The Annals of Applied Probability
1996, Vol. 6, No. 4, 1284–1302

ON TREE-GROWING SEARCH STRATEGIES

BY JANICE LENT AND HOSAM M. MAHMOUD[1]

George Washington University

Using the concept of a "tree-growing" search strategy, we prove that for most practical insertion sorting algorithms, the number of comparisons needed to sort n keys has asymptotically normal behavior. We prove and apply a sufficient condition for asymptotically normal behavior. The condition specifies a relationship between the variance of the number of comparisons and the rate of growth in height of the sequence of trees that the search strategy "grows."

1. Introduction. Insertion sort is a popular on-line sorting algorithm. At each stage of the sorting, the data obtained so far make up a sorted array. The algorithm reads in a new datum, searches for its proper position in the array and inserts it. When a data file of size $n \geq 2$ is to be sorted, the ith search is for the position of the $(i + 1)$st key, $i = 1, 2, \ldots$. We shall denote the algorithm that performs the ith search by S_i and call the collection of search algorithms $\mathscr{S} = \{S_i\}_{i=1}^{\infty}$ a *search strategy*. Doberkat (1982) and Panny (1986) have studied the moments of the number of comparisons required for insertion sort with one particular (linear) search strategy. In this paper, we show that if the values in a data array have been randomly selected from a continuous distribution, the number of comparisons needed to sort the values by most practical search strategies has asymptotically normal behavior. Thus we investigate properties of search strategies in general, with insertion sort serving as a conspicuous application.

In order to insert a new key in a sorted data array, an implementation of insertion sort selects a sequence of *probes:* keys in the sorted array to which the algorithm compares the new key. If the new key is larger than a given probe, the algorithm selects a larger probe; if the probe's value exceeds that of the new key, the search continues in the segment of the data array containing keys smaller than the probe. In general, the search algorithms applied for different keys in an insertion sort may be independent of each other. One may, for example, imagine a search strategy $\mathscr{S} = \{S_i\}_{i=1}^{\infty}$ in which S_k is binary search and S_{k+1} is linear search. Practitioners, however, usually prefer strategies in which all the algorithms applied are coded as a single procedure which is then invoked with different parameters at different stages of the sorting. Most efficient, easy-to-use strategies thus possess certain

Received January 1995; revised August 1996.
[1]Research partially supported by NSA Grant MDA904-92-H3086.
AMS 1991 *subject classifications*. Primary 60F05, 68P10; secondary 05C05.
Key words and phrases. Searching, sorting, limit distributions.

consistency properties, which we will specify and use to prove our main result. Rudimentarily, when all the search algorithms S_i in a strategy $\mathscr{S}$ specify probe positions through a reasonable function of the *length* of the fragment of the data array being searched, we will call $\mathscr{S}$ a *consistent* strategy.

To identify a class of reasonable probe-selection functions, we introduce *decision trees:* deterministic binary trees whose nodes correspond to the positions of the probes selected by a search algorithm. Since the binary tree is therefore a fundamental instrument in our analysis, we briefly review its definition and basic properties. A *binary tree* is a hierarchical structure comprising nodes organized in levels. A node can have either (a) no children —in which case the node is called a *leaf*, (b) a left child, (c) a right child or (d) two distinct children, one left and one right. It is customary to draw a binary tree with the *root*—the only node with no ancestors—at the top and the leaves at the bottom. The *level* or *depth* of a specific node is the length of the path (the number of edges) from the node to the root node, which is on level zero. The *size* of a binary tree is the number of its nodes. If level k of a tree has 2^k nodes—the maximum number possible—it is said to be *saturated*. A tree in which all existing levels, except possibly the lowest, are saturated is *complete*. (Some authors require that the nodes on the lowest level of a complete tree be positioned as far to the left as possible, but the positioning of the nodes within a level is irrelevant to our analysis.) The *height* of the tree is the length of the longest root-to-leaf path in the tree. We extend a binary tree by adding enough *external* nodes (squares in our diagrams) to ensure that each original *internal* node (bullets in our diagrams) has two children. Each leaf thus has two external nodes as its left and right children. An extended binary tree of size n has $n + 1$ external nodes.

