterizes all optimally efficient encodings of cubes in lines; Iordansk'ii [11] derives lower bounds on the efficiency of encodings of trees in lines. In all three cases, the measure of efficiency is the average "dilation" of the edges of the guest structure.
(1.6) Gotlieb and Tompa [6] describe a somewhat complicated procedure for choosing an efficient storage-encoding of a given data structure, given the intended use of the structure. Their most detailed example involves a depth-first search algorithm.
(1.7) In Section 2.3.3 of [12], Knuth discusses encodings of trees in lines and in ring structures of various degrees of complexity. Each of the encodings described is intended to enhance ease of element-access and ease of structure-traversal simultaneously, given a particular class of intended uses of the trees.

A detailed look at the encodings in (1.1)–(1.7) renders compelling the conclusion that there is a general notion of data encoding under whose aegis all the described encodings lie. It is our purpose to develop such a notion in the sections to follow. A number of features of the cited examples will serve as beacons in our quest for a general model.

1) The aim of many data encodings is to accommodate the guest data structure to the exigencies of the architecture and/or memory layout of the host environment.

2) The general notion of encoding should reflect such exigencies and permit one to compare the relative merits of competing candidate host structures and encodings; that is, the notion of encoding should be accompanied by a notion of the *cost* of an encoding.

3) The proposed notion of the cost of a data encoding must account for the intended layout in storage of the host structure. For example, traversal-moves that cross page boundaries (such as those between nodes of B-trees) must be assessed a higher cost than moves that stay within a page, hence within main memory (such as those within nodes of B-trees).

4) The proposed notion of the cost of a data encoding must account for the intended patterns of using the guest structure. The "optimal" storage structure selected by Gotlieb and Tompa's procedure (1.6) for a depth-first search algorithm could turn out to be pessimal for a breadth-first search algorithm.

The two features that characterize our investigation and distinguish it from the discursive treatments in [21, 24] are our formulating our notion in a mathematical framework, emphasizing the mathematical implications of our various decisions, and our stressing the quantitative aspects of data encodings. Indeed, after introducing our formal notion in Section 2, we devote Section 3 to uncovering those basic results about the model that suggest its appropriateness, and we dedicate Section 4 to the derivation of bounds on the costs of data encodings.

2. A Framework for Studying Data Encoding

A. Encodings

Many issues concerning data structures are best dealt with in a graph-theoretic framework. Graph-oriented models can depict rather faithfully many of the features of data structures that one infers, from the "more practical" literature such as [12], to be essential to an understanding of data structures; such models have the further benefits of tractability-through-simplicity and a rich literature to drawn on for techniques, inspiration, and results.

The simplest variety of graph that seems to be suitable for the study of data encodings is a finite, directed graph. More elaborate alternatives, which include edge-labels or root nodes, or which posit either the graph's acyclicity or strong connectivity (as but a few examples of embellishments that have appeared in graph models for data structures) contain features that are not germane to the issues we wish to study. On the other hand, further simplifying the model by studying undirected graphs would not allow us to make certain crucial distinctions: consider for a moment the encodings alluded to in (1.1). The motivation for replacing arbitrary-degree graphs by degree-two graphs (generalizing the encoding in (1.1)) is to accommodate one's graph to the fixed word length of the host computer. However, it is obviously only large *outdegrees* of nodes that require such accommodation; graphs such as the trees used in the M. Fischer-Galler equivalence-relation-processing algorithm [12, Section 2.3.3, Algorithm

E], that have outdegree 1 at every node albeit potentially enormous indegrees, obviously require no such encoding. We therefore base our study on the following formalism.

(2.1) A *graph* $G=(V, E)$ consists of

(a) a set V of *vertices* (or *nodes*);
(b) a set $E \subseteq V \times V$ of *edges*.

Remarks. For convenience, we shall write "$v \in G$" instead of "$v \in V$" to assert v's being a vertex of G. Similarly, we shall denote by "$|G|$", rather than by "$|V|$", the cardinality of G's vertex set. (In general, $|S|$ denotes the cardinality of the set S.) We shall call v the *source* and v' the *target* of the edge $\langle v, v' \rangle \in E$. The *outdegree* of $v \in G$ is the number $|\{v\} \times V \cap E|$ of edges with source v; dually, the *indegree* of v is the number $|V \times \{v\} \cap E|$ of edges with target v.

(2.2) A *path* in $G=(V,E)$ is a finite sequence

$$p = \langle v_1, v_2 \rangle \langle v_2, v_3 \rangle \cdots \langle v_n, v_{n+1} \rangle$$

with all $v_i \in G$ and all $\langle v_i, v_{i+1} \rangle \in E$.

Remarks. Given the path p of (2.2), we designate $v_1 = \text{source}(p)$ and $v_{n+1} = \text{target}(p)$; and we denote by $|p|$ the *length* of p ($|p| = n$ in (2.2)). We call p a (*vertex-*)*simple* path if no two edges in p share either source or target. Finally, we denote by $\text{Paths}(G)$ the set of all paths in G.

With these preliminaries out of the way, we are prepared to present our formal notion of data encoding.

(2.3) An *encoding of the graph* $G=(V, E)$ *in the graph* $H=(V', E')$ is an injection (= one-to-one function)

$$\varepsilon: E \to \text{Paths}(H)$$

that induces an injection

$$\iota: V \to V';$$

that is to say, $\text{source}(e) = \text{source}(e') \in V$ if and only if $\text{source}(\varepsilon(e)) = \text{source}(\varepsilon(e')) \in V'$ for all $e, e' \in E$, and similarly for targets.

Figure 1 depicts two simple graphs, a 5-vertex "ring" and a 3×3 "array." (The double-headed arrows in the array abbreviate separate single-headed arrows in opposing directions.) One possible encoding of the ring in the array is given in Table 1. Both the edge-injection ε and the vertex-injection ι are tabulated.

