families of structures. We conjecture that this gives optimal P_D within a constant factor (which may depend on r and k). Needless to say, similar improvements are available, both in preprocessing time and in query time, for a number of other transformations, given sufficiently fast-growing cost functions. Only a small fraction of the possibilities have been explored.

5. Online Transformations

All the transforms in Section 3 have the property that some insertions are very cheap while others are very expensive. For example, in the binary transform the 1023rd insertion is much less costly than the 1024th. While this situation is quite acceptable in certain applications (such as when the total cost of accessing a structure throughout an entire algorithm is counted), it is prohibitive in others (such as online data bases). In this section we will show how the transforms in Sections 3 can be modified to amortize the cost of building static structures over the time of many insertions.

In Section 4, we worked on the principle that any static structure might as well be formed as soon as all its elements became available, since the cost of building it would eventually have to be paid anyway. While this is reasonable if we are concerned only with the total cost of all insertions, it is inappropriate if we wish to make sure that no individual insertion is inordinately expensive. Figure 5.1 shows a strategy which is similar to the binary strategy of Subsection 3.1, except that each structure of cardinality C is completed at the end of the Cth insertion that all its elements are available, rather than at the end of the first such insertion. A structure, s, is said to be *pending* during the Nth insertion if the all elements of s become available at or before the beginning of the Nth insertion and s is completed during the Nth insertion or later. (The $\times$'s in Fig. 5.1 denote the structures that are pending during the eighth insertion). A structure of cardinality C will therefore be pending during exactly C insertions.

To limit the work done in any insertion step, we require that $1/C$ of the work required to build any structure of size C be performed during each of the C steps in which that structure is pending.[19] We call the resulting

[19] The exact means by which this is ensured are left unspecified. We may modify the static algorithm to include appropriate breakpoints (generally an easier task than totally reworking the algorithm into a dynamic algorithm by *ad hoc* methods), or we could assume that we can determine the required computation time in advance (at negligible cost) and set a hardware interrupt. For our present purposes, we will assume that the ability to partition the compute time of a call to insert is available by magic. It should also be noted that the partitioning of the work into equal parts will not be exact in practice; this will lead to slightly greater insertion times than those we are about to advertise.

(as the reader may again wish to verify), the total preprocessing cost being split evenly (within a constant factor) between the two families of structures. The preceding results may be generalized to arbitrary polynomial preprocessing costs and arbitrary binomial transforms, as shown in the following theorem.

THEOREM 4.13 (shift-of-strategy speed-ups). *Let k be an arbitrary positive integer and let r be a real number*[18] *gretaer than* 1. *Suppose that we are given a static structure for a decomposable searching problem with cost functions satisfying the following criteria*:

$Q_S(N)$ *is monotone nondecreasing.*

$P_S(N)$ *grows at least linearly, and*

$P_S(N) = \omega(N^r)$.

Then, a dynamic data structure can be constructed such that

$$Q_D(N) \leq kQ_S(N) \qquad \text{and} \qquad P_D(N) = O\big(N^R P_S(N)\big),$$

where

$$R = (r-1)/(r^k-1).$$

Proof. We maintain a set of structures satisfying the following invariants:

(1) After any insertion there are at most k static structures.

(2) Let j be a positive integer. After the Nth insertion, the cardinality, C_j, of the jth largest structure (if there are at least j structures in existence) satisfies

$$C_j \leq N^{(r^k-r^{j-1})/(r^k-1)}.$$

When an element is inserted, we see how many structures already exist. If there are fewer than k, we simply build the new element into a static structure of cardinality one. If k structures already exist, we rebuild the smallest structure to include the new element. We then repeatedly (zero or more times) merge the smallest two structures until (2) is satisfied. We leave it to the reader to verify that this strategy achieves the advertised performance. Q.E.D.

In any strategy based on the construction in the previous proof, the total preprocessing will be divided evenly (up to constant factors) among k

[18] The nit-picking reader will delight in noting that it is not quite correct to allow r to be an arbitrary real number. In order for the desired transform to be implementable, r must be Turing computable. Even then, if r is very expensive to compute, the bookkeeping costs may kill us. Similar considerations apply to the function h in Theorem 4.11.