A decision tree representing a search algorithm is constructed as follows: The position of the first probe chosen becomes the root of the decision tree; the two possible positions of the second probe—at most one on each side of the first probe—become its children and so forth. A probe falling at one of the ends of the data array will not have two internal nodes as children; one or both of its children will then be external nodes in the extended tree. The algorithm S_i is represented by T_i, a decision tree of size i whose root-to-leaf paths represent the possible probe sequences of S_i. We shall call the search strategy $\mathscr{S}$ a *tree-growing strategy* if, for every positive integer i, the shape of T_{i+1} may be obtained by adding an internal node in one of the insertion positions—one of the external nodes—of T_i. Since the tree-growing property provides an element of consistency across algorithms in the search strategy, we will restrict our class of consistent strategies to those possessing this property.

Assume the ranks of the data elements being sorted form a random permutation of the integers $\{1, \ldots, n\}$, as is the case when the data values are selected at random from any continuous distribution. Let the random variable C_n be the total number of times $\mathscr{S}$ compares a new key to a probe during the sorting of the first n keys. The class of *normal* search strategies

comprises strategies for which C_n is asymptotically normal; that is,

$$\frac{C_n - \mathbf{E}[C_n]}{\sqrt{\mathbf{Var}[C_n]}} \to_{\mathscr{D}} N(0,1).$$

We shall see that the normality of a subclass of tree-growing strategies—the consistent strategies—may be established by relating the variance of C_n to the rate of growth in height of the decision trees in the family $\mathscr{T} = \{T_i\}_{i=1}^{\infty}$. Our sufficient condition for normality establishes asymptotic normality but does not give the asymptotic mean or variance of C_n, which obviously depend on the particular strategy $\mathscr{S}$. Identifying these requires additional investigation of the strategy. For linear search and binary search, the two most commonly used strategies, we will establish asymptotic normality and calculate the asymptotic means and variances which serve as centering and scaling factors.

The next section provides examples of tree-growing search strategies. In Section 3 we identify a subclass of tree-growing strategies as *consistent*. The remaining sections focus on the normality of consistent strategies. Section 4 gives a sufficient condition for the normality of tree-growing strategies. In Section 5, we use this condition to establish the asymptotic normality for binary search and linear search. We compute the mean and variance for the binary search strategy—the variance includes periodic fluctuations—and use the well-known mean and variance for the linear search strategy [Gonnet and Baeza-Yates (1991)] to derive full asymptotic distributions. In Section 6, we use the sufficient condition to prove our main result: all consistent strategies are normal. (This result is further extended in the Appendix to include some tree-growing implementations of Fibonaccian search.)

2. Examples of tree-growing strategies. We illustrate the tree-growing property through simple examples. One commonly used strategy known as linear search from the bottom always selects as a first probe the largest value in the data array being searched. When searching for the position of the $(i+1)$st key, linear search from the bottom probes positions i, $i-1$, $i-2$ and so forth, until it discovers a key less than the $(i+1)$st key. If it finds such a key in position j, it inserts the new key in position $j+1$. Though linear search from the bottom is inefficient, it is useful for searching sequential access data files [e.g., data stored on a tape, as discussed by Hu and Wachs (1987)]. Figure 1 shows the decision trees of linear search from the bottom. The shape of T_{i+1} is obtained from T_i by adjoining a leaf to replace the leftmost external node of T_i. The i nodes of T_i are relabeled in T_{i+1}, and it is the *shape* of T_{i+1} that evolves from that of T_i by "tree-growing."

Variants of linear search from the bottom include the obvious linear search from the top as well as a slightly more complex method called c-jump search. Still based on sequential searching, c-jump search starts at the "top" of the data array and advances c positions at a time. When it finds a probe larger than the new key, it reverses direction and performs a linear search "from the bottom" of the relevant fragment (a fragment of size $c-1$) of the array. Three-jump search is illustrated by the decision trees of Figure 2.

