SIAM J. ALG. DISC. METH.
Vol. 4, No. 4, December 1983

0196-5212/83/0404-0014 501.25/0

PERFECT STORAGE REPRESENTATIONS FOR FAMILIES OF DATA STRUCTURES*

F. R. K. CHUNG†, A. L. ROSENBERG‡ AND LAWRENCE SNYDER§

Abstract. In this paper we investigate the problem of finding efficient universal storage representations for certain families of data structures, such as the family T_n of n-node binary trees, where the constituent parts of family members are labelled according to a uniform naming scheme. For example, each node of a tree in T_n can be labelled by a binary string describing the sequence of left and right edges taken to reach that node from the root. If one preassigns a distinct memory location to each possible distinct name, then any member of T_n can be stored by storing the contents of each node in the location assigned to the label of that node. However, this would require 2^n-1 memory locations and is wasteful of space, since certain labels can never occur together in a tree in T_n and hence could share a single memory location. We consider the problem of minimizing the number of memory locations needed, viewed in the following general form:

Consider a collection Γ of labelled finite graphs, where each graph has distinctly labelled vertices but different graphs in Γ may share certain vertex labels. A graph U is universal for Γ if U contains every graph $G \in \Gamma$ as a subgraph; U is perfect-universal for Γ if it is universal and there exists a perfect hash function h that maps the labels of graphs in Γ to vertices of U such that h is one-to-one on the vertex labels of each $G \in \Gamma$.

We will show that the smallest perfect-universal graph for T_n has size roughly the square root of the size of the name space and the vertex degree need be no more than 9. We also consider several other families of graphs motivated by data structures: the family $\mathbf{T}_n^{(k)}$ of n node k-ary trees, the family $\mathbf{C}_n$ of $(\leqslant n)$-position two-dimensional chaotic arrays, the family $\mathbf{R}_n$ of $(\leqslant n)$-position two-dimensional ragged arrays and the family of $\mathbf{A}_n$ of $(\leqslant n)$-position rectangular arrays. Sharp bounds for the perfect universal graphs for $\mathbf{T}_n^{(k)}, \mathbf{C}_n, \mathbf{R}_n, \mathbf{A}_n$ are established and perfect hash functions are explicitly constructed.

Introduction. In this paper we study three questions related to the problem of finding flexible storage representations for data structures. The formal framework evolved from the following considerations.

Many families of graphs and of data structures admit consistent families of naming schemes for their constituent parts (cf. [11]). For example, the atomic entries of d-dimensional arrays are referenced via d-tuples of integers; entries of ragged arrays are also often so referenced; and the "name" $<i,j>$ refers to the same entry of a two-dimensional array no matter what size or shape the array has. Similarly, the leftmost grandchild of the root of an ordered binary tree is often referred to as "left, left" or as "LL" or, in the case of LISP S-expressions, as "car(car(root))"; and all of these naming schemes assign the same name to this leftmost grandchild no matter what shape or size the tree has. Thus one can consistently regard these as being a single familial naming scheme for binary trees or for d-dimensional arrays.

One consequence of the existence of these familial naming schemes is that, in place of dynamically allocating storage for an n-element member of a family of data structures that admits such a naming scheme, one could statically allocate storage for

*Received by the editors November 24, 1980, and in revised form January 24, 1983. This paper was typeset at Bell Laboratories, Murray Hill, New Jersey, using the **troff** program running under the UNIX™ operating system. Final copy was produced on April 6, 1983.

†Bell Laboratories, Murray Hill, New Jersey 07974.

‡Department of Computer Science, Duke University, Durham, North Carolina 27706. A portion of the research of this author was supported by the National Science Foundation under grant MCS 8116522, and a portion was done while this author was with the IBM Research Center, Yorktown Heights, New York.

§Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47907. The research of this author was supported in part by the National Science Foundation under grant MCS 78-04749, while this author was at the Department of Computer Science, Yale University, New Haven, Connecticut.

the entire subfamily of n-element structures by using the familial naming scheme: One could set aside storage space sufficient to accommodate all names that could ever occur together in some n-element member of the family. (For instance, the names $<1,4>$ and $<4,1>$ cannot coexist in the same 8-element rectangular array and, so, when allocating space for such arrays, one would permit these elements to share the same space.) One would then store (and retrieve) any particular n-element structure by hashing (with a guarantee of no collisions) to the locations corresponding to the names of the particular elements used. Rosenberg and Stockmeyer [15] use this strategy to allocate storage for rectangular two-dimensional arrays. The most straightforward realization of this strategy for ordered binary trees is based on the fact that every n-node binary tree is a subtree of the depth-n complete binary tree; or, equivalently, the name space for n-node binary trees is a transliteration of the set of binary strings of length less than n. Thus, by laying out the complete depth-n binary tree in contiguous memory locations without pointers (e.g., the root is assigned to relative location 1, location(left(x)) = 2location(x), and location(right(x)) = 2location(x)+1), one has effectively stored any n-node binary tree without pointers. The obvious flaw in this example is that one has allocated 2^n-1 storage locations to save $n-1$ pointers. Similar scenarios can be described using the $n \times n$ array to "store" all n-element two-dimensional rectangular arrays or ragged arrays. The first question we shall address in this paper is: Do more efficient static allocation strategies exist, and if so, how conservative of storage can they be? In the course of answering this question, we shall be extending Sprugnoli's [16] work on collision-free hashing schemes to data structures rather than unstructured sets; Rosenberg and Stockmeyer's [15] work on storage schemes for rectangular arrays of unspecified sizes to data structures other than rectangular arrays; and Lipton, Rosenberg, and Yao's [9] work on hashing schemes for extendible data structures to the case where the hashing schemes must be *perfect* in Sprugnoli's sense (i.e., collision-free). (The use of "perfect" in our title derives from Sprugnoli's use of this term.)

A second (but closely related) use one can make of familial naming schemes is the following. There are a variety of situations in which one wishes to view either data structures [10], [13], [14] or circuits [17] as graphs. In such circumstances one often wishes to deal with families of graphs but soon finds it onerous to have to deal with each graph in the family individually: one would like to have a single "universal" graph that contains as a subgraph each of the graphs in the family in question. In the context of [17], for instance, one would be able, in the presence of a universal graph for trees, to design a single circuit that can be specialized to any individual tree circuit, rather than having to design a special circuit for each individual tree. Indeed, F. R. K. Chung and R. L. Graham have (with coauthors) [2]-[6] studied in detail the problem of finding universal graphs for the family of n-node trees, as well as the special version of the problem where the universal graph must itself be a tree. Aleliunas and Rosenberg [1] study the analogous problem for rectangular arrays. What is not addressed in the cited works is the problem of how hard it is to find the placement of a given graph in the universal graph. One approach to this problem is to have the placement of each individual graph be given by a perfect hash function from the sets of vertices of the individual graphs into the set of vertices of the universal graph. One would not be surprised to learn that universal graphs that are *perfect* in this sense must often be bigger than universal graphs that are constructed without an eye to the layout problem, but how much bigger need they be? The second task of this paper is to quantify the difference in the sizes of universal graphs and *perfect-universal* graphs for the families of graphs that we study.