We have already mentioned that is is often possible to obtain superior dynamic data structures for individual decomposable problems (e.g., Member) by using specific properties of those problems. Another assumption on which our lower bounds depend is that Theorems 4.1 and 4.2 are the strongest possible results of their kind, because we assume no knowledge about the performance of the original static algorithm. As we saw at the end of Subsection 3.1 the penalty factors, $F(N)$ and $G(N)$, may be greatly reduced (from $\theta(\lg N)$ to $\theta(1)$ in the example of Subsection 3.1) if the cost functions of the static structure are already fast-growing. We now present some results concerning a slightly different way of lowering the penalty functions given fast-growing cost functions for the original static structure.

Suppose we are given a static structure for a decomposable searching problem having preprocessing cost $P_S(N)$ and query cost $Q_S(N)$. We will make only the usual assumption about Q_S—that it is monotone nondecreasing. We will, however, make the assumption that $P_S(N)$ not only grows at least linearly with N, but is actually $\theta(N^2)$. If we apply the 2-binomial (triangular) transform, we will obtain a dynamic structure having cost functions, P_D and Q_D, which satisfy

$$Q_D(N) \leq 2Q_S(N) \qquad \text{and} \qquad P_D(N) = \theta(N^{5/2}).$$

The reader is advised to go through the exercise of verifying the latter assertion. The penalty factor in preprocessing is given by

$$G(N) = P_D(N)/P_S(N) = \theta(N^{1/2}),$$

which is at most a constant factor improvement over the worst-case result given in Theorem 3.2. We appear to get negligible compensation for the fact that the preprocessing cost is already much more than linear. If we look a little more carefully, however, we may notice an interesting phenomenon.

In the triangular strategy, we maintain two structures, a large one, having cardinality $O(N)$, and a small one, having cardinality $O(N^{1/2})$. If we break down $P_D(N)$ into the cost of forming all the large structures built during the first N insertions and the cost of forming all the small structures built during the first N insertions, we find that the large structures have a total cost of $\theta(N^{5/2})$, while the total cost of the small structures is only $\theta(N^2)$. If P_S had been linear, then the costs of the two families of structures would have been equal within a constant factor, each being $\theta(N^{3/2})$. The present disparity suggests that it might be better to merge the small structures into the large ones less frequently. And, indeed, if we adopt the strategy of rebuilding all the elements into a single structure only when the size of the small structure would exceed $N^{2/3}$, we achieve a dynamic structure having

$$Q_D(N) \leq 2Q_D(N) \qquad \text{and} \qquad P_D(N) = \theta(N^{7/3}) = O(N^{1/3}P(N))$$

optimal for space in the worst case. An even more serious objection is that there are a number of problems that satisfy the definition of decomposability only when the unions involved are of disjoint sets.

Our intuitive justification for Restriction 4.2 (contiguity of static structures) is the belief that a partial history which does not satisfy this restriction can be turned into one that does, at no cost in $f(N)$ or $g(N)$, by a kind of "permutation of the names of the elements." To show this would justify the restriction at least for the cases where f is bounded or grows slowly and smoothly, so that the immutability of history is not a significant problem.

For Restriction 4.3 (eagerness of static structures), we can actually give a rigorous justification, at least over the class of transforms which already satisfy Restrictions 4.1 and 4.2. We express this in the following theorem:

THEOREM 4.12 (optimality of eager strategies). *Let N be a positive integer. For any partial history consisting of the first N insertions and satisfying Restrictions* 4.1 *and* 4.2, *there exists a partial history which also satisfies Restriction* 4.3 *and which has $f(N)$ and $g(N)$ no greater than those for the original partial history.*

Proof. Any partial history which satisfies the first two restrictions may be represented by a history diagram. We may ensure that the rectangle in the upper left corner of the diagram represents a structure which is formed as soon as all its elements become available, for any diagram that does not have this property can be transformed at no cost into one that does. The construction is as follows:

> Let R be the upper left rectangle in the diagram. Consider the leftmost rectangle immediately below R. If it is wider than R, then we extend it upward to the top of the diagram, obliterating R; if it is narrower than R, then we extend R downward by one step. This process is repeated until the property holds.

But now the rest of the diagram (excluding the upper-left rectangle) must consist of zero, one, or two staircase-shaped pieces to which the same process may be applied recursively, finally yielding a diagram satisfying Restriction 4.3. No step in this process increases either the total preprocessing cost or the maximum number of simultaneously existing structures, so Restriction 4.3 has been formally justified. Q.E.D.