A word about Definition (2.3) is in order. One might think at first blush that the vertex injection ι should be the "encoding" of G in H. However, this injection is not adequately prescriptive. An encoding of one data structure in another should include a rule for "translating" walks in the guest structure into their equivalents (under the encoding) in the host structure; some such path-translation would seem to be prerequisite to assigning a meaningful cost to an

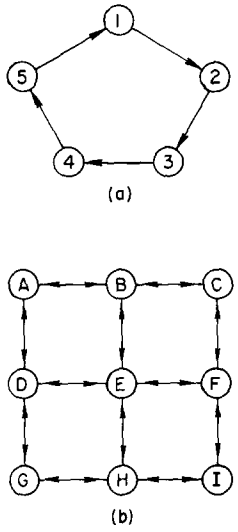

Fig. 1. a A ring graph. **b** An array graph

Table 1. a) An encoding of the ring of Figure 1a in the array of Figure 1b. b) The induced vertex-injection

a)		b)	
$e \in E_{\text{ring}}$	$\varepsilon(e) \in \text{Paths(Array)}$	$v \in V_{\text{ring}}$	$\iota(v) \in V_{\text{array}}$
$\langle 1, 2\rangle$	$\langle G, H\rangle\langle H, I\rangle$	1	G
$\langle 2, 3\rangle$	$\langle I, F\rangle$	2	I
$\langle 3, 4\rangle$	$\langle F, E\rangle\langle E, B\rangle$	3	F
$\langle 4, 5\rangle$	$\langle B, A\rangle$	4	B
$\langle 5, 1\rangle$	$\langle A, D\rangle\langle D, G\rangle$	5	A

encoding; and we feel that an explicit path-correspondence, as ε yields in (2.3), is preferable to any conventional alternative (such as "shortest path" with some tie-breaking mechanism).

B. The Cost of an Encoding

The "cost" of a data encoding should reflect the consumption of resources engendered by executing one's algorithm on the "physical storage structure" H in an actual computer rather than on the "logical data structure" G in an idealized computer. We recall our conclusion in the Introduction that any meaningful assessment of cost must take into account both the costs engendered by the implementation of H, that is, how H is laid out in memory, and the intended "usage patterns" for G. Before discussing each of these factors in turn, we remark that this assessment of cost need not be an *a posteriori* accounting

measure. *A priori* estimates of cost may aid one in accepting or rejecting proposed hosts or, given a host, may aid one in deciding how best to implement the host, all decisions being relative to the intended uses of the guest.

Implementation Costs. Costs incurred because of computer architecture and/or storage layout can be represented faithfully and conveniently by weighting the edges of the host graph.

(2.4) An *edge-weighting* function on the graph $H=(V, E)$ is a function

$$\omega: E \to (\text{positive Reals}).$$

ω is extended to paths in H additively:

$$\omega^+: \text{Paths}(H) \to (\text{positive Reals})$$

by

$$\omega^+(e_1 e_2 \cdots e_n) = \sum_{1 \leqq i \leqq n} \omega(e_i).$$

We shall henceforth identify ω^+ with ω.

We can best explain the intended use of the function ω by illustration. Focus on the array of Figure 1b, which is the host in the encoding of Table 1.

1) Say that the array is stored by "sequential allocation" [12, Section 2.2.6] so that transitions from any node to any other node are easy to effect. One might reflect this ease of transition by having $\omega \equiv 1$ so that the only "penalty" incurred by the encoding is that caused by the "dilation" of G's (unit-length) edges into paths in H. This is the edge-weighting studied in the sources in example (1.5).

2) Say next that the array is stored by sequential allocation but that it is so big that it must be segmented and allocated to more than one page of memory. Such a situation could be modeled by assigning a unit weight $\omega(e)=1$ to those edges whose termini both reside in the same page, and a large weight, say $\omega(e)=1000$, to those edges that cross page boundaries and so whose traversal would cause a page swap. A deeper memory hierarchy would be modeled analogously.

3) Say that the array's rows are stored sequentially, but its columns are stored as linked lists. One might reflect such an implementation by assigning $\omega(e)=1$ to each horizontal edge and, say, $\omega(e)=1.5$ to each vertical edge, in order to reflect the overhead of accessing and following a pointer.

4) Finally, say that the array is stored as an "orthogonal list" [12, Section 2.2.6] so that both rows and columns are linked. If one's host computer accommodates only two pointers per word, one might set

(2.5) $\omega(e) = [\log_2 \text{outdegree source}(e)]$

to represent the cost of implementing an (outdegree source(e))-way switch.

Variations on this theme will readily occur to the reader.

Usage Patterns. Let us say that the ring of Figure 1a, which is the guest in the encoding of Table 1, is always processed by entering at node 1 and proceeding

thence to some other node, all termini being equally likely. This pattern of usage of the graph is described by the following four paths and the fact that they are equally likely.

(2.6)

Path	*Probability*
$\langle 1,2\rangle$	1/4
$\langle 1,2\rangle\langle 2,3\rangle$	1/4
$\langle 1,2\rangle\langle 2,3\rangle\langle 3,4\rangle$	1/4
$\langle 1,2\rangle\langle 2,3\rangle\langle 3,4\rangle\langle 4,5\rangle$	1/4

What one infers from (2.6) is that edge $\langle 5,1\rangle$ of the ring will never be used, that edge $\langle 1,2\rangle$ is four times more likely to be traversed than edge $\langle 4,5\rangle$, and so on. This information can be encapsulated most usefully, for the purposes of cost assessment, in terms of probabilities that reflect the relative frequencies of the guest graph's edges in the anticipated pattern of traversing the guest. The pattern (2.6) translates to the following probabilities.

(2.7)

Edge	*Probability*
$\langle 1,2\rangle$	2/5
$\langle 2,3\rangle$	3/10
$\langle 3,4\rangle$	1/5
$\langle 4,5\rangle$	1/10
$\langle 5,1\rangle$	0

We generalize from this example to the following formalism.

(2.8) A *usage pattern* for the graph $G=(V,E)$ is a function

$$\pi: E \to \{x: 0 \leqq x \leqq 1\}$$

such that

$$\sum_{e \in E} \pi(e) = 1.$$

In fact, a data structure is seldom used in just one way. It is more usual that a computer application will process a data structure by using it according to pattern A part of the time, according to pattern B some other part of the time, and so on. Such composite usage patterns are modeled quite naturally in the framework of (2.8).

(2.9) An *application* of the graph $G=(V,E)$ is a set $\Pi=\{\pi\}$ of usage patterns for G, which is *convex* in the following sense. For any subset $\{\pi_i\} \subseteq \Pi$ and associated positive reals $\{\alpha_i\}$ with $\Sigma\alpha_i = 1$, the function

$$\pi(e) = \sum_i \alpha_i \pi_i(e)$$

is in Π.

The convexity of Π in (2.9) corresponds to allowing G to be shared among the usage patterns in Π.