In [10], [13] it is proposed to study the problem of finding linked storage representations for data structures via a type of graph embedding. One problem with the approach advocated in these papers and their successors is that they overlook the question of programmability. Specifically, they describe the following scenario: One has a structurally complicated graph that represents one's logical data structure. For reasons related to the particulars of one's computing environment, it would prove onerous to store this graph structure directly. Instead, one "encodes" this graph in a structurally simpler one, replacing edges in the complicated graph by paths in the simpler one, thereby trading traversal time for simplicity of storage management. The problem not addressed in these studies is: When one is traversing one's data structure in its encoding form, how does one find the paths corresponding to the edges one wishes to traverse. An attempt is made in [14] to resolve this problem by imposing a notion of "uniformity" on data encodings; but the message of that paper is that uniformity leads to insufferable time- and space-inefficiency in encodings. We make another attempt at the programmability problem here. Say that one has a universal graph for the family of target graphs in a data encoding siutation. Then any ensemble of encodings of the source graphs into this target graph induces, in a natural way, a universal graph for the source family: vertices of the universal-source are the images (under the encodings) of the vertices of the source graphs; and edges are induced in the obvious way by the source edges (so vertices v and v' are adjacent in the universal-source whenever they are the images of adjacent vertices in one of the individual source graphs). The programmability problem is alleviated somewhat by this use of universal graphs, since there is now only one encoding to worry about, namely the encoding of the universal-source in the universal-target, rather than a whole family of encodings. Of course this latter allegation is true only if the universal graphs in question have bounded vertex-degrees, for otherwise one still has unboundedly many edge-path associations to distinguish among at each step. Indeed, we shall place great stock here on our universal graphs' having bounded vertex-degrees.

Having outlined the problems that motivated our study, let us begin to develop our formal framework.

1. Basic definitions.

A. Perfect hash functions. Let $S_1, S_2, \ldots, S_n$ be a collection of finite sets, and let T be a set. The function

$$\phi: S_1 \cup S_2 \cup \ldots \cup S_n \rightarrow T$$

is a *perfect hash function from the collection of S_i into T* if ϕ is one-to-one on each set S_i.

B. Perfect universal graphs. Let $\Gamma = \{G_i\}$ be a family of finite labelled undirected graphs. For brevity, we shall often call a graph $G \in \Gamma$ a "Γ-graph." The *size* of the collection Γ, denoted Size(Γ), is defined to be the cardinality of the union of the vertex-sets of all Γ-graphs.

An undirected graph U is *universal for n-vertex Γ-graphs* if U contains each n-vertex Γ-graph as a subgraph.

An undirected graph U is *perfect-universal for n-vertex Γ-graphs* if it is universal for these graphs and if its universality is witnessed by a perfect hash function from the collection of vertex-sets of Γ-graphs into the vertex-set of the graph

U; that is to say, there is a total *allocation* function

$$\alpha\colon \cup_{G \text{ a } (\leqslant n)\text{-vertex } \Gamma\text{-graph}} \text{Vertices}(G) \rightarrow \text{Vertices}(U)$$

satisfying the following. For every Γ-graph G having at most n vertices, if there is an edge in G connecting vertices v and v', then there is an edge in U connecting vertices $\alpha(v)$ and $\alpha(v')$. If the graphs in Γ have mutually disjoint vertex-sets, then the notions of perfect and ordinary universal graph coincide; however, for the cases we shall be studying, the graphs in each Γ will share many names, and so perfect universality will be a much stronger property than unrestricted universality.

C. The perfection number of a family of graphs. The *perfection number* of the family of graphs Γ is denoted $\text{Perf}(\Gamma)$ and is defined to be the size (= number of vertices) of the smallest perfect-universal graph for Γ. It is an immediate consequence of our framework that $\text{Perf}(\Gamma)$ is also the size of the smallest set T into which the vertex-sets of the graphs in Γ can be hashed perfectly.

D. The families of graphs of interest. We now present the families of graphs we shall be studying, by formal definition and by picture (see Fig. 1). In preparation, we present the following notational conventions.

For each nonnegative integer n, we denote

by $[n]$ the set $\{0,1,\ldots,n-1\}$;
by $\{0,1\}^n$ the set of all the 2^n length-n *binary* strings;
by $\{0,1\}^*$ the set of all finite-length binary strings;
and by $[k]^*$ the set of all finite-length k-ary strings.

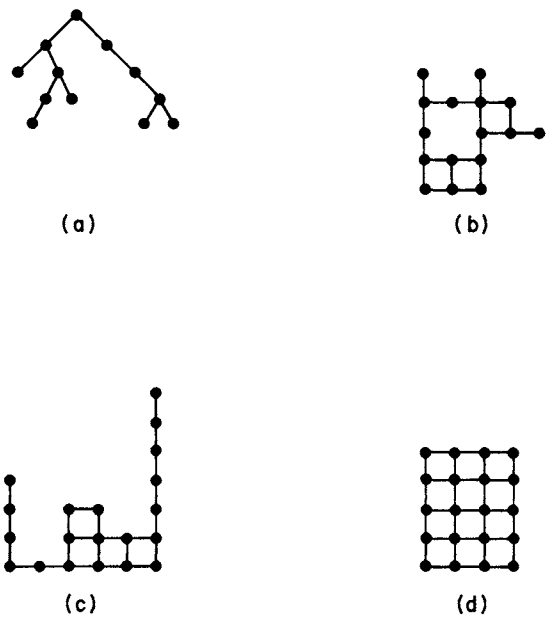

Fig. 1. *Instances of the four graph families:* (a) *binary trees*, (b) *chaotic arrays*, (c) *ragged arrays*, (d) *rectangular arrays.*

Trees. An *n-node (rooted, ordered) binary tree* is a graph whose vertex-set is an n-element *prefix-closed* subset of $\{0,1\}^*$ (i.e., the string x is in the set whenever either $x0$ or $x1$ is), and whose edge-set comprises all 2-element subsets of the vertex-set of the form $(x, x\sigma)$ where $x \in \{0,1\}^*$ and $\sigma \in \{0,1\}$. We denote by $\mathbf{T}_n$ the family of binary trees having n or fewer nodes and by $\mathbf{T}_n^{(k)}$ the family of k-ary trees whose vertex sets are $(\leqslant n)$-element prefix-closed subsets of $[k]^*$.

Chaotic arrays. An *n-position (two-dimensional) chaotic aray* is a graph whose vertex-set is an *order-closed* n-element subset of $N \times N$ (i.e., each vertex is a pair of nonnegative integers, and for each pair $\langle r,s\rangle \neq \langle 0,0\rangle$ in the vertex-set, at least one of $\langle r-1,s\rangle$ and $\langle r,s-1\rangle$ is also in the set), and whose edge-set comprises all 2-element subsets of the vertex-set of the form $(p, p+\delta)$ where $p \in N \times N$ and $\delta \in \{\langle 0,1\rangle, \langle 1,0\rangle\}$. We denote by $\mathbf{C}_n$ the family of chaotic arrays having n or fewer vertices.

Ragged arrays. An *n-position (two-dimensional) ragged array* is a chaotic array whose vertex-set satisfies the following two conditions.

> If $\langle r,s\rangle \in N \times N$ is a vertex of the array, then so also is every element of the set $\{r\} \times [s]$;
> if $\langle r,0\rangle$ is a vertex of the array, then so also is every element of $[r] \times \{0\}$.