4.8 *Limitations on the Significance of the Lower Bounds*

The lower bounds we have derived in this section are based on the model of computation given in Subsection 4.1. Before concluding the section, we will mention some of the limitations which this implies for the applicability of our results.

Then, there exists a transform having

$$f(N) \leq \lceil h(N) \rceil, \tag{I}$$

$$g(N) \sim (h(N)/e)N^{1+1/h(N)}. \tag{II}$$

Moreover, given (I), (II) *is optimal up to lower-order terms.*

Proof. A structure having the performance described may be formed by a process of "cutting and pasting" from the history diagrams of the various k-binomial strategies. We omit the details for brevity and for the sake of keeping the reader awake. The optimality of (II), given (I), is implicit in the proof of Theorem 4.10b. Q.E.D.

Our results for transforms in which $f(N) = \omega(\lg N)$ are much less complete. In particular, we know that the performance of the dual k-binomial transforms falls substantially short of the bound given by the inequality

$$g(N) \geq L_{f(N)}(N + 1).$$

We conjecture that this is an inevitable penalty of the immutability of history, and that the dual binomial transforms are in fact optimal in some strong sense, similar to that of Theorem 4.9 for the ordinary binomial transforms. The problem of finding optimal transforms in which $f(N)$ grows faster than $\lg N$ but slower than n^{ϵ} for any positive ϵ remains open.[17]

4.7 *Justification of the Restriction to Arboreal Transforms*

In Subsections 4.1 and 4.3, we introduced three restrictions which together constrained our investigation to arboreal transforms. While we conjecture that arboreal strategies are optimal, in the sense that for any nonarboreal transform there exists an arboreal transform which is at least as good (given the "black box" model described in Subsection 4.1), we have not yet found a rigorous proof. In this subsection, we will summarize our reasons for considering each of the restrictions reasonable.

Restriction 4.1 forbids the existence of multiple structures containing the same element. Our intuition is that any strategy that permits such overlapping structures can be improved by omitting the shared elements from all but one of the overlapping structures. To justify this intuition would require careful examination of the consequences of this omission when that one structure is finally destroyed. We may also forbid overlapping structures on the grounds that transformations which allow them cannot be

[17] We may equivalently view this as the problem of optimizing f when $g(N)$ grows asymptotically faster than N but slower than $N \lg N$.

more expensive (again by more than a constant factor) for insertions. We state this result more formally in the following theorem.

THEOREM 4.10b (optimality of the binary transform). *For any arboreal transform such that* $f(N) = o(\lg N)$, $g(N) = \omega(N \lg N)$.

Proof. Let the function h be defined by

$$h(N) = (\lg N)/f(N).$$

From the hypothesis that $f(N) = o(\lg N)$, it follows that $h(N) = \omega(1)$. Moreover, since $M(N, f(N)) \geq \lg N$, we have $f(N) = o(M(N, f(N))$, which means that the approximation in Theorem 4.8 remains valid.[16] This gives us

$$\begin{aligned} g(N) &\geq L_{f(N)}(N+1) \\ &\geq L_{f(N)}(N) \\ &\sim [f(N)/(f(N)+1)]\, f(N)!^{1/f(N)} N^{1+1/f(N)} \\ &\sim [1](f(N)/e) N^{1/f(N)} N \\ &= [(\lg N)/(e \cdot h(N))] 2^{h(N)} N \\ &= \omega(N \lg N). \end{aligned}$$

Q.E.D.

This implies that any arboreal transform which achieves $F(N) = o(\lg N)$ in the worst case must also pay $G(N) = \omega(\lg N)$.

In the preceding proof, we saw that the approximation given in Theorem 4.8 still serves to provide a lower bound on the growth of g even when f is allowed to grow without bound, provided that $f(N) = o(\lg N)$. The next natural question is whether this bound can always be achieved. It turns out that this is not always possible. If f grows in a very irregular manner, having sudden spurts of growth separated by intervals of almost no change, then the immutability of history will cause $g(N)$ to be much larger than $L_{f(N)}(N+1)$ for values of N immediately following the sudden increases. If f grows "smoothly" (the precise meaning of this term is implicit in the following theorem), however, this lower bound for $g(N)$ is very nearly obtainable. We state this result formally as follows.