The Cost of an Encoding. In informal terms, the cost of a data encoding should reflect the amount of additional resource consumption required to effect an "average move" in the host structure rather than in the guest structure. The framework of (2.4), (2.8), and (2.9) affords us a simple expression for this cost. The benefits that accrue from this simplicity are discussed in Section 3.

(2.10) The *cost of the encoding* ε *of the graph* $G=(V,E)$ *in the graph* H, *under the edge-weighting function* ω (for H) *and the application* Π (of G) is the function

$$\mathrm{Cost}(\varepsilon;\Pi;\omega)=\max_{\pi\in\Pi}\sum_{e\in E}\pi(e)\,\omega(\varepsilon(e)).$$

To complete our running example, we note that the cost of the encoding ε of Table 1 under the singleton application π of (2.7) and the edge-weighting function ω of (2.5) is

$$\mathrm{Cost}(\varepsilon;\{\pi\};\omega)=2.5.$$

C. The Cost of "Doing Business"

We would anticipate two uses for the model developed in this section. On the one hand, the model affords one a vehicle for studying data encodings within a mathematical framework. On the other hand, the model affords one a simple vehicle for estimating the costs of specific encodings. With regard to both uses of the model, one must be concerned with the "accuracy" of the model, the extent to which it mirrors reality; these concerns are addressed in Section 3. With regard to the latter use of the model, one must be concerned also with how hard the model is to use; we discuss these latter concerns now.

Estimating Probabilities. There is no algorithm for converting a natural-language description of a program's path on a data structure to a usage pattern in the sense of (2.8). But perhaps a few examples will point the reader in the right direction.

Let our guest be an n-node *line* (=doubly-linked linear list), that is, a graph with vertices $V=\{1,2,\ldots,n\}$ and edges $E=\{\langle i,i+1\rangle,\langle i+1,i\rangle: 1\leqq i\leqq n\}$. We shall use the line as though it were the tape of a Turing machine and describe a number of algorithms using the tape. We shall call each node of the line a tape *square*.

1) Tape-Folding. P. Fischer et al. [3] describe a "tape-folding" algorithm that engenders the following head trajectory on the Turing machine's tape. At stage k, the head proceeds from square 2^{k-1} to square 2^k, returns to square 2^{k-1}, and completes the stage by going to square 2^k; each of these three motions comprises a straight sweep; see Figure 2a. Now, in the described trajectory, each rightgoing (leftgoing) edge of our line is crossed precisely two (resp., one) times.

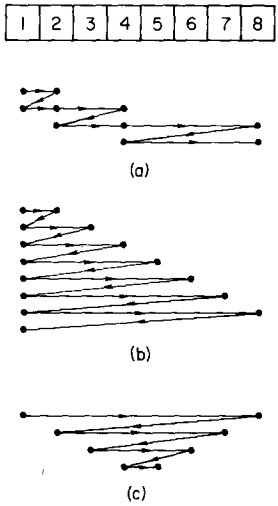

Fig. 2a–c. A schematic view of the head trajectories for **a** tape folding, **b** oblivious computation, and **c** palindrome recognition when $n=8$

Accordingly, after appropriate normalization, we find that for $1 \leqq i < n$,

$$(2.11) \quad \pi_{tf}(\langle i, i+1\rangle) = \frac{2}{3(n-1)} \quad \text{and} \quad \pi_{tf}(\langle i+1, i\rangle) = \frac{1}{3(n-1)}.$$

2) Oblivious Computation. Pippenger and M. Fischer [16] discuss Turing machines whose head trajectories are independent of the input. One possible trajectory for such an "oblivious" machine – albeit not the most efficient one – is to have its head move in stages: at stage k, the head moves from square 1 to square $k+1$ and returns thence to square 1, each motion being a straight sweep; see Figure 2b. After stage $n-1$ (which is as far as one can go on our length n line), the head will have traversed a total of $n(n-1)$ edges. Each edge $\langle i, i+1\rangle$ will have been crossed $n-i$ times, as will its mate $\langle i+1, i\rangle$. Accordingly, for $1 \leqq i < n$,

$$(2.12) \quad \pi_{oc}(\langle i, i+1\rangle) = \pi_{oc}(\langle i+1, i\rangle) = \frac{n-i}{n(n-1)}.$$

3) Palindrome Recognition. Hennie [10] studies the efficiency of one-tape Turing machines that test whether their input is a palindrome, that is a string that reads the same in either direction. He shows that no algorithm on such a machine is materially better than the following naive one. At stage k, the head proceeds from square k to square $n-k+1$, and returns thence to square $k+1$; see Figure 2c. The described trajectory traverses a total of $n(n-1)/2$ edges. Each edge $\langle i, i+1\rangle$ is crossed either i or $n-i$ times, according as $i \leqq n/2$ or not (assume for simplicity that n is even); each edge $\langle i+1, i\rangle$ is traversed $i-1$ or $n-i$ times, according as $i \leqq n/2$ or not. Converting these frequencies to probabilities, we have:

(2.13) for $1 \leqq i \leqq n/2$,

$$\pi_{pr}(\langle i, i+1\rangle) = \frac{i}{\binom{n}{2}} \quad \text{and} \quad \pi_{pr}(\langle i+1, i\rangle) = \frac{i-1}{\binom{n}{2}};$$

for $n/2 < i < n$,

$$\pi_{pr}(\langle i, i+1\rangle) = \pi_{pr}(\langle i+1, i\rangle) = \frac{n-i}{\binom{n}{2}}.$$

Determining Costs of Encodings. Given an encoding ε of a graph G in a graph H, together with a weighting function ω for H and a usage pattern π for G, a patently simple computation will determine $\text{Cost}(\varepsilon; \{\pi\}; \omega)$: one needs only compute the inner product of the "vectors" $\pi(e)$ and $\omega(\varepsilon(e))$.

Computing the cost of ε is only slightly harder if one is presented with any finite set of usage patterns for G from which to generate (via convex combination) an application for G. Indeed, inequality (3.1) of Proposition 3.3 combines with the textbook algorithm for inner products and the obvious algorithm for computing the maximum of n numbers to establish the following.

Let $\mathfrak{C}(\pi_1, \ldots, \pi_n)$ denote the convex closure of the usage patterns $\pi_1, \ldots, \pi_n$.