We denote by $\mathbf{R}_n$ the family of ragged arrays having n or fewer vertices.

Rectangular arrays. An *n-position rectangular array* is an n-position ragged array whose vertex-set is of the form

$$[a] \times [b]$$

for some nonnegative integers a and b (perforce, $ab = n$). We denote by $\mathbf{A}_n$ the family of rectangular arrays having n or fewer vertices.

2. Perfect universal graphs for binary trees and k-ary trees. The main result of this section is that, even though $\text{Size}(\mathbf{T}_n) = 2^n - 1$, there are perfect-universal graphs for $\mathbf{T}_n$ whose size is roughly only the square root of this quantity. This savings of a square root appears to be very positive, but it contrasts unfavorably with the result by Chung and Graham [5] to the effect that there are (unrestricted) universal graphs for $\mathbf{T}_n$ (or $\mathbf{T}_n^{(k)}$) of n vertices and $O(n \log n)$ edges, and with the result by Chung, Coppersmith, and Graham [3] to the effect that there are (unrestricted) universal *trees* for $\mathbf{T}_n$ (or $\mathbf{T}_n^{(k)}$) of size roughly $n^{O(\log n)}$. For the case of k-ary trees, $k \geqslant 3$, the perfect-universal graph of $\mathbf{T}_n^{(k)}$ is again of size roughly the square root of $\text{Size}(\mathbf{T}_n^{(k)}) = (k^n - 1)/(k-1)$.

Notation. Let $x = \sigma_1 \ldots \sigma_n$ be a binary string (each $\sigma_i \in \{0,1\}$). We define:

$$\text{prefix}(x;k) = \textbf{if } 1 \leqslant k \leqslant n \textbf{ then } \sigma_1 \ldots \sigma_k \textbf{ else } \lambda;$$

and

$$\text{suffix}(x;k) = \textbf{if } 1 \leqslant k \leqslant n \textbf{ then } \sigma_{n-k+1} \ldots \sigma_n \textbf{ else } \lambda.$$

λ here and throughout denotes the null string. We say that the string x has *length* $|x| = n$. When x is viewed as a node in a tree, it is said to *reside at level n* of the tree.

A. Basic lemmas.

LEMMA 2.1. *If the binary strings x and y satisfy*

$$|x| \leqslant \lfloor (n-1)/2 \rfloor \quad \textit{and} \quad |y| \leqslant \lfloor n/2 \rfloor$$

then x and y can coexist in the same n-node binary tree (i.e., there is such a tree whose vertex-set contains both x and y).

Proof. The path from x to the root node λ to y traverses a rooted ordered tree. The number of nodes in this tree is at most the number of nodes encountered on the path, namely,

$$|x|+|y|+1 \leqslant \lfloor (n-1)/2 \rfloor + \lfloor n/2 \rfloor + 1 = n. \quad \square$$

LEMMA 2.2. *Let x and y be binary strings of common length $\lfloor n/2 \rfloor + m$, where $0 \leqslant m \leqslant \lfloor (n-1)/2 \rfloor$. Then x and y can coexist in the same $(\leqslant n)$-node binary tree if and only if*

$$\text{prefix}(x;2m+1-(n \bmod 2)) = \text{prefix}(y;2m+1-(n \bmod 2)).$$

Proof. We shall force x and y into the same tree and then see what happens if we insist that the trees have at most n nodes.

Consider the tree that has unary nodes at levels 0 through $k-1$; a binary node at level k; and two length-l chains from this binary node to x and y. See Fig. 2. Now this tree has depth $l+k$, and it has $2l+k+1$ nodes. By assumption on the size of T, then,

$$l+k = \lfloor n/2 \rfloor + m;$$

and by our insistence,

$$2l+k+1 \leqslant n.$$

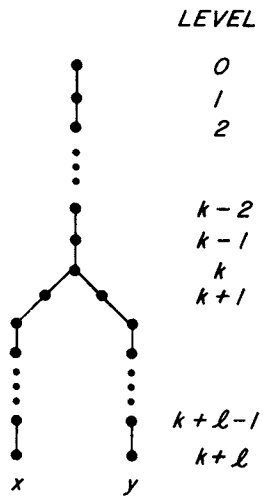

Fig. 2. *Illustrating Lemma 2.2: the n-node tree containing both x and y.*

Now if n is even (say $n=2a$), the first equation becomes

$$l+k = a+m,$$

while the inequality becomes

$$2l+k+1 \leqslant 2a,$$

so that

$$k \geqslant 2m+1.$$

Similar manipulation shows that when n is odd,

$$k \geqslant 2m.$$

Since k is the length of the prefix that is common to x and y, we have thus established the lemma. □

LEMMA 2.3. *The binary strings $x = 0x'$ and $y = 1y'$ can coexist in the same $(\leqslant n)$-node binary tree if and only if*

$$|x|+|y| \leqslant n-1.$$

Proof. Necessity follows from the fact that the path connecting x and y passes through at least $|x|+|y|+1$ distinct nodes (including x and y). Sufficiency follows from the fact that the shortest path connecting x and y is a $(|x|+|y|+1)$-node tree. □

B. The lower bound. Our lower bound on the size of a perfect-universal graph for n-node binary trees is independent of any degree constraints: it holds for unbounded-degree universal graphs as well as for bounded-degree ones.

THEOREM 2.4. *For each integer n,*

$$\text{Perf}(\mathbf{T}_n) \geqslant (3-(n \bmod 2))\cdot\exp 2(\lfloor (n-1)/2 \rfloor)-1.$$

(Throughout this paper, $\exp k(a) = k^a$.)

Proof. By definition of "perfect", the allocation function α for $\mathbf{T}_n$ must be one-to-one on the set of nodes of any n-node binary tree. By Lemma 2.1, therefore, α cannot identify (= map to the same vertex) any two strings that both have length $\leqslant l = \lfloor (n-1)/2 \rfloor$; this forces the target set of α to have at least $\sum_{0\leqslant i\leqslant l} 2^i = 2^{l+1}-1$ elements. When n is odd, this number cannot be raised on the basis of Lemma 2.1. However, Lemma 2.1 informs us that any string of length $\lfloor n/2 \rfloor$ can also coexist with any of the already mentioned strings in an n-node binary tree; and when n is even, any string of length $\lfloor n/2 \rfloor = l+1$ can coexist with any of the strings of length $\leqslant l$. According to Lemma 2.2, two strings of length $l+1$ can coexist in an n-node binary tree only if they start with the same symbol. Therefore, we need assign only 2^l images for these "long" strings. Thus, when n is even, the range of α must have $2^{l+1}-1+2^l = 3\exp 2(\lfloor (n-1)/2 \rfloor)-1$ vertices. □

C. The upper bound. Obviously, the *complete binary tree of depth $n-1$*, that is, the tree whose vertex-set is the prefix-closure of the set of binary strings of length $n-1$, is perfect-universal for n-node binary trees. However, this tree has 2^n-1 nodes, the *square* of the number that Theorem 2.4 asserts a perfect-universal graph must have. In fact, Theorem 2.4's necessary number of nodes is correct not only in order of

magnitude, it is *exactly* the right number: the upper bound we derive now coincides exactly with the lower bound of that theorem.