THEOREM 4.11 (optimizing g for slowly growing f). *Let h be a monotone non-decreasing differentiable function such that*

$$h(x) = \omega(1) \qquad \textit{and} \qquad h'(x) = o(1/x).$$

[16] That is, consideration (1) may be disregarded.

(2) Our previous caveat about the immutability of history may become more significant.

Since the asymptotic approach of $L_k(n)/[k/(k+1)]k!^{1/k}n^{1+1/k}$ to unity (as n grows and k remains constant) is from below, (1) may be ignored for the purpose of investigating upper bounds. Since the immutability of history can never make it easier to devise efficient transforms, this consideration may be ignored for the investigation of lower bounds. Because of these complicating factors, our results for transforms with unbounded f are less precise than those for bounded f. A few results are nonetheless worth noting, the first of which is the following.

THEOREM 4.10a (optimality of the binary transform). *For any arboreal transform such that* $f(N) = O(\lg N)$, $g(N) = \Omega(N \lg N)$.

Proof. Since constraining the growth of f can only increase and never decrease the necessary growth of g, we need only consider the case where $f(N) = \theta(\lg N)$. We must show that $L_{f(N)}(N+1) = \Omega(N \lg N)$. We define the function M by

$$M(n,k) = \text{Max}\left\{m \middle| \binom{m}{k} \leq n\right\}.$$

From the fact that $f(N) = \theta(\lg N)$, it follows that $M(N,f(N)) - f(N) = \theta(\lg N)$. This gives us

$$\begin{aligned} g(N) &\geq L_{f(N)}(N+1) \\ &\geq L_{f(N)}(N) \\ &\geq f(N)\binom{M(N,f(N))}{f(N)+1} \\ &\sim [f(N)/(f(N)+1)][M(N,f(N)) - f(N)]N \\ &= \theta(N \lg N) = \Omega(N \lg N). \end{aligned}$$

Q.E.D.

This result tells us that the binary transform is optimal in the sense that any transform that pays as small a penalty in search cost (within a constant factor) must pay at least as large a penalty in insertion (again within a constant factor); any arboreal transform which achieves $F(N) = O(\lg N)$ in the worst case must also pay $G(N) = \Omega(\lg N)$.[15] The binary transform is also optimal in the sense that any transform which is actually cheaper (by more than a constant factor) for searches must be strictly

[15] This follows from the fact that Theorems 4.1 and 4.2 are the tightest results possible within our model.

THEOREM 4.9 (optimality of k-binomial transforms). *For any positive integer, k, the k-binomial transform achieves*

$$f(N) \leq k \qquad \text{and} \qquad g(N) \leq L_k(N+1) + N$$

for all positive N.

Proof. Examination of the optimal construction given in the proof of Theorem 4.7 shows that the k-binomial strategy achieves the optimal value of

$$f(N) = L_k(N+1)$$

when N is of the form

$$N = \binom{m}{k} - 1$$

for some $m \geq k$. For intermediate values of N, we need only note that, after the first N insertions under the k-binomial strategy, the sum of the cardinalities of all structures formed so far except *those still in existence after the Nth insertion* (note that the latter must have a total cardinality of N) will never be greater than $L_k(n)$. This fact may be established by induction on k, using the fact that values of $L_k(n)$ are given exactly by linear interpolation between points at which the k-binomial transform gives absolutely minimal values of $f(N)$. Q.E.D.

4.6 *Allowing the Number of Static Structures to Grow*

So far in this section we have only considered minimizing $g(N)$ where $f(N)$ is bounded by a constant. In other words, we have considered only strategies which allow some fixed maximum number of static structures to exist at one time. In Section 3, however, we also investigated strategies (the binary and the dual k-binomial transforms) which allow the number of static structures to grow without limit as the total number of elements in the dynamic structure increases. We will now, therefore, briefly investigate transforms which allow $f(n)$ to grow without bound.

To study the efficiency of transforms in which $f(N)$ is unbounded, we may consider the behavior of $L_k(n)$, where k is allowed to vary with n.[14] We must be aware of two possible consequences of allowing k to grow:

(1) For any particular k, n may never grow large enough for $L_k(n)$ to approach the asymptotic behavior given by Theorem 4.9.

[14] In accordance with the notational conventions of this section, we have $k = f(n) = f(N+1)$, since the first N insertions always give a history diagram which induces a tree of weight $N+1$.