Proposition 2.1. *Given the encoding ε of $G=(V,E)$ in H, the edge-weighting ω for H, and the usage patterns $\pi_1, \ldots, \pi_n$ for G, one can determine in $|E|\,n$ multiplications, $(|E|-1)\,n$ additions, and $n-1$ comparisons the cost*

$$\mathit{Cost}(\varepsilon; \mathfrak{C}(\pi_1, \ldots, \pi_n); \omega).$$

Optimizing Encodings. If one is given only the graphs G and H, an application Π for G, and a weighting ω of H, the problem of finding a corresponding minimal-cost encoding of G in H is computationally intractable (to be precise, *NP*-complete), even when H, Π, and ω are very simple.

Proposition 2.2. *Given graphs $G=(V,E)$ and H, an application Π for G, an edge-weighting ω for H, and a constant k, the problem of determining whether or not there is an encoding ε of G in H with*

$$\mathit{Cost}(\varepsilon; \Pi; \omega) \leqq k$$

is NP-complete.

This remains true even under one or more of the simplifying assumptions: (a) *H is a line;* (b) *ω is the constant function $\omega(e) \equiv 1$;* (c) *Π is the application $\mathfrak{C}(\pi_1^*, \ldots, \pi_{|E|}^*)$, where the π_i^* are the "characteristic" usage patterns,*

$$(2.14) \quad \pi_i^*(e) = \begin{cases} 1, & \text{if } e = e_i \\ 0, & \text{if } e \neq e_i \end{cases}$$

and $E=\{e_1, \ldots, e_{|E|}\}$. (This application is designated *ALL* later.) (d) *Π is as in part (c), and G is a tree.* (e) *Π is the singleton application $\{\pi_G\}$, where π_G is the*

"*average-case*" *usage pattern*

$$\pi_G(e) = \frac{1}{|E|} \tag{2.15}$$

for all $e \in E$.

Proof. The general assertion follows from the various simplified cases which in turn follow from the *NP*-completeness of either the Bandwidth Minimization Problem (assumptions (a, b, c) [15]), the Simplified Bandwidth Minimization Problem (assumptions (a, b, c, d) [4]), or the Simple Optimal Linear Arrangement Problem (assumptions (a, b, e) [5]). □

Deciding Encodability. In the preceding two subsections, we have assumed, first, that we are given an encoding of G in H (Proposition 2.1) and, next, that we know that G is encodable in H and we have only to find a good encoding (Proposition 2.2). However, the fact that we are modeling data structures by *directed* graphs renders problematical the issue of whether G is encodable in H at all. In fact, even this apparently simple question turns out to be *NP*-complete. (This is trivially not the case if either strongly connected graphs or undirected graphs are used to model "physical storage structures;" Propositions 2.1 and 2.2 would be unaffected by these changes to the model, which would not be unreasonable ones given our intended interpretation.)

Proposition 2.3. *Given graphs G and H, the problem of determining the existence of an encoding of G in H is NP-complete.*

Proof. The following simple proof was suggested by N. Pippenger. Let us be given n disjunctive clauses $C_1, C_2, \ldots, C_n$; and let L denote the set of literals occurring in the clauses. Construct the graph H as follows.

Vertices. If the literal $l \in L$ occurs in clause C_k, then the pair $\langle l, k \rangle$ is a vertex of H. If $\langle l_1, k_1 \rangle$ and $\langle l_2, k_2 \rangle \in L \times \{1, \ldots, n\}$ are vertices of H, then the quadruple $\langle l_1, k_1, l_2, k_2 \rangle$ is a vertex of H. These pairs and quadruples exhaust the vertices of H.

Edges. For all pairs of vertices $\langle l_1, k_1 \rangle$ and $\langle l_2, k_2 \rangle \in L \times \{1, \ldots, n\}$ such that $k_1 \neq k_2$ and $l_1 \neq \sim l_2$ (the negation of l_2), there are two directed edges in H, both having source $\langle l_1, k_1, l_2, k_2 \rangle$, one having target $\langle l_1, k_1 \rangle$, and the other having target $\langle l_2, k_2 \rangle$. These are all the edges of H.

We let our proposed guest graph G be the graph with vertices

$$\{1, \ldots, n\} \cup \{\langle i, j \rangle : 1 \leqq i < j \leqq n\}$$

and edges

$$\{\langle \langle i, j \rangle, i \rangle, \langle \langle i, j \rangle, j \rangle : 1 \leqq i < j \leqq n\}.$$

One verifies easily that G is encodable in H iff G is isomorphic to a subgraph of H iff the conjunction of the n clauses $C_1, \ldots, C_n$ is satisfiable. The *NP*-completeness of the Encodability Problem thus follows from the well-known completeness of the Satisfiability Problem [1]. □

3. Basic Properties of the Framework

Any proposed formalization of a "real" notion must be tested for four properties: the formal notion must be *faithful* in the sense of agreeing with observations about its real counterpart; the formal notion must be *elucidative* in the sense of rendering transparent observations that are not otherwise easily understood; the formal notion must be *stable* in the sense of not magnifying slight perturbations in the input data; the formal notion must be *tractable* in the sense of being a vehicle for the in-depth study of its real counterpart. This section is devoted to justifying our claim that the formal notions of data encoding and cost of an encoding developed in Section 2 enjoy the first three of these properties. The last property, tractability, can be tested only over a period of time; we allege, however, that the initial "returns," as exemplified, say, by [2, 11, 14, 20, 23] all of which studies fall directly into the framework proposed here, are encouraging.

A. Faithfulness of the Model

We note just three instances of the model's agreeing with "reality." With this sampler in mind, the reader can easily carry on with other tests of faithfulness. We leave to the reader the simple proofs of the following propositions.

Proposition 3.1. *For any encoding ε of a graph G in a graph H and any edge-weighting ω of H, if Π and Π' are applications of G, and if $\Pi \subseteq \Pi'$, then*

$$Cost(\varepsilon; \Pi; \omega) \leqq Cost(\varepsilon; \Pi'; \omega).$$

In particular, for all applications Π of G,

$$Cost(\varepsilon; \Pi; \omega) \leqq Cost(\varepsilon; ALL; \omega)$$

where ALL is the set of all usage patterns for G (cf. Proposition 2.2c).

Proposition 3.1 asserts that asking a data structure to do less can never raise the cost of any encoding of the structure.

The next proposition asserts the equally expected fact that employing non-simple paths as images of edges under an encoding is self-defeating in the sense that it increases costs.