THEOREM 2.5. *For each integer n,*

$$\text{Perf}(\mathbf{T}_n) = (3-(n \bmod 2))\cdot\exp2(\lfloor(n-1)/2\rfloor)-1.$$

Moreover, there is a perfect-universal graph for n-node binary trees having just this many vertices and having vertex-degree $\leqslant 9$.

Proof. We shall describe the universal graph U and the allocation function α in tandem.

Vertices. Letting $m = \lfloor(n-1)/2\rfloor$, the vertex set of U is the set

$$A \cup B$$

where

$$A = \cup_{0\leqslant k\leqslant m} \{0,1\}^k$$

and

$$B = \textbf{if } n \text{ even } \textbf{then } 0\{0,1\}^m \textbf{ else } \text{EMPTY}.$$

Edges. We shall not enumerate the edges of U explicitly, choosing instead to specify them implicitly by the rule:

Vertices v and v' of U are connected by an edge precisely when $\alpha^{-1}(v)$ contains a tree-node $x \in \{0,1\}^*$ and $\alpha^{-1}(v')$ contains either $x0$ or $x1$, or when this situation occurs with the roles of v and v' reversed.

Allocation. We describe the allocation function α by cases. Let the string $x \in \{0,1\}^*$ be the argument to α:

(1) **if** $|x| \leqslant \lfloor(n-1)/2\rfloor$, **then**

$$\alpha(x) = x;$$

(2) **else if** $|x| = \lfloor n/2\rfloor$, *and n is even,* **then**

$$\alpha(x) = 0\cdot\text{suffix}(x;\lfloor(n-1)/2\rfloor);$$

(3) **else if** x is of the form $\sigma x'$ for $\sigma \in \{0,1\}$, **and if** $|x| = \lfloor(n-1)/2\rfloor+m$ for $1 \leqslant m \leqslant \lfloor n/2\rfloor$, **then**

$$\alpha(x) = (\sim\sigma)\cdot\text{suffix}(x;\lfloor n/2\rfloor-m).$$

where $\sim\sigma$ denotes the element in $\{0,1\}-\{\sigma\}$. (See Fig. 3.)

Verification. It remains to show that the function α is a perfect hash function, i.e., is one-to-one when restricted to any n-node binary tree. This demonstration follows simply from Lemmas 2.2 and 2.3.

First of all, by Lemma 2.2, strings x and y, both of length $\lfloor n/2\rfloor+m$, *cannot* coexist in the same n-node binary tree unless their length-$(2m+1-(n \bmod 2))$ prefixes are identical. The first consequence of this is that, in particular, strings $x = 0x'$ and $y = 1y'$, both of length $\lfloor n/2\rfloor$, *cannot* coexist in the same n-node binary tree if n is even. Hence, clause (2) in the definition of α cannot keep α from being one-to-one on n-node trees. The second consequence is that the identifications of "long" strings in clause (3) of the definition of α cannot keep α from being one-to-one on n-node binary trees. Specifically, in that clause, α identifies all length-$(\lfloor n/2\rfloor+m)$ strings that begin with the same symbol and have the same length-$(\lfloor n/2\rfloor-m)$ suffix.

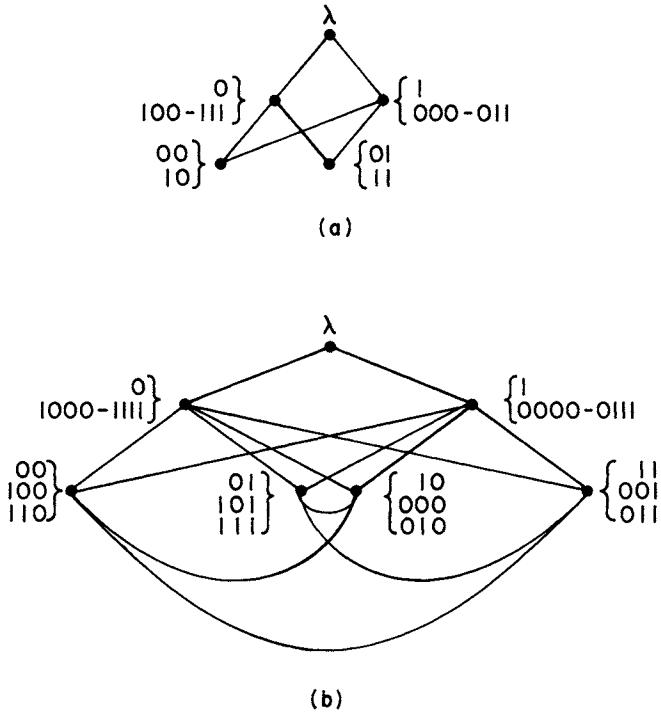

Fig. 3. *Sample perfect-universal graphs for n-node binary trees:* (a) $n = 4$, (b) $n = 5$.

Since the identical strings share this long suffix, they cannot share the long prefix required by Lemma 2.2 and yet remain distinct. In other words, distinct identified strings of length $\geqslant \lfloor n/2 \rfloor + m$ cannot coexist in the same n-node tree.

Secondly, by Lemma 2.3, strings $x = 0x'$ and $y = 1y'$ *cannot* coexist in an n-node binary tree if one of these has length $\lfloor n/2 \rfloor + m$ and the other has length $> \lfloor n/2 \rfloor - m$. Consequently, the ("long" string)-("short" string) identifications made by α in clause (3) cannot prevent its being one-to-one on n-node trees: α identifies length-$(\lfloor n/2 \rfloor + m)$ strings with length-$(\lfloor n/2 \rfloor - m + 1)$ strings; and it follows directly from Lemma 2.3 that no such long-short pair of strings can coexist in the same n-node binary tree.

The bound on vertex-degrees in U is immediate by calculation: a vertex in U has at most 1 edge "entering it from above", at most 2 edges "leaving it to below", at most 4 edges "entering it from below", and at most 2 edges "leaving it to above". The edges of U are of course undirected, but the suggestive "entering, leaving, above, below" should be helpful in following the enumeration. □

The reader can verify easily that, at the cost of (at most) doubling the number of nodes in U, one can make U a perfect *ordered* universal graph, i.e, a graph for which every ordered n-node binary tree appears as an ordered subgraph.

Our results on binary trees can be easily generalized to the family $\mathbf{T}_n^{(k)}$ for $k \geqslant 3$.

THEOREM 2.6. *For each integer n,*

$$\text{Perf}(\mathbf{T}_n^{(k)}) = (k+1-(n \bmod 2))\exp k(\lfloor (n-1)/2 \rfloor)-1 .$$

Moreover, there is a perfect-universal graph for n-node k-ary trees having just this many vertices and having vertex-degree $\leqslant 1+2k+k^2$.

Proof. Lemmas 2.1, 2.2, 2.3 hold for k-ary trees. Therefore, for n odd we have $\text{Perf}(\mathbf{T}_n^{(k)}) \geqslant (k^{l+1}-1)/(k-1)$ where $l = \lfloor (n-1)/2 \rfloor$, since any two strings of length $\leqslant l$ can coexist in the same $(\leqslant n)$-node k-ary tree. For n even, we need an additional k^l vertices in the perfect-universal graph for $\mathbf{T}_n^{(k)}$, since any string of length $l+1$ can coexist with any of the strings of length $\leqslant l$, and two strings of length $l+1$ can coexist in an n-node k-ary tree only if they start with the same symbol. Thus we have

$$\text{Perf}(\mathbf{T}_n^{(k)}) \geqslant (k+1-(n \bmod 2))k^l-1 .$$

To prove the equality we need a perfect hash function α which can be defined the same way as that in Theorem 2.5 except for replacing (3) by (3').