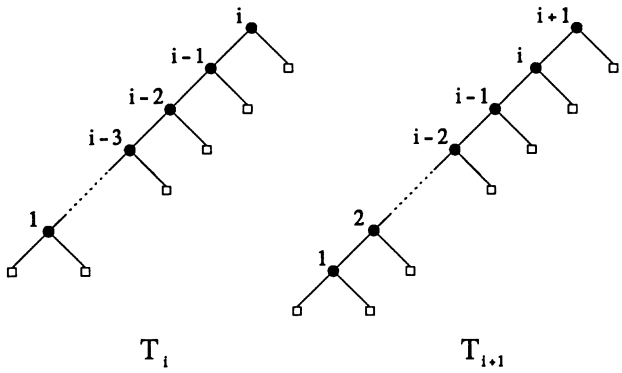

FIG. 1. *The extended decision trees for linear search from the bottom.*

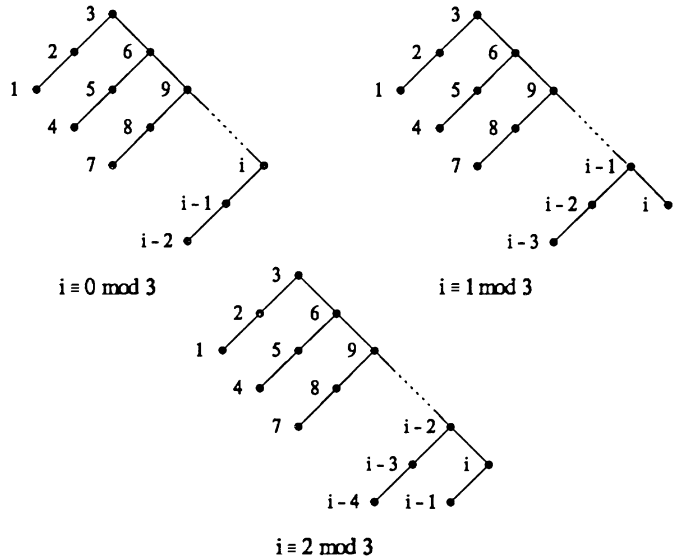

FIG. 2. *The (unextended) decision trees of three-jump search.*

A more efficient search method also commonly used is binary search, which always probes the middle position of the remaining portion of the list. Thus when the search has been confined to a fragment of the data array between an upper bound u and a lower bound l, binary search chooses the probe in position $\lceil (l + u)/2 \rceil$. For $i = 6$, the decision tree representing binary search is that of Figure 3.

Though all of the search strategies discussed above lie in the class of consistent strategies defined in the next section, strategies reasonable for some applications lie outside this class. Examples include alternating linear

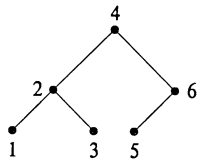

FIG. 3. *The (unextended) decision tree of binary search.*

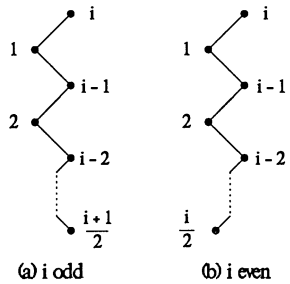

FIG. 4. *The (unextended) decision trees of alternating linear search.*

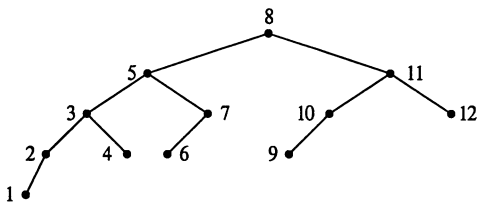

FIG. 5. *The (unextended) decision tree of Fibonaccian search in an array of* 12 *keys.*

search—a hybrid of linear search from the bottom and linear search from the top—which is represented by the decision trees of Figure 4. Figure 5 shows a decision tree for Fibonaccian search, a method so called because its decision tree is endowed with a recursive partition that follows the Fibonacci number sequence. (The usual definition of the Fibonacci number F_j is $F_j = F_{j-1} + F_{j-2}$, with $F_0 = 0$, $F_1 = 1$. Thus $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, $F_5 = 5$ and so forth.) When the tree size is $F_{k+1} - 1$, where F_j is the jth Fibonacci number, the two subtrees of the root node have sizes $F_k - 1$ and $F_{k-1} - 1$, and the property propagates recursively in the subtrees. Fibonaccian search is sometimes preferred to binary search because Fibonaccian search requires only the operations of addition and subtraction to compute the position of the next probe [see Knuth (1973), Section 6.2.1].