Proposition 3.2. *Let ε encode the graph $G=(V,E)$ in the graph H. Let ε' be another such encoding with the property that, for all $e \in E$, the path $\varepsilon'(e) \in Paths(H)$ is a simplification of $\varepsilon(e) \in Paths(H)$: $\varepsilon'(e)$ is obtained from $\varepsilon(e)$ by removing loops (paths with the same source and target). For all applications Π of G, and all weightings ω of H,*

$$Cost(\varepsilon'; \Pi; \omega) \leqq Cost(\varepsilon; \Pi; \omega).$$

The final "test of faithfulness" asserts that various ways of combining usage patterns and/or applications of guest graphs yield the expected costs.

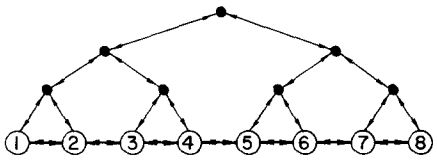

Fig. 3. The encoding ε_3 of the line L_3 in the leaves of the tree T_3. *Thin lines* represent tree edges; *thick lines* represent line edges

Proposition 3.3. *Let ε encode the graph G in the graph H, and let ω be a weighting of H. For any applications Π, Π' of G,*

$$Cost(\varepsilon; \Pi \cup \Pi'; \omega) = \max[Cost(\varepsilon; \Pi; \omega), Cost(\varepsilon; \Pi'; \omega)],$$

and

$$Cost(\varepsilon; \Pi \cap \Pi'; \omega) \leqq \min[Cost(\varepsilon; \Pi; \omega), Cost(\varepsilon; \Pi'; \omega)].$$

For any convex combination $\pi = \Sigma \alpha_i \pi_i$ of usage patterns π_i for G ($\Sigma \alpha_i = 1$),

$$(3.1) \quad Cost(\varepsilon; \{\pi\}; \omega) \leqq \max_i Cost(\varepsilon; \{\pi_i\}; \omega).$$

B. Nonobvious Equivalences Exposed by the Model

The elucidative qualities of our model will be illustrated by an analysis of one family of encodings.

Let ε_n ($n = 1, 2, \ldots$) encode the 2^n-node line L_n in the leaves of the depth n full binary tree T_n, in natural order: node $1 \in L_n$ is placed at T_n's leftmost leaf and so on; see Figure 3 for clarification: as in Figure 1, double-headed arrows abbreviate opposing single-headed arrows. For simplicity, we weight the edges of T_3 with the function $\omega_o \equiv 1$, so that $\omega_o(\varepsilon(e)) \equiv |\varepsilon(e)|$, whence $\omega_o(\varepsilon(\langle 1, 2\rangle)) = \omega_o(\varepsilon(\langle 2, 1\rangle)) = 2$, $\omega_o(\varepsilon(\langle 2, 3\rangle)) = \omega_o(\varepsilon(\langle 3, 2\rangle)) = 4$, and so on.

Recalling the usage patterns π_{tf}, π_{oc}, and π_{pr} for L_n from (2.11), (2.12) and (2.13), respectively, we claim the following nonobvious equivalences.

Proposition 3.4.

a) $Cost(\varepsilon_n; \{\pi_{oc}\}; \omega_o) = Cost(\varepsilon_n; \{\pi_{tf}\}; \omega_o)$.

b) $Cost(\varepsilon_n; \{\pi_{pr}\}; \omega_o) = (1 - 2^{-n})\ Cost(\varepsilon_n; \{\pi_{tf}\}; \omega_o)$.

Now, $Cost(\varepsilon_n; \{\pi_{tf}\}; \omega_o) \sim 4$ [20]; hence, $Cost(\varepsilon_n; \{\pi_{pr}\}; \omega_o) \sim Cost(\varepsilon_n; \{\pi_{tf}\}; \omega_o)$, both asymptotic equalities holding as $n \to \infty$.

Proof. Since $\omega_o(\varepsilon_n(\langle i, i+1\rangle)) = \omega_o(\varepsilon_n(\langle i+1, i\rangle))$ for all i, we find that

$$\mathrm{Cost}(\varepsilon_n; \{\pi\}; \omega_o) = \sum_{1 \leqq i < 2^n} [\pi(\langle i, i+1\rangle) + \pi(\langle i+1, i\rangle)]\, \omega_o(\varepsilon_n(\langle i, i+1\rangle))$$

for any usage pattern π for L_n. As a consequence, if we let $m = 2^n$, then

$$(3.2) \quad \mathrm{Cost}(\varepsilon_n; \{\pi_{tf}\}; \omega_o) = \frac{1}{2^n - 1} \sum_{1 \leqq i < m} |\varepsilon_n(\langle i, i+1\rangle)|;$$

$$\text{(3.3)}\quad \text{Cost}(\varepsilon_n;\{\pi_{oc}\};\omega_o)=\sum_{1\leq i<m}\frac{m-i}{\binom{m}{2}}|\varepsilon_n(\langle i,i+1\rangle)|;$$

$$\text{(3.4)}\quad \text{Cost}(\varepsilon_n;\{\pi_{pr}\};\omega_o)=\sum_{1\leq i\leq m/2}\frac{2i-1}{\binom{m}{2}}|\varepsilon_n(\langle i,i+1\rangle)|$$

$$+\sum_{m/2<j<m}\frac{2(m-j)}{\binom{m}{2}}|\varepsilon_n(\langle j,j+1\rangle)|.$$

Now, it is a straightforward matter to verify that, for all $1\leq i<2^{n-1}$,

$$\text{(3.5)}\quad |\varepsilon_n(\langle i,i+1\rangle)|=|\varepsilon_n(\langle 2^n-i,2^n-i+1\rangle)|;$$

$$\text{(3.6)}\quad |\varepsilon_n(\langle i,i+1\rangle)|=|\varepsilon_n(\langle 2^{n-1}+i,2^{n-1}+i+1\rangle)|,$$

a fortuitous coincidence as we shall now see. Using either of the equalities (3.5) or (3.6), we transform (3.2) to

$$\text{(3.7)}\quad \text{Cost}(\varepsilon_n;\{\pi_{tf}\};\omega_o)=\frac{2}{2^n-1}\sum_{1\leq i<m/2}|\varepsilon_n(\langle i,i+1\rangle)|+\frac{2n}{2^n-1},$$

where the "dangling" term $2n/(2^n-1)$ is the contribution of edge $\langle 2^{n-1},2^{n-1}+1\rangle$ to the cost. Using Equation (3.5) in a similar way, we transform (3.3) to

$$\text{(3.8)}\quad \text{Cost}(\varepsilon_n;\{\pi_{oc}\};\omega_o)=\frac{2}{2^n-1}\sum_{1\leq i<m/2}|\varepsilon_n(\langle i,i+1\rangle)|+\frac{2n}{2^n-1}.$$

Equations (3.7) and (3.8) prove part (a) of the Proposition.