(3') **else if** x is of the form $\sigma x'$ for $\sigma \in [k]$, **and if** $|x| = \lfloor (n-1)/2 \rfloor + m$ for $1 \leqslant m \leqslant \lfloor n/2 \rfloor$, **then**

$$\alpha(x) = (\sigma+1 (\text{mod } k)) \cdot \text{suffix}(x; \lfloor n/2 \rfloor - m) .$$

It is straightforward to check that α is indeed a well-defined perfect hash function. The bound on vertex-degree in the perfect universal graph U can be calculated as follows: a vertex in U has at most 1 edge "entering it from above", at most k edges "leaving it to below", at most k^2 edges "entering it from below", and at most k edges "leaving it to above". Therefore, U has vertex-degree no more than $1+2k+k^2$. □

3. Perfect universal graphs for chaotic arrays. The main result of this section is that no material compaction of chaotic arrays is possible as it was with binary trees. In particular, $\text{Size}(\mathbf{C}_n) = n(n+1)/2$; and any perfect-universal graph for $\mathbf{C}_n$ must have at least roughly half this number of vertices. The (unrestricted) universal graph for $\mathbf{C}_n$ has $O(n)$ vertices and $O(n^{3/2})$ edges [2]. (In fact, this is the bound for the universal graph for the family of planar graphs on n vertices.)

Notation. Let $p = \langle p_1,p_2 \rangle$ be an ordered pair of nonnegative integers. The pair p is said to have *size* $\Sigma(p) = p_1+p_2$. When p is viewed as a position of a chaotic array, it is said to *reside at level* $\Sigma(p)$ of the chaotic array. If $q = \langle q_1,q_2 \rangle$ is another pair of nonnegative integers, then we denote by $M(p,q)$ the third pair

$$M(p,q) = \langle \max(p_1,q_1), \max(p_2,q_2) \rangle .$$

A. The basic lemma.

LEMMA 3.1. *The integer pairs p and q can coexist in the same* $(\leqslant n)$*-position chaotic array if and only if* $\Sigma(M(p,q)) < n$.

Proof. Sufficiency. Consider the graph whose vertex set consists of lattice points encountered in the following walk: start at the origin and proceed as directly as possible to the point $\langle \mu,\nu \rangle$, where

$$\mu = \min(p_1,q_1)$$

and

$$\nu = \min(p_2,q_2).$$

Now go as directly as possible from $<\mu,\nu>$ to p, backtrack to $<\mu,\nu>$ using no new vertices, and go from $<\mu,\nu>$ as directly as possible to q. A straightforward calculation shows that the number of distinct points encountered along this walk is

$$1+\mu+\nu+(p_1-\mu)+(q_1-\mu)+(p_2-\nu)+(q_2-\nu).$$

But this quantity is just $1+\Sigma(M(p,q))$, which by assumption is at most n. Therefore, if we take this set of lattice points and augment it by a set of edges using the rules determining the edges in chaotic arrays, then we find that we have a $(\leqslant n)$-position chaotic array holding both p and q. The sufficiency of the lemma's condition follows.

Necessity. Necessity of the lemma's condition is obvious, for any path including the "origin" $<0,0>$ and both p and q must encounter at least

$$1 + \max(p_1,q_1)$$

distinct lattice points while "moving up" and another

$$\max(p_2,q_2)$$

while "moving across" (viewing the points as first-quadrant lattice points). By the order-closure of the set of vertices of a chaotic array, any two points in a chaotic array are connected to the origin via a path encountering only the points in the array. Hence the number of points encountered cannot exceed n in number, since p and q are assumed to reside in the same n-position chaotic array. □

B. The lower bound.

THEOREM 3.2. *For all integers n,*

$$\text{Perf}(\mathbf{C}_n) \geqslant \lceil n/2 \rceil (\lfloor n/2 \rfloor + 1).$$

Proof. By Lemma 3.1, two points p and q can coexist in the same n-position chaotic array whenever $\Sigma(M(p,q)) < n$. It follows that a perfect hash function α cannot identify any two points p and q for which

$$\max(p_i,q_i) < \lceil n/2 \rceil$$

and

$$\max(p_j,q_j) \leqslant \lfloor n/2 \rfloor$$

where $\{i,j\} = \{1,2\}$. Therefore, the image space of α must contain enough points to give all these pairs distinct images. □

C. The upper bound. As was the case with trees, we establish here an upper bound on the perfect number for chaotic arrays that coincides exactly with the lower bound.

THEOREM 3.3. *For each integer n,*

$$\textit{Perf}(\mathbf{C}_n) \leqslant \lceil n/2 \rceil (\lfloor n/2 \rfloor + 1).$$

Moreover, there is a perfect-universal graph for n-position chaotic arrays having just this number of vertices and having vertex-degree $\leqslant 5$.

Proof. The graph U. The universal graph U will have for vertices the set

$$[\lceil n/2 \rceil] \times [\lfloor n/2 \rfloor + 1].$$

U's edges will be induced by the allocation function α as follows: there will be an edge connecting vertices v and v' of U just when $\alpha^{-1}(v)$ contains a point $p \in N \times N$ and $\alpha^{-1}(v')$ contains either $p+<0,1>$ or $p+<1,0>$. It remains only to describe and validate the function α.

Allocation. The allocation function α is defined by cases:

(1) **if** $p \in [\lceil n/2 \rceil] \times [\lfloor n/2 \rfloor + 1]$, **then**

$$\alpha(p) = p;$$

(2) **if** $p_1 \geqslant \lceil n/2 \rceil$, **then**

$$\alpha(p) = <p_2, n-p_1>;$$

(3) **if** $p_2 > \lfloor n/2 \rfloor$, **then**

$$\alpha(p) = <n-p_2, p_1>.$$

Verification. We must show that the function α is both well defined and one-to-one on n-position chaotic arrays. Both tasks are immediate by Lemma 3.1. By hypothesis, each p in the domain of α resides in some n-position chaotic array; hence, $\Sigma(p) < n$. Thus, if p_1 (resp., p_2) is big, in the sense of case (2) (resp., (3)) above, then neither of p_2 (resp., p_1) or $n-p_1$ (resp., $n-p_2$) can be big. It follows that the mapping α is well defined in the sense that it maps positions of n-position chaotic arrays into vertices of U. Now, each vertex v of U is the image of either one or two chaotic array positions. If v receives only one position, then it cannot prevent α from being one-to-one. If v receives two positions, then one has the form $<q_1,q_2>$ where $q_1 < \lceil n/2 \rceil$ and $q_2 \leqslant \lfloor n/2 \rfloor$, and the other has the form $<p_1,p_2>$ where either $p_1 = q_2$ and $p_2 = n-q_1$, or vice-versa. In either case, $\Sigma(M(p,q)) = n$, so p and q cannot coreside in the same n-position chaotic array. Thus α is one-to-one on all such chaotic arrays and so is a witness to U's being a perfect-universal graph for such arrays, as was claimed. It can be easily verified that most of the vertices in U have degree $\leqslant 4$. Only those vertices $<p_1,p_2>$ with $p_1 \in \{\lceil n/2 \rceil - 1, \lceil n/2 \rceil\}$ or $p_2 \in \{\lfloor n/2 \rfloor, \lfloor n/2 \rfloor + 1\}$ are of degree $\leqslant 5$. □

4. Perfect universal graphs for ragged arrays. Although ragged arrays seem to be closer to rectangular than to chaotic arrays in terms of the amount of uniformity in their structure, they behave for the purposes of our study much more like chaotic arrays. Specifically, we shall see in the next section that rectangular arrays have a perfection number of n. In contrast, we have seen in the last section that chaotic arrays have a perfection number that is only half of the number of vertices in the most naive possible universal graph for chaotic arrays. We shall see now that ragged arrays' perfection number is roughly $n^2/6$, while $\text{Size}(\mathbf{R}_n) = n(n+1)/2$.