So,

$$\begin{aligned} L_k(n) &= k\binom{m}{k+1} \\ &= kn(m-k)/(k+1) \\ &\sim [k/(k+1)]k!^{1/k}n^{1+1/k}. \end{aligned}$$

Since the growth of $L_k(n)$ is very well behaved,[13] the preceding may be extended to cover all values of n.

THEOREM 4.8 (asymptotic behavior of $L_k(n)$). *Let k be any positive integer. Then,*

$$L_k(n) \sim [k/(k+1)]k!^{1/k}n^{1+1/k}.$$

Proof. The result follows directly from the preceding text. Q.E.D.

By precisely characterizing $L_k(n)$, Theorem 4.7 gives us a bound on the efficiencies of arboreal static-to-dynamic transforms. Any such strategy which has $f(N) \leq k$ for all N must always have $g(N) \geq L_k(N+1)$. The asymptotic behavior of $L_k(n)$ given by Theorem 4.8 and our knowledge that Theorems 4.1 and 4.2 are the best possible within our model tell us that whenever we have

$$F(N) < k$$

for any positive integer k, we must also have

$$G(N) \geq L_k(N+1)/N \sim (k!N)^{1/k}.$$

This is precisely the behavior achieved by the k-binomial transforms, up to lower-order terms. Note, however, that the exact lower bound is not always achievable. The reason for this is the consideration of *immutability of history*. If we know in advance that there will be exactly N insertions, then an optimal strategy can be devised by working backward from an economical tree of weight $N+1$ and right height k. If the total number of insertions to be made turns out to be larger, though, then a different strategy for the first N insertions may have been appropriate. Fortunately, the results of this restriction turn out not to be too severe, since the k-binomial strategies have efficiency very close to this theoretical limit. The following theorem shows that, for any k, the $G(N)$ achieved by the k-binomial transform is optimal (for $(F(N) \leq k)$ not only to within lower-order terms but actually to within an additive constant of unity.

[13] Given the values where n is of the form "m choose k," we can find the *exact* values at all other n by linear interpolation.

This establishes that our expression is an upper bound on $L_k(n)$. To establish that this is also a lower bound, we must show that no other way of expressing n as the sum of two positive numbers, a and b, will give a smaller value for

$$L_k(a) + a + L_{k-1}(b). \tag{III}$$

To show this, we consider the effect on the value of expression (III) of increasing or decreasing a by steps of 1.[11] Suppose we start with a and b chosen to satisfy (II), and then start incrementing a and decrementing b by steps of 1. So long as a remains less than $\binom{m}{k}$ and b remains greater than $\binom{m-1}{k-1}$, the effect of each increment will be to increase $L_k(a) + a$ by $((m-1) - k + 1) + 1 = m - k + 1$ and to decrease $L_{k-1}(b)$ by $(m-1) - (k-1) + 1 = m - k + 1$, leaving the total value of (III) unchanged.[12] However, as soon as either a or b exceeds the stated bound, one or more of the following things will happen:

1. the incremental growth of $L_k(a)$ will increase while the incremental shrinkage of $L_{k-1}(b)$ decreases or remains the same,

2. the incremental shrinkage of $L_{k-1}(b)$ will decrease while the incremental growth of $L_k(a)$ increases or remains the same, or

3. b will diminish to 0.

In any case, a smaller value for (III) will not be obtained. Similarly, if we start with a and b as in (II) and decrease the value of a while increasing b, then we will have zero or more steps at which (III) remains unchanged, zero or more steps where the increase in $L_{k-1}(b)$ exceeds the decrease in $L_k(a) + a$, and finally the step at which a diminishes to zero. Thus, the rules given in (II) give an optimal partitioning of n into a and b. This completes the induction step and the proof. Q.E.D.

The use of the auxiliary variable, m, in expression (I) makes it a bit difficult to grasp intuitively what is being said about the effects of n and k on $L_k(n)$. To make the picture clearer, we will briefly study the asymptotic behavior of $L_k(n)$ as k remains fixed and n grows without bound. Consider first what happens as n ranges only over binomial coefficients of the form $\binom{m}{k}$. We note that $n = \binom{m}{k}$ implies

$$m - k + 1 \leq (n/k!)^{1/k} \leq m.$$

[11] In the following, we assume that $k > 1$. If $k = 1$ we must always take $b = 1$ (and $a = n - 1$), since only then is $L_0(b)$ defined.