Now we use equation (3.6) in the same manner to transform (3.4) to

$$\text{(3.9)}\quad \textit{Cost}(\varepsilon_n;\{\pi_{pr}\};\omega_o)=\frac{1}{2^{n-1}}\sum_{1\leq i<m/2}|\varepsilon_n(\langle i,i+1\rangle)|+\frac{n}{2^{n-1}}$$

$$=(1-2^{-n})\,\text{Cost}(\varepsilon_n;\{\pi_{tf}\};\omega_o),$$

proving part (b) of the Proposition. □

While there is obviously no way to argue that the formulation of Section 2 is indispensable to rendering the equivalences of Proposition 3.4 transparent, we believe that the proof just presented does establish at least that the formulation does elucidate those nonobvious equivalences.

C. Stability of the Model

The stability of our model evidences itself both in the model's graceful accommodation of errors or misestimates in usage patterns or in implementation costs and in the way data encodings compose.

Misestimates. The bilinearity of our expression (2.10) for the cost of a data encoding guarantees stability in the face of perturbations of either of the cost variables.

Proposition 3.5. *Let ε encode the graph $G=(V,E)$ in the graph $H=(V',E')$.*

(a) *If the usage patterns π, π' for G satisfy*

$$|\pi(e)-\pi'(e)| \leqq \delta \cdot \pi(e) \tag{3.10}$$

for some $\delta>0$ and all $e\in E$, then, for all edge-weightings ω of H,

$$|Cost(\varepsilon;\{\pi\};\omega)-Cost(\varepsilon;\{\pi'\};\omega)| \leqq \delta \cdot Cost(\varepsilon;\{\pi\};\omega).$$

(b) *If the edge-weightings ω, ω' for H satisfy*

$$|\omega(e)-\omega'(e)| \leqq \delta' \cdot \omega(e)$$

for some $\delta'>0$ and all $e\in E'$, then, for all usage patterns π for G,

$$|Cost(\varepsilon;\{\pi\};\omega)-Cost(\varepsilon;\{\pi\};\omega')| \leqq \delta' \cdot Cost(\varepsilon;\{\pi\};\omega).$$

Proof. We prove only part (a), part (b) then following by similar reasoning. Let π and π' stand in relation (3.10) to one another. We then find, for arbitrary ω,

$$\begin{aligned}
|\mathrm{Cost}(\varepsilon;\{\pi\};\omega)-\mathrm{Cost}(\varepsilon;\{\pi'\};\omega)| &= |\sum_{e\in E}(\pi(e)-\pi'(e))\,\omega(\varepsilon(e))| \\
&\leqq \sum_{e\in E}|\pi(e)-\pi'(e)|\,\omega(\varepsilon(e)) \\
&\leqq \sum_{e\in E}\delta\cdot\pi(e)\,\omega(\varepsilon(e)) \\
&= \delta\cdot\mathrm{Cost}(\varepsilon;\{\pi\};\omega). \quad \square
\end{aligned}$$

In informal terms, Proposition 3.5 asserts that any relative error in estimating either a usage pattern or an edge-weighting engenders a relative error of like magnitude in the estimation of the cost of the data encoding.

Compositions of Encodings. If $G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$, and $G_3=(V_3,E_3)$ are graphs, and if ε_1 and ε_2 encode, respectively, G_1 in G_2 and G_2 in G_3, then the composite function $\varepsilon_1\varepsilon_2$ encodes G_1 in G_3. What, however, is the cost of this composite encoding? We proceed now through a natural development to an answer to this question.

Let π be a usage pattern for the graph G_1. When G_1 is encoded in G_2 (by ε_1), π induces a pattern of use on the edges of G_2. The induced pattern can be derived as follows.

Let $e_1, e_2, \ldots, e_n$ be the edges of G_1. In a natural way, the encoding ε_1 transforms the (real number)-(G_1-edge) sequence

$$\pi(e_1), e_1, \pi(e_2), e_2, \ldots, \pi(e_n), e_n$$

into the (real number)-(G_2-path) sequence

$$\pi(e_1), \varepsilon_1(e_1), \pi(e_2), \varepsilon_1(e_2), \ldots, \pi(e_n), \varepsilon_1(e_n).$$

The import of this translation is that when the edge e_i of G_1 is crossed with probability $\pi(e_i)$, then, under encoding ε_1, the path $\varepsilon_1(e_i)$ in G_2 is traversed with precisely this probability also. What this means for the edges of G_2 is clear: under encoding ε_1 and usage pattern π, each edge e' of G_2 is traversed with frequency (relative to the other edges of G_2)

$$(3.11) \quad \varphi(e') = \sum_{e \in E_1} \pi(e) \cdot \#(e'; \varepsilon_1(e))$$

where $\#(e'; \varepsilon_1(e))$ is the number of occurrences of edge e' in the path $\varepsilon_1(e)$. Now, if one sums these frequencies in preparation for normalizing them to probabilities, one finds that

$$(3.12) \quad \sum_{e \in E_2} \varphi(e) = \mathrm{Cost}(\varepsilon_1; \{\pi\}; \omega_o)$$

where, as in Section 3B, ω_o is the constant edge-weighting $\omega_o(e) \equiv 1$ for G_2. Combining (3.11) and (3.12), we arrive at the desired natural notion of *the usage pattern* $\pi^{(\varepsilon_1)}$ *for* G_2 *induced under encoding* ε_1 *by the usage pattern* π *for* G_1, namely,

$$(3.13) \quad \text{for } e \in E_2, \quad \pi^{(\varepsilon_1)}(e) = \varphi(e) \div \mathrm{Cost}(\varepsilon_1; \{\pi\}; \omega_o).$$

Proposition 3.6. *Let* G_1, G_2, G_3, ε_1, *and* ε_2 *be as before. Let* π *be a usage pattern for* G_1, *and let* ω *be an edge-weighting of* G_3. *Then*

$$\mathit{Cost}(\varepsilon_1 \varepsilon_2; \{\pi\}; \omega) = \mathit{Cost}(\varepsilon_1; \{\pi\}; \omega_o) \cdot \mathit{Cost}(\varepsilon_2; \{\pi^{(\varepsilon_1)}\}; \omega).$$