Like alternating linear search, Fibonaccian search fails to meet our consistency criteria. We will see, however, that all of the search strategies discussed

here are normal, except possibly some implementations of Fibonaccian search. (The normality of Fibonaccian search may depend on the search strategy used when the number of keys in the array being searched is not of the form $F_k - 1$. In the Appendix, we discuss two implementations of Fibonaccian search for which we can prove normality.)

3. Consistent strategies. In this section we present a condition on the search strategy $\mathscr{S}$ that is equivalent to the tree-growing property and give a rigorous definition of consistency. Let T_i be the decision tree representing the search algorithm S_i—the algorithm used to insert the $(i+1)$st key. Let A_i be the sorted array formed by the first i keys. We search for the correct position of the $(i+1)$st key by successively selecting probes in A_i, at each step comparing the $(i+1)$st key to the selected probe. At the first step in the insertion, we select one probe; at the second step, we again select one probe, but the probe we select depends on the outcome of our comparison of the $(i+1)$st key to the first probe. Thus there are two possible second probes: one in case the $(i+1)$st key is larger than the first probe and another in case it is smaller (unless, of course, the first probe selected falls at one of the ends of A_i—in this case, the algorithm would specify only one probe at the second step). In general then, at the jth step in the insertion, the algorithm specifies up to 2^{j-1} possible probes.

Let P_{ij} be the set of all possible positions for the first j probes in the insertion of the $(i+1)$st key. Then P_{ij} has at most $2^j - 1$ elements, which correspond to the nodes on levels zero through $j-1$ of T_i. Let $A(i,j,1), A(i,j,2), \ldots, A(i,j,2^j)$ be the subarrays of A_i (in increasing order, though some may be empty) created by the partitioning which results from deleting all elements of P_{ij}. If we have some adjacent probes or probes at either of the ends of the array, some of the subarrays $A(i,j,k)$ will be empty. Let l_{ijk} be the length of $A(i,j,k)$, $k = 1, 2, \ldots, 2^j$ [zero if $A(i,j,k)$ is empty]. For the $(j+1)$st step in the insertion, the algorithm specifies one probe from each of the nonempty subarrays. If $A(i,j,k)$ is not empty, let p_{ijk} be the position of the next probe within the subarray $A(i,j,k)$, relative to the other elements of the subarray.

It is easy to show that the search strategy $\mathscr{S}$ has the tree-growing property if and only if, for all i, j and k such that $A(i,j,k)$ is not empty,

$$p_{ijk} = f(l_{ijk}, j, k),$$

for some function f such that

$$f(l+1, j, k) = f(l, j, k) + \alpha(l, j, k)$$

and $\alpha(l,j,k)$ takes values in the set $\{0, 1\}$, where l is any natural number. So the tree-growing property implies that (1) the probe selected in $A(i,j,k)$ at the $(j+1)$st step cannot depend directly on i, the index of the insertion, and (2) if the array $A(i,j,k)$ were larger by one element, the position of the probe selected would be either p_{ijk} or $p_{ijk} + 1$. The tree-growing property thus provides some similarity of probe selection functions within and across the

algorithms S_i, $i = 1, \ldots, n$. We restrict our class of *consistent* strategies to those which possess additional consistency properties:

DEFINITION. Let $\mathscr{S} = \{S_i\}_{i=1}^{\infty}$ be a tree-growing strategy for insertion sort. We call $\mathscr{S}$ a *consistent* strategy if, for all i, j and k such that $A(i, j, k)$ is not empty, S_i specifies the relative rank [within $A(i, j, k)$] of the probe selected in $A(i, j, k)$ by

$$p_{ijk} = \varphi_{\mathscr{S}}(l_{ijk}), \qquad 1 \le p_{ijk} \le l_{ijk},$$

where the function $\varphi_{\mathscr{S}}$ has the following property: if $g_{\mathscr{S}}(n) = \min(\varphi_{\mathscr{S}}(n) - 1, n - \varphi_{\mathscr{S}}(n))$, then $\lim_{n\to\infty} g_{\mathscr{S}}(n)/n$ exists.