A. The basic lemma.

LEMMA 4.1. *Let* $p = <p_1,p_2>$ *and* $q = <q_1,q_2>$ *be integer pairs with* $p_1 \neq q_1$. *The pairs* p *and* q *can coexist in the same* $(\leqslant n)$*-position ragged array if and only if*

$$\max(p_1,q_1)+p_2+q_2 < n .$$

Proof. Sufficiency. The ragged array with vertices

$$([\max(p_1,q_1)+1]\times\{0\}) \cup (\{p_1\}\times[p_2+1]) \cup (\{q_1\}\times[q_2+1])$$

contains both $<p_1,p_2>$ and $<q_1,q_2>$ and has precisely $\max(p_1,q_1)+p_2+q_2+1$ vertices.

Necessity. Follows directly from the fact that, by definition, if the point $<p_1,p_2>$ is in the ragged array R, then $([p_1+1]\times\{0\}) \cup (\{p_1\}\times[p_2+1]) \subseteq R$. □

B. The lower bound.

THEOREM 4.2. *For all integers n,*

$$\text{Perf}(\mathbf{R}_n) \geqslant (\lfloor n/3\rfloor+1)(3\lceil 2n/3\rceil-n)/2 .$$

Proof. We consider the set S of all points (x_1,x_2) satisfying

$$x_1+x_2 < \lceil 2/3\ n\rceil,\ 0 \leqslant x_2 \leqslant \lfloor n/3\rfloor,\ 0 \leqslant x_1 .$$

There are $(\lfloor n/3\rfloor+1)(3\lceil 2n/3\rceil-n)/2$ such points. Suppose (p_1,p_2) and (q_1,q_2) are in S and $p_1 < q_1$. Then

$$\max(p_1,q_1)+p_2+q_2 \leqslant p_2+q_1+q_2 < n.$$

Thus by Lemma 4.1 any two points in S can coexist in the same $(\leqslant n)$-position ragged array and a perfect hash function α cannot identify any two points in S. Therefore we have

$$\text{Perf}(\mathbf{R}_n) \geqslant |S| \geqslant (\lfloor n/3\rfloor+1)(3\lceil 2n/3\rceil-n)/2 . \qquad \square$$

C. The upper bound. We will establish here an upper bound on the perfect number for ragged arrays that coincides exactly with the lower bound.

THEOREM 4.3. *For each integer n,*

$$\text{Perf}(\mathbf{R}_n) = (\lfloor n/3\rfloor+1)(3\lceil 2n/3\rceil-n)/2 .$$

Moreover, there is a perfect-universal graph for n-position ragged arrays having just this number of vertices and having vertex degree $\leqslant 16$.

Proof. Consider the graph U with vertex set S as defined in Theorem 4.2. The edges of U will be induced by the allocation function α which maps points in $\{(x_1,x_2): 0 \leqslant x_1,x_2,\ 0 \leqslant x_1+x_2 < n\}$ to S as defined by cases as follows:

(1) If $p \in S$, then $\alpha(p) = p$.

(2) If $x_1+x_2 \geqslant \lceil 2/3\ n\rceil$, $x_1 > \lfloor n/3\rfloor$, then

$$\alpha(x_1,x_2) = (x_2,n-x_1-x_2).$$

(3) If $x_2 > \lfloor n/3\rfloor$ and $x_1 \leqslant \lfloor n/3\rfloor$, then

$$\alpha(x_1,x_2) = (n-x_1-x_2,x_1).$$

It is straightforward to verify that the function α is well-defined in the sense that it maps positions of n-position ragged arrays into vertices of U. Now, each vertex v of U is the image of at most three ragged array positions. It can be easily checked that α is one-to-one on any n-position ragged array using Lemma 4.1. A vertex $p = (p_1,p_2)$ in U is adjacent to vertices $(p_1+\epsilon,p_2+\epsilon')$ for any $\epsilon, \epsilon' \in \{0,1,-1\}$ and ϵ,ϵ' not both 0 if $(p_1+\epsilon,p_2+\epsilon')$ is in $V(U)-\{p\}$. In fact most vertices of U have degree 8 except for a few with degree $\leqslant 16$. □

5. Perfect universal graphs for rectangular arrays. This section contains two main results. First, in common with trees, rectangular arrays admit significant compaction: Size($\mathbf{A}_n$) is roughly $n \log n$ [12], yet there is a perfect-universal graph for $\mathbf{A}_n$ having only n vertices. Second, although the universal graphs just mentioned have vertex-degrees that are not bounded, independent of n, there are perfect-universal graphs for $\mathbf{A}_n$ having only $2n$ vertices whose vertex-degrees do not exceed 4.

We shall present the initial results about perfect-universal graphs for $\mathbf{A}_n$ in a somewhat cursory manner, since these graphs were studied under a different guise by Rosenberg and Stockmeyer [15].

A. The basic lemma.

LEMMA 5.1. [15] (a) *The point* $<p_1,p_2> \in N\times N$ *resides in some* $(\leqslant n)$*-position rectangular array if and only if*

$$(p_1+1)(p_2+1) \leqslant n.$$

(b) *The points* $p = <p_1,p_2>$ *and* $q = <q_1,q_2>$ *can coexist in the same* $(\leqslant n)$*-position rectangular array if and only if the point* $M(p,q)$ *resides in some* n*-position rectangular array.*

B. Upper and lower bounds.

THEOREM 5.2. [15] *For each integer* n,

$$\text{Perf}(\mathbf{A}_n) = n.$$

Proof. The lower bound on Perf being immediate by the pigeon-hole principle, we turn to the upper bound.

The graph U. Fix on n, let the graph U have for vertices the set $[n]$, and let U's edges be induced by the allocation function

$$\alpha\colon \{p \in N\times N \mid (p_1+1)(p_2+1) \leqslant n\} \to \text{Vertices}(U)$$

as in the previous sections.

Allocation. Define the function

$$\alpha\colon \{p \in N\times N \mid (p_1+1)(p_2+1) \leqslant n\} \to [n]$$

as follows:

(1) For each $m \in [n]$, $\alpha(<m,0>) = m$;

(2) for each $m \in [n]-\{0\}$, α "assigns" to the points in $[\lfloor n/(m+1)\rfloor]\times\{m\}$ the first $\lfloor n/(m+1)\rfloor$ integers in increasing order in the set $[n]-\alpha([\lfloor n/(m+1)\rfloor]\times[m])$.