Proof. We proceed by direct calculation.

$$\begin{aligned}
\mathrm{Cost}(\varepsilon_1 \varepsilon_2; \{\pi\}; \omega) &= \sum_{e \in E_1} \pi(e)\, \omega(\varepsilon_1 \varepsilon_2(e)) \\
&= \sum_{e \in E_1} \pi(e) \sum_{e' \in E_2} \#(e'; \varepsilon_1(e))\, \omega(\varepsilon_2(e')) \\
&= \sum_{e' \in E_2} \varphi(e')\, \omega(\varepsilon_2(e')) \\
&= \mathrm{Cost}(\varepsilon_1; \{\pi\}; \omega_o) \cdot \mathrm{Cost}(\varepsilon_2; \{\pi^{(\varepsilon_1)}\}; \omega).
\end{aligned}$$

Here we used (3.11) to proceed from the double summation to the single sum, and we invoked (3.12) and (3.13) to obtain the desired expression. □

Basically, Proposition 3.6 asserts that the cost of the composite encoding is the cost of the second encoding magnified by just that dilation caused by the first encoding, an eminently reasonable allocation of costs. We conclude that our model is stable under composition of encodings.

4. Bounds on Costs of Encodings

The analysis of an encoding, with an eye toward obtaining an exact or an asymptotic expression for its cost, can be an arduous task. If one is willing to settle for a bound on the cost of the encoding, easier techniques will often suffice.

Throughout this section, let ε encode the graph $G=(V,E)$ in the graph $H=(V',E')$, let Π be an application and π a usage pattern for G, and let ω be an edge-weighting function for H.

A. Generally Applicable Bounds

Our first bound verifies that a natural candidate for "worst-case" usage pattern for a data encoding does in fact fill precisely that role.

Proposition 4.1. $Cost(\varepsilon;\Pi;\omega)\leqq\max_{e\in E}\omega(\varepsilon(e))$.

Equality holds whenever $\Pi=ALL$ (cf. Proposition 2.2c).

Proof.
$$\begin{aligned}\mathrm{Cost}(\varepsilon;\Pi;\omega)&=\max_{\pi\in\Pi}\sum_{e\in E}\pi(e)\,\omega(\varepsilon(e))\\&\leqq\max_{\pi\in\Pi}\sum_{e\in E}\pi(e)\cdot\max_{e\in E}\omega(\varepsilon(e))\\&=\max_{e\in E}\omega(\varepsilon(e))\end{aligned}$$

since each $\pi\in\Pi$ is a probability function, so that $\Sigma_e\,\pi(e)=1$.

The fact that equality holds when $\Pi=ALL$ is immediate from the fact that ALL contains the "characteristic" probability functions on E, as defined in (2.14). □

Proposition 4.1 lends special import and interest to studies of "worst-case" cost of data encodings as in [9, 14, 18, 19, 20, 22].

Our second result bounds the cost of ε both above and below, but differs from Proposition 4.1 in that the bounding quantities may not bear any relation to the encoding ε.

Let $\pi_1,\pi_2,\ldots,\pi_{|E|}$ be, in nondecreasing order, the elements of the set $\pi(E)$; and let $\omega_1,\omega_2,\ldots,\omega_{|E|}$ be the similarly ordered sequence of elements of the set $\omega(\varepsilon(E))$.

Proposition 4.2. $\sum_{1\leqq i\leqq|E|}\pi_i\,\omega_{|E|-i+1}\leqq Cost(\varepsilon;\{\pi\};\omega)\leqq\sum_{1\leqq i\leqq|E|}\pi_i\,\omega_i$.

Proof. The result, which appears as Theorem 368 of [7, Section 10.2], is proved easily by noting that, given nonnegative a, b, α, β, the sum

$$ab+(a+\alpha)(b+\beta)$$

exceeds the sum

$$a(b+\beta)+(a+\alpha)b$$

by the quantity $\alpha\beta$. □

B. Bounds for Special Circumstances

When π and/or ω have special properties, then the bounds of Section 4A can be supplemented by often finer counterparts.

The first bounds we find depend on how well the usage pattern π "fits" the encoding ε and the implementation costs ω.

(4.1) The usage pattern π is *well-suited* for the encoding ε and edge-weighting ω if $\pi(E)$ and $\omega(\varepsilon(E))$ are oppositely ordered: for all $e, e' \in E$,

$$\pi(e) \leqq \pi(e') \quad \text{implies} \quad \omega(\varepsilon(e)) \geqq \omega(\varepsilon(e'));$$

π is *ill-suited* for ε and ω if $\pi(E)$ and $\omega(\varepsilon(E))$ are similarly ordered:

$$\pi(e) \leqq \pi(e') \quad \text{implies} \quad \omega(\varepsilon(e)) \leqq \omega(\varepsilon(e')).$$

Proposition 4.3. *If π is well-suited for ε and ω, then*

$$Cost(\varepsilon; \{\pi\}; \omega) \leqq Cost(\varepsilon; \{\pi_G\}; \omega);$$

if π is ill-suited for ε and ω, then

$$Cost(\varepsilon; \{\pi\}; \omega) \geqq Cost(\varepsilon; \{\pi_G\}; \omega),$$

where π_G is the all-edges-equally-likely usage pattern of (2.15).

Proof. Immediate from Tchebyshef's inequalities [7, Section 2.17] and the fact that π is a probability function, so that $\Sigma\, \pi(e) = 1$. □

Just as Proposition 4.1 accentuated the importance of the "worst-case" cost of a data encoding, so does Proposition 4.3 focus attention on the special role of the "average-case" or "all-edges-equally-likely" cost of a data encoding, as studied in [2, 8, 11, 20, 23].

The final bounds we present are of interest only because of the method of generating them, namely, by focussing attention not on the individual edges of G but rather on the equivalence classes of edges induced by the function $\varepsilon\omega$.

(4.2) For $e \in E$, denote by $[e]$ the set

$$[e] = \{e' \in E: \omega(\varepsilon(e)) = \omega(\varepsilon(e'))\}.$$

Let $E^* = \{[e]: e \in E\}$, and for each usage pattern π for G, define the probability function π^* on E^* by

$$\pi^*([e]) = \sum_{e' \in [e]} \pi(e').$$

The transformation (4.2) can be useful in analyzing data encodings because the probability function π^* can be more tractable than π, as the following shows.