[12] The incremental changes given here are found by substitution into the second term of the right-hand side of (I), under the induction hypothesis.

For the case $m = k$, we have

$$\begin{aligned} k\binom{m}{k+1} + (m-k+1)N &= k\binom{k}{k+1} + (k-k+1)N \\ &= k(0) + (1)0 \\ &= 0 \\ &= L_k(1). \end{aligned}$$

Inductive step ($n > 1$). We first show that the right-hand side of (I) gives an upper bound on $L_k(n)$. Note that

$$\binom{m-1}{k} + \binom{m-1}{k-1} = \binom{m}{k} \le n \le \binom{m+1}{k} = \binom{m}{k} + \binom{m}{k-1}.$$

We now pick a and b such that

$$\begin{aligned} \binom{m-1}{k} \le a \le \binom{m}{k}, \\ \binom{m-1}{k-1} \le b \le \binom{m}{k-1}, \\ a + b = n. \end{aligned} \tag{II}$$

By Theorem 4.6, we have

$$\begin{aligned} L_k(n) &\le L_k(a) + a + L_{k-1}(b) \\ &= k\binom{m-1}{k+1} + ((m-1)-k+1)(A) \\ &\quad + \binom{m-1}{k} + A \\ &\quad + (k-1)\binom{m-1}{k} + ((m-1)-(k-1)+1)(B) \\ &= k\left[\binom{m-1}{k+1} + \binom{m-1}{k}\right] + (m-k+1)(A+B) \\ &= k\binom{m}{k+1} + (m-k+1)N. \end{aligned}$$

where

$$\begin{aligned} A &= a - \binom{m-1}{k}, \\ B &= b - \binom{m-1}{k-1}, \\ N &= n - \binom{m}{k}. \end{aligned}$$

Then,

$$L_k(n) = k\binom{m}{k+1} + (m - k + 1)N, \tag{I}$$

where

$$N = n - \binom{m}{k}.$$

Proof. Our proof will proceed by induction on k and, for each fixed positive value of k, by induction on n.

Base step ($k = 0$). In this case, we have

$$\binom{m}{k} = 1 = \binom{m+1}{k}.$$

This implies that $n = 1$, so the right-hand side of (I) reduces to

$$\begin{aligned}(0)\binom{m}{0+1} + (m - 0 + 1)\left(n - \binom{m}{k}\right) &= 0 + (m + 1)(1 - 1).\\ &= 0\\ &= L_0(1).\end{aligned}$$

Inductive step ($k > 0$). We now must show that the theorem holds for any $k > 0$ assuming it holds for all smaller k. We proceed by induction on n. In doing this, we must take note of the interaction between m and n. Since k is positive, $\binom{m}{k}$ increases monotonically with m. Thus, the minimum possible value of n is $\binom{k}{k} = 1$, and for any positive value of n, there is at least one possible value for m (and occasionally there will be two).

Base step ($n = 1$). In this case we may choose either

(a) $m = k - 1$; $N = 1 - \binom{k-1}{k} = 1 - 0 = 1$, or

(b) $m = k$; $N = 1 - \binom{k}{k} = 1 - 1 = 0$.

We must show that (I) hold for either choice of m. For the case $m = k - 1$, we have

$$\begin{aligned}k\binom{m}{k+1} + (m - k + 1)N &= k\binom{k-1}{k+1} + ((k - 1) - k + 1)N\\ &= k(0) + (0)1\\ &= 0\\ &= L_k(1).\end{aligned}$$

height, and left path length of T may be recursively computed from properties of A and B by the relations

$$|T| = |A| + |B|,$$
$$\mathrm{rh}(T) = \max(\mathrm{rh}(A), \mathrm{rh}(B) + 1),$$
$$\mathrm{L}(T) = \mathrm{L}(A) + |A| + \mathrm{L}(B).$$

From this, we obtain the following recurrence for $L_k(n)$:

THEOREM 4.6 (recurrence for $L_k(n)$). *Let n and k be any positive integers. Then,*

$$L_k(n) = \begin{cases} 0, & n = 1, \\ n - 1 + L_k(n-1) = \binom{n}{2}, & k = 1, n > 1, \\ \min\limits_{1 \le i \le n-1} \left[L_k(i) + i + L_{k-1}(n-i) \right], & k > 1, n > 1. \end{cases}$$