In words, $g_{\mathscr{S}}(n)$ is the size of the smaller subtree rooted at level one of T_n, and consistency requires that the proportion of nodes belonging to the smaller subtree approach a limit as n approaches infinity. Note also that for consistent strategies, the position of the probe p_{ijk} within $A(i, j, k)$ depends only on l_{ijk}, the length of the subarray. Since consistent strategies are tree-growing, we have

$$\varphi_{\mathscr{S}}(l + 1) = \varphi_{\mathscr{S}}(l) + \alpha_{\mathscr{S}}(l)$$

and $\alpha_{\mathscr{S}}(l)$ takes values in the set $\{0, 1\}$. Examples of consistent strategies include c-jump search, which is specified by the function

$$\varphi_{\mathscr{S}}(l) = \begin{cases} c, & \text{if } l > c, \\ l, & \text{otherwise.} \end{cases}$$

In this case, $\lim_{n\to\infty} g_{\mathscr{S}}(n)/n = 0$. For alternating linear search, however, the position of the next probe is a function of both l_{ijk} and j:

$$p_{ijk} = \begin{cases} 1, & j \text{ odd}, \\ l_{ijk}, & j \text{ even}. \end{cases}$$

Thus p_{ijk} is not a function of the length only—it depends explicitly on j (the step number). So, though alternating linear search is tree-growing (see Figure 4), it is not consistent.

4. A sufficient condition for normality of tree-growing strategies. Let T_i be the decision tree for S_i, the search algorithm used to insert the $(i + 1)$st key by the tree-growing search strategy $\mathscr{S} = \{S_i\}_{i=1}^{\infty}$. Let h_i denote the height of T_i, let X_i denote the number of comparisons made by S_i and let $C_n = \sum_{i=1}^{n-1} X_i$, the total number of comparisons required by $\mathscr{S}$ for sorting n keys. Each insertion is performed independently of all others, so we take the X_i's to be independent random variables. Let $s_n^2 = \mathrm{Var}[C_n]$.

We also define the symbol Ω: if χ and ψ are functions of n, we say that $\chi(n) = \Omega(\psi(n))$ if $\psi(n) = O(\chi(n))$.

LEMMA 1. *If $h_n = o(s_n)$, then $\mathscr{S}$ is a normal strategy.*

PROOF. For $i = 1, \ldots, n-1$, let $Y_i = X_i - \mathbf{E}[X_i]$ and let $\mathscr{F}_i$ denote the distribution function of Y_i. Then

$$P\{|Y_i| > h_n + 1\} \le P\{\max(X_i, \mathbf{E}[X_i]) > h_i + 1\} = 0,$$

since X_i cannot exceed $h_i + 1$. Now if $h_n = o(s_n)$ and $\varepsilon > 0$, there is a constant N_ε such that, for all $n > N_\varepsilon$, $h_n + 1 < \varepsilon s_n$. Then for $i = 1, \ldots, n-1$,

$$\int_{\{|y| \ge \varepsilon s_n\}} y^2 \, d\mathscr{F}_i(y) \le \int_{\{|y| > h_n + 1\}} y^2 \, d\mathscr{F}_i(y) = 0,$$

because the integral is taken over a set of zero probability. Thus for $n > N_\varepsilon$,

$$\sum_{i=1}^{n-1} \int_{\{|y| \ge \varepsilon s_n\}} y^2 \, d\mathscr{F}_i(y) = 0.$$