Verification. The fact that α is indeed an allocation function for a perfect-universal graph for the family $\mathbf{A}_n$ of rectangular arrays follows immediately from the proof in [15] that the function α is one-to-one on all rectangular arrays having n or fewer positions. □

C. A bounded-degree perfect-universal graph for $\mathbf{A}_n$ Although the perfect-universal graph constructed in Theorem 5.2 is optimal in size, it is deficient in one major respect: the maximum degree of the vertices of the graph grows with the size of the array-graphs being imbedded. We believe that this growth is inevitable, but we have been unable to verify or refute the following.

CONJECTURE. *Let the family of graphs* $U_1,U_2,\ldots$, *each* $|U_n| = n$, *be perfect-universal for the collections* $\mathbf{A}_1,\mathbf{A}_2,\ldots$, *respectively. There is no constant* c *such that*

every graph U_n has vertex-degree $\leqslant c$.

We do know, however, that one can attain the $(n \log n)$-to-n compactification of Theorem 5.2 together with bounded degrees if one is willing to suffer a modest increase in the number of vertices in the perfect-universal graph.

THEOREM 5.3. *For each integer n, there is a perfect-universal graph U for $\mathbf{A}_n$ having $|U| = 2n$ vertices and vertex-degree 4.*

Proof. We describe the graph U explicitly.

Vertices. The graph U has the vertex-set

$$\text{Vertices}(U) = \{p \in N\times N \mid 0 \leqslant \text{both}(p) < \lfloor\sqrt{n}\rfloor \text{ or } \lfloor\sqrt{n}\rfloor \leqslant \text{both}(p) < 2\lfloor\sqrt{n}\rfloor\}$$

where references to both(p) indicate that the inequalities govern both p_1 and p_2. Thus, when pictured as a plane set, U looks like two square blocks joined at one corner; see Figure 4(a).

As usual, we shall let the edges of U be induced by our specification of the allocation function

$$\alpha\colon \{p \in N\times N \mid (p_1+1)(p_2+1) \leqslant n\} \to \text{Vertices}(U).$$

Allocation. The function α will be symmetric in the sense that, if $\alpha(\langle p_1,p_2\rangle) = \langle a_1,a_2\rangle$, then $\alpha(\langle p_2,p_1\rangle) = \langle a_2,a_1\rangle$; hence, in defining $\alpha(p)$, we shall assume with no loss of generality that $p_2 < \lfloor\sqrt{n}\rfloor$ (since by Lemma V.1, at least one of p_1,p_2 must be this small). Let $p \in N\times N$ satisfy the following conditions.

(1) $(p_1+1)(p_2+1) \leqslant n$;

(2) $p_2 < \lfloor\sqrt{n}\rfloor$;

(3) $Lp_1/(2\lfloor\sqrt{n}\rfloor) = \Sigma_k \delta_k 2^k$.

Then, letting

$$\pi_1 = p_1 \bmod 2\lfloor\sqrt{n}\rfloor,$$

and

$$\pi_2 = p_2 + \Sigma_k \delta_k \lfloor \lfloor\sqrt{n}\rfloor / 2^{k+1} \rfloor$$

we have

$$\alpha(p) = \textbf{if } \pi_1 < \lfloor\sqrt{n}\rfloor \textbf{ then } \langle\pi_1,\pi_2\rangle \textbf{ else } \langle\pi_1,\pi_2+\lfloor\sqrt{n}\rfloor\rangle.$$

Verification. We have three things to verify: that the function α is well-defined, that it is a perfect hash function for A_n, and that the resulting graph U has vertex-degrees $\leqslant 4$. We treat each issue in turn.

First, let

$$a = \lfloor \log_2 \lfloor p_1/(2\lfloor\sqrt{n}\rfloor)\rfloor \rfloor.$$

By condition (1), we must have

$$p_2 < n/(2^{a+1}\lfloor\sqrt{n}\rfloor + 1).$$

It follows, therefore, that $\pi_2 < \lfloor\sqrt{n}\rfloor$; moreover, it is immediate by definition that π_1, which is the first coordinate of $\alpha(p)$, is less than $2\lfloor\sqrt{n}\rfloor$. Hence, when $\pi_1 < \lfloor\sqrt{n}\rfloor$, both coordinates of $\alpha(p)$ are nonnegative but less than $\lfloor\sqrt{n}\rfloor$; and when $\pi_1 \geqslant \lfloor\sqrt{n}\rfloor$, both coordinates of $\alpha(p)$ are at least $\lfloor\sqrt{n}\rfloor$ but less than $2\lfloor\sqrt{n}\rfloor$. In other words, for every vertex p of a $(\leqslant n)$-position rectangular array, $\alpha(p) \in \text{Vertices}(U)$.

Second, assume that there are distinct pairs p and q of the right form such that $\alpha(p) = \alpha(q)$. Trivially, we must have $p_1 \neq q_1$, since p and q are assumed to be

distinct. Assume that $q_1 > p_1$, and let

$$a = \lfloor \log_2 \lfloor q_1/(2\lfloor \sqrt{n} \rfloor) \rfloor \rfloor.$$

Since α identifies p and q, it must be that

$$p_2 \geqslant \lfloor \sqrt{n} \rfloor / 2^{a+1}.$$

But simple calculation now demonstrates that

$$(p_2+1)(q_1+1) > n$$

so that the point $M(p,q)$ cannot reside in any n-position rectangular array (by Lemma 5.1a), so the points p and q cannot coreside in the same n-position such array (by Lemma 5.1b). It follows that the function α is a perfect hash function for $\mathbf{A}_n$, as was claimed.

It is easy to verify that the vertex degrees in U are bounded by 4 since any edge e of U is in one of the following types (see Fig. 4):

(1) $e = \{<p_1,p_2>, <p_1,p_2+1>\}$;

(2) $e = \{<p_1,\lfloor \sqrt{n} \rfloor -1>, <p_1+\lfloor \sqrt{n} \rfloor, \lfloor \sqrt{n} \rfloor>\}$ where p_1 satisfies $a \leqslant p_1 < \lfloor \sqrt{n} \rfloor$;