Proof. The results for $k = 1$ follow by considering the unique binary tree of any weight which has right height ≤ 1. For the case $k > 1$, we consider a tree T having weight $n > 1$ and height k. Let t be the root of T, and let A and B be the subtrees rooted at $a = \mathrm{lson}(t)$ and $b = \mathrm{rson}(t)$, respectively. Then we must have

$$1 \le |A| < n,$$
$$|A| + |B| = n,$$
$$\mathrm{rh}(A) \le k,$$
$$\mathrm{rh}(B) \le k - 1.$$

Moreover, if the left path length of T is to be minimal, then the left path lengths of A and B must be minimal. That is, we must have

$$\mathrm{L}(A) = L_k(|A|) \quad \text{and} \quad \mathrm{L}(B) = L_{k-1}(|B|).$$

These requirements are precisely captured by our recurrence. Q.E.D.

We now come to the principal theorem of this section, wherein the behavior of $L_k(n)$ is precisely characterized in terms of binomial coefficients.

THEOREM 4.7 (characterization of $L_k(n)$). *Let k and m be nonnegative integers such that $k \le m$, and let n be a positive integer satisfying*

$$\binom{m}{k} \le n \le \binom{m+1}{k}.$$

first N insertions into a dynamic structure maintained by some arboreal strategy. Then, $\mathrm{L}(T) = g(N)$.

Proof. Given in the preceding text. Q.E.D.

We may also characterize N and f in terms of tree properties:

THEOREM 4.5 (relation of N and f to tree properties). *Let N be a positive integer and let T be the tree induced from the history diagram representing the first N insertions into a dynamic structure maintained by some arboreal strategy. Then,*

$$|T| = N + 1 \qquad \textit{and} \qquad \mathrm{rh}(T) = f(N).$$

Proof. Inspection of Fig. 4.1 will reveal that these results are obvious. Q.E.D.

In the remainder of this section we will use N to denote the number of elements inserted into a dynamic structure under some arboreal strategy, and $n = N + 1$ to indicate the number of leaves in the corresponding tree.

The theorems proved so far in this section allow use to address the problem of "simultaneously minimizing" F and G by investigating a closely related problem about trees, namely, that of "simultaneously minimizing" the right height and left path length of a tree with a fixed number of nodes. To discuss this more precisely, we make the following definition:

DEFINITION (minimal left path length). Let n and k be positive integers. We define

$$L_k(n) = \mathrm{Min}\{\mathrm{L}(T) | T \text{ is a tree such that } |T| = n \text{ and } \mathrm{rh}(T) \leq k\}.$$

Since the only tree with zero right height is the tree of one node (which also has zero left path length), we also define

$$L_0(1) = 0.$$

By convention, we will regard $L_0(n)$ as "positive infinity" whenever $n > 1$. A tree with n leaves, right path length k, and left path length $L_k(n)$ will be called an *economical* tree.

In the next subsection, we will investigate the behavior of $L_k(n)$ as k and n vary, and then restate our findings in terms of lower bounds on worst-case penalty functions.

4.5 *The Behavior of* $L_k(n)$

Consider a binary tree, T, with root node t. Let A and B be the subtrees rooted at $a = \mathrm{lson}(t)$ and $b = \mathrm{rson}(t)$, respectively. The weight, right

4.4 Tree Properties and Their Relation to Performance

We now introduce some basic vocabulary for discussing properties of binary trees.

DEFINITIONS (tree properties). Let T be a binary tree. Then leaves(T) denotes the set of all leaves of T and nodes(T) denotes the set of all internal nodes of T. The *weight* of T, denoted $|T|$, is defined as the cardinality of leaves(T). For any internal node, a, of T the left and right sons of a are denoted lson(a) and rson(a), respectively. If a is a leaf of T, then the *right depth* of a, written rd(a), is defined as the number of right branches along the path from the root of T to a. The *right height* of T, rh(T), is the maximum right depth of any leaf of T. The *right path length* of T, R(T), is defined as the sum of the right depths of all leaves of T. Left depth, left height, and left path length are defined analogously.

We will sometimes identify a (not necessarily internal) node, x, of a tree with the subtree rooted at x. For example, we may write $|x|$ to indicate the number of leaves which are descendants of x.