So

$$\lim_{n \to \infty} \frac{1}{s_n^2} \sum_{i=1}^{n-1} \int_{\{|y| \ge \varepsilon s_n\}} y^2 \, d\mathscr{F}_i(y) = 0$$

and the normality of $\mathscr{S}$ follows from the Lindeberg theorem [see Billingsley (1986), page 368]. □

Although most practical strategies satisfy the condition of the lemma, the condition does not hold for all tree-growing strategies. Consider, for example, a strategy described by a sequence of decision trees $T_1, T_2, \ldots$ that satisfies the following property: let $k_1 = 2$ and for each integer $i \ge 2$, let $k_i = k_{i-1} + 2^{k_{i-1}/2}$. The k_i's form a sequence of even numbers. For each $i \ge 1$, each of the trees T_n for $n = 2^{k_i+1}, \ldots, 2^{k_i+1} + 2^{k_i/2} - 1$ has levels 0 through k_i saturated, while its remaining $n - 2^{k_i+1} + 1$ nodes form a thin branch at the bottom of the tree; that is, levels $k_i + 1, \ldots, n - 2^{k_i+1} + k_i + 1$ of T_n each contain only one internal node. In a recursive fashion, the strategy first grows a complete tree of height k_i, then grows a thin branch of length $2^{k_i/2}$ at the bottom of the complete tree. Finally, it fills out levels $k_i + 1$ through $k_i + 2^{k_i/2}$ to obtain the next complete tree, a complete tree of height k_{i+1}. Let $n_i = 2^{k_i+1} + 2^{k_i/2} - 1$, the size of the smallest decision tree whose thin branch is grown to length $2^{k_i/2}$. The height of T_{n_i} is $h_{n_i} = k_i + 2^{k_i/2} = \Omega(2^{k_i/2})$, whereas it can be shown that $s_{n_i} = O(2^{k_i/2})$. Thus $\limsup_{i \to \infty} h_{n_i}/s_{n_i} > 0$. While this strategy is not of practical interest, it provides an example of a tree-growing strategy that violates the condition of Lemma 1. It can also be shown, by the Lindeberg–Feller theorem, that this strategy is not normal.

5. Normality of two commonly used search strategies. We illustrate the use of the sufficient condition for normality by showing the normality of two search strategies commonly used for insertion sort: binary search and linear search. We begin by introducing some notation to be used here and in later sections.

NOTATION. Let $\mathscr{S} = \{S_i\}_{i=1}^{\infty}$ be a consistent strategy and let $\mathscr{T} = \{T_i\}_{i=1}^{\infty}$ be the sequence of decision trees representing $\mathscr{S}$. For $i = 1, \ldots, n$, let h_i be the height of T_i, and for $k = 1, 2, \ldots,$ let $L_k = \min\{i\colon h_i = k\}$ and $U_k = \max\{i\colon h_i = k\}$. That is, $T_{L_k}, T_{L_k+1}, \ldots, T_{U_k}$ are the only trees in $\mathscr{T}$ that have height k; thus $U_k + 1 = L_{k+1}$. Define $U_0 = 1$, and for $k = 1, 2, \ldots,$ let $m_k = U_k - U_{k-1}$. As before, let C_n denote the number of comparisons needed to sort the first n keys by insertion sort using $\mathscr{S}$ and let X_i denote the number of comparisons required for inserting the $(i + 1)$st key.

We may write

$$C_n = \sum_{i=1}^{n-1} X_i.$$

So

$$\mathbf{E}[C_n] = \sum_{i=1}^{n-1} \mathbf{E}[X_i]$$

and, since we assume the X_i's are independent,

$$\mathrm{Var}[C_n] = \sum_{i=1}^{n-1} \mathrm{Var}[X_i].$$

Binary search is a consistent strategy which, when searching a subarray of length l, selects as a probe the element whose rank (relative to the other elements in the subarray) is $\lceil l/2 \rceil$. This strategy results in a sequence of complete decision trees. The external nodes of the complete decision tree representing the binary search algorithm S_i lie only on one or two levels: the lowest level $