(3) $e = \{<p_1,0>, <p_1,2\lfloor \sqrt{n} \rfloor -1>\}$ where

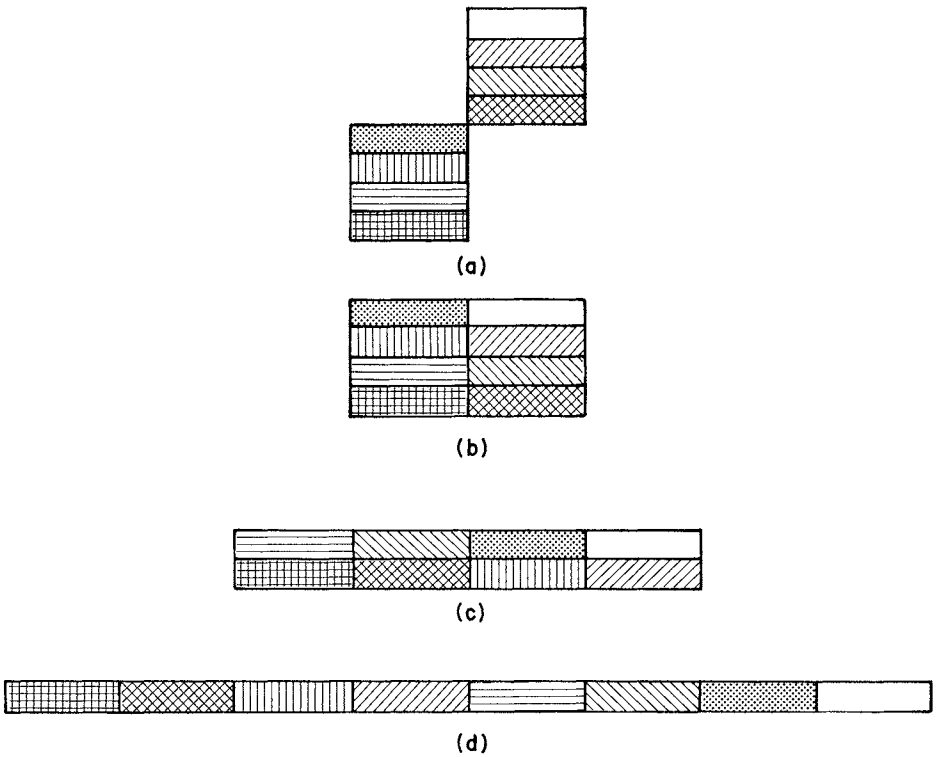

Fig. 4. *Illustrating how the* $2n$*-vertex degree-4 perfect-universal graph for rectangular arrays "covers" short wide arrays:* (a) *a schematic view of the graph* U; (b) U *covering arrays with* $\leqslant 2\lfloor \sqrt{n} \rfloor$ *columns and* $\lfloor \sqrt{n} \rfloor$ *rows;* (c) U *covering arrays with* $\leqslant 4\lfloor \sqrt{n} \rfloor$ *columns and* $0.5\lfloor \sqrt{n} \rfloor$ *rows;* (d) U *covering arrays with* $\leqslant 8\lfloor \sqrt{n} \rfloor$ *columns and* $0.25\lfloor \sqrt{n} \rfloor$ *rows.*

$$p_1 = \sum_k \delta_k \lfloor \lfloor \sqrt{n} \rfloor / 2^{k+1} \rfloor,$$

$$p_2 = \sum_k \delta_k' \lfloor \lfloor \sqrt{n} \rfloor / 2^{k+1} \rfloor,$$

$$\Sigma \delta_k 2^k + 1 = \Sigma \delta_k' 2^k.$$

(4) $e = \{<p_1,p_2>, <q_1,q_2>\}$ where

$e' = \{<p_2,p_1>, <q_2,q_1>\}$ is of type 1,2, or 3.

6. Summary. We summarize our results in Table 1.

TABLE 1

	Size	The order of the universal graph	The order of the perfect universal graph
$\mathbf{T}_n$	2^n-1	$O(n \log n)$	$(3-(n \bmod 2))2^{(\lfloor (n-1)/2 \rfloor)}-1$
$\mathbf{T}_n^{(k)}$	$\frac{k^n-1}{k-1}$	$O(n \log n)$	$(k+1-(n \bmod 2))k^{(\lfloor (n-1)/2 \rfloor)}-1$
$\mathbf{C}_n$	$n(n+1)/2$	$O(n^{3/2})$	$\lceil n/2 \rceil (\lfloor n/2 \rfloor +1)$
$\mathbf{R}_n$	$n(n+1)/2$	$O(n^{3/2})$	$(\lfloor n/3 \rfloor +1)(3\lceil 2n/3 \rceil -n)/2$
$\mathbf{A}_n$	$n \log n$	n	n

where

$\mathbf{T}_n$: The family of $(\leqslant n)$-position binary trees,

$\mathbf{T}_n^{(k)}$: The family of $(\leqslant n)$-position k-ary trees,

$\mathbf{C}_n$: The family of $(\leqslant n)$-position chaotic arrays,

$\mathbf{R}_n$: The family of $(\leqslant n)$-position ragged arrays,

$\mathbf{A}_n$: The family of $(\leqslant n)$-position rectangular arrays.

REFERENCES

[1] R. ALELIUNAS and A. L. ROSENGERG, *On embedding rectangular grids in square grids,* IEEE Trans. Computers, C-31 (1982), pp. 907-913.

[2] L. BABAI, F. R. K. CHUNG, P. ERDOS, R. L. GRAHAM and J. H. SPENCER, *On graphs which contain all sparse graphs,* Annals of Discrete Math., 12 (1982), pp. 21-26.

[3] F. R. K. CHUNG, D. COPPERSMITH and R. L. GRAHAM, *On trees containing all small trees,* in The Theory and Applications of Graphs, G. Chartrand, ed., John Wiley, New York, 1981, pp. 265-272.

[4] F. R. K. CHUNG and R. L. GRAHAM, *On graphs which contain all small trees,* J. Comb. Th.(B) 24 (1978), pp. 14-23.

[5] ———, *On universal graphs for spanning trees.* Typescript, 1979.

[6] F. R. K. CHUNG, R. L. GRAHAM, and N. J. PIPPENGER, *On graphs which contain all small trees,* II., Proc. 1976 Hungarian Colloq. on Combinatorics, North-Holland, Amsterdam, 1978, pp. 213-223.

[7] R. A. DeMILLO, S. C. EISENSTAT and R. J. LIPTON, *On small universal data structures and related combinatorial problems,* Proc. John Hopkins Conference on Information Sciences and System, 1978, pp. 408-411.

[8] J.-W. HONG, K. MEHLHORN and A. L. ROSENBERG, *Cost tradeoffs in graph embeddings, with applications,* J. ACM, to appear.

[9] R. J. LIPTON, A. L. ROSENBERG, A. C. YAO, *External hashing schemes for collections of data structures,* J. ACM, 27 (1980), pp. 81-95.

[10] R. J. LIPTON, S. C. EISENSTAT and R. A. DeMILLO, *Storage hierarchies for classes of control structures and data structures,* J. ACM, 23, (1976), pp. 720-732.

[11] A. L. ROSENBERG, *Data graphs and addressing schemes,* J. Comp. Syst. Sci., 5 (1971), pp. 193-238.

[12] ———, *Managing storage for extendible arrays,* this Journal, (1975), pp. 287-306.

[13] ———, *Data encodings and their costs,* Acta Inform, 9 (1978), pp. 273-292.

[14] A. L. ROSENBERG, L. SNYDER and L. J. STOCKMEYER, *Uniform data encodings,* Theor. Comp. Sci., 11 (1980), pp. 145-165.

[15] A. L. ROSENBERG and L. J. STOCKMEYER, *Storage schemes for boundedly extendible arrays,* Acta Inform., 7 (1977), pp. 289-303.

[16] R. SPRUGNOLI, *Perfect hashing functions: a single probe retrieving method for static sets,* C. ACM, 20 (1977), pp. 841-850.

[17] L. VALIANT, *Universality considerations in VLSI circuits,* IEEE Trans. Computers, C-30 (1981), 135-140.