Let $\omega(\varepsilon(e_1)) < \omega(\varepsilon(e_2)) < \cdots < \omega(\varepsilon(e_n))$, each $e_i \in E$, comprise, in the indicated order, all the distinct values of $\omega(\varepsilon(E))$.

Proposition 4.4. *If $\varepsilon\omega$ increases subarithmetically, that is, for some $c > 0$,*

$$\omega(\varepsilon(e_{i+1})) \leqq \omega(\varepsilon(e_i)) + c,$$

and if π^* decreases geometrically, that is, for some $\rho > 0$, $\alpha > 1$,

$$\pi^*([e_i]) \leqq \rho\,\alpha^{-i},$$

then

$$Cost(\varepsilon; \{\pi\}; \omega) \leqq \omega(\varepsilon(e_1)) - c + c\,\rho\frac{\alpha}{(\alpha-1)^2}.$$

Proof. Immediate by direct calculation, since

$$\text{Cost}(\varepsilon; \{\pi\}; \omega) = \sum_{[e] \in E^*} \pi^*([e])\,\omega(\varepsilon(e)). \quad \square$$

This technique can be used to estimate the cost

$$\text{Cost}(\varepsilon_n; \{\pi_{tf}\}; \omega_o)$$

of Section 3B. To remind the reader, ε_n encodes the 2^n-node line in the depth n full binary tree as in Figure 3; π_{tf} is the usage pattern (2.11) for the line; ω_o is the identically 1 edge-weighting for the tree. It is not hard to verify that

$$\omega(\varepsilon_n(E)) = \{2, 4, 6, \ldots, 2n\}$$

so that

$$\omega(\varepsilon_n(e_1)) = c = 2.$$

Moreover (and here is where the technique pays off),

$$\pi^*_{tf}([e_i]) = \left(\frac{1}{1-2^{-n}}\right) 2^{-i}.$$

By Proposition 4.4, then, the estimate

$$\text{Cost}(\varepsilon_n; \{\pi_{tf}\}; \omega_o) \leqq \frac{4}{1-2^{-n}} \sim 4$$

is direct. The determination in [20] of the exact value of this cost is just a more detailed application of this technique.

It goes without saying that the specific growth rates of ω and π^* in Proposition 4.4 were for illustration only and have no bearing on this technique of cost estimation.

Acknowledgments. Conversations with N. Pippenger concerning "real" data encodings were of inestimable value. Indeed, his contributions to the formulation in Section 2 were fully as substantive as the author's. The comments, criticisms, and suggestions of L. Snyder and B. Shneiderman were most helpful and are gratefully acknowledged.

References

1. Cook, S.A.: The complexity of theorem-proving procedures. Proc. 3rd ACM Symp. on Theory of Computing, 1970, pp. 151–158
2. DeMillo, R.A., Eisenstat, S.C., Lipton, R.E.: Preserving average proximity in arrays. Comm. ACM (to appear)
3. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Real-time simulation of multihead tape units. J. Assoc. Comput. Mach. **19**, 590–607 (1972)
4. Garey, M.R., Graham, R.L., Johnson, D.S., Knuth, D.E.: Complexity results for bandwidth minimization. Unpublished typescript, 1977
5. Garey, M.R., Johnson, D.S., Stockmeyer, L.J.: Some simplified *NP*-complete graph problems. Theoret. Comput. Sci. **1**, 237–267 (1976)
6. Gotlieb, C.C., Tompa, F.W.: Choosing a storage schema. Acta Informat. **3**, 297–319 (1974)
7. Hardy, G.H., Littlewood, J.E., Pólya, G.: Inequalities. Cambridge Univ. Press 1967
8. Harper, L.H.: Optimal assignments of numbers to vertices. J. Soc. Indust. Appl. Math. **12**, 131–135 (1964)
9. Harper, L.H.: Optimal numberings and isoperimetric problems. J. Combinatorial Theory **1**, 385–393 (1966)
10. Hennie, F.C.: One-tape, off-line Turing machine computations. Information and Control **8**, 553–578 (1965)
11. Iordansk'ii, M.A.: Minimalnye numeratsii vershin derevyev [in Russian]. Problemy Kibernet. **31**, 109–132 (1976)
12. Knuth, D.E.: The art of computer programming. I. Fundamental algorithms. Reading, MA: Addison-Wesley 1968
13. Knuth, D.E.: The art of computer programming. III. Sorting and searching. Reading, MA: Addison-Wesley 1973
14. Lipton, R.E., Eisenstat, S.C., DeMillo, R.A.: Space and time hierarchies for classes of control structures and data structures. J. Assoc. Comput. Mach. **23**, 720–732 (1976)
15. Papadimitriou, Ch. H.: The *NP*-completeness of the bandwidth minimization problem. Computing **16**, 263–270 (1976)
16. Pippenger, N., Fischer, M.J.: Relations among complexity measures. IBM Report RC-6569, 1977
17. Pfaltz, J.L.: Representing graphs by Knuth trees. J. Assoc. Comput. Mach. **22**, 361–366 (1975)
18. Rosenberg, A.L.: Preserving proximity in arrays. SIAM J. Comput. **4**, 443–460 (1975)
19. Rosenberg, A.L.: Storage mappings for extendible arrays. IBM Report RC-5798, 1976. In: Current trends in programming methodology. IV. Data structuring (R.T. Yeh, ed.). Englewood Cliffs, NJ: Prentice-Hall (to appear)
20. Rosenberg, A.L., Snyder, L.: Bounds on the costs of data encodings. Math. Systems theory (to appear)
21. Scheuermann, P., Heller, J.: A view of logical data organization and its mapping to physical storage. Proc. 3rd Texas Conf. on Computing Systems, 1974
22. Sekanina, M.: On an ordering of the set of vertices of a connected graph. Publ. Fac. Sci. Univ. Brno, No. 412, 137–142 (1960)
32. Sheidvasser, M.A.: O dline i shirine razmeshchenii grafov v reshetkakh [in Russian]. Problemy Kibernet. **29**, 63–102 (1974)
24. Shneiderman, B., Shapiro, S.C.: Toward a theory of encoded data structures and data translation. Internat. J. Comput. Information Sci. **5**, 33–43 (1976)
25. Standish, T.A.: Data structures—an axiomatic approach. In: Current trends in programming methodology. IV. Data structuring (R.T. Yeh, ed.). Englewood Cliffs, NJ: Prentice-Hall (to appear)

Received May 2, 1977