We now make the following observation:

THEOREM 4.3 (alternate characterization of left path length). *Let T be a tree. Then,*

$$L(T) = \sum_{n \in \text{nodes}(T)} |\text{lson}(n)|.$$

Proof. Consider any leaf, x, of T. We need only note that the left branches along the path from the root of T to x emanate precisely from those nodes of T whose left sons contain x. Q.E.D.

With this characterization of left path length in mind, we may now relate the trees induced by arboreal strategies to the penalty functions associated with those strategies.

Consider the tree in Figure 4.1b. To each static structure created during the partial history represented by that tree, there corresponds a right (horizontal in the diagram) branch whose length (in the diagram) is proportional to the cardinality of that static structure. Moreover, for any internal node, n, of the tree, the length (in the diagram) of the right branch from n corresponds precisely to the number of leaves in the *left* son of n. By summing over all internal nodes of the tree, we establish the following result:

THEOREM 4.4 (relation of g to left path length). *Let N be a positive integer and let T be the tree induced from the history diagram representing the*

case, however, that all transforms are so representable; in order for a static structure to be represented as a (contiguous) rectangle in a history diagram, it is necessary that it be built from a set of elements which were inserted consecutively during the history of the structure. We now impose our second restriction on the class of dynamic structures to be considered:

RESTRICTION 4.2 (continguity of static structures). We will restrict our attention to transforms whose histories are representable by history diagrams.

Indeed, we will further restrict our attention to history diagrams (such as those in Section 3) in which every rectangle reaches to the "diagonal" of the diagram. We may state this otherwise as

RESTRICTION 4.3 (eagerness of static structures). We will restrict our attention to transforms in which each static structure is built as soon as all its elements have been inserted, and in which the elements of any discarded static structure are always built into a single new static structure (along with some additional elements).

Strategies which satisfy Restrictions 4.1, 4.2, and 4.3 will be called *arboreal* strategies for a reason that will soon become obvious.

Consider the history diagram for the first N insertions into a dynamic structure which is maintained by an arboreal strategy. Any such diagram induces a binary tree, as shown in Fig. 4.1. We may draw this tree by tracing the left and upper edges of each rectangle in the diagram. The internal nodes of the tree will thus be at the upper left corners of the various rectangles; each internal node of the tree corresponds to a (unique) static structure. We will now go on to study some relationships between the efficiencies of arboreal strategies and properties of their induced trees.

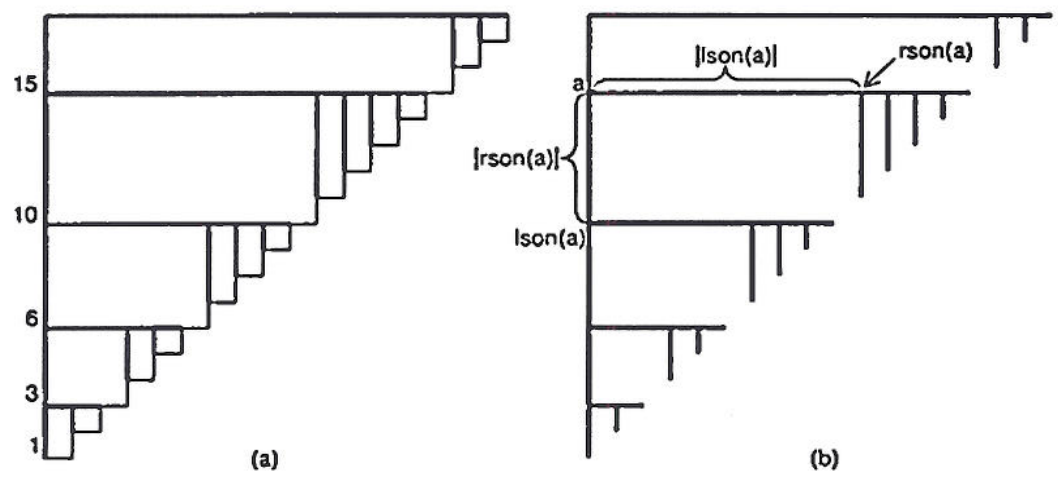

FIG. 4.1. A history diagram and its induced tree. (a) A partial history diagram. (b) The induced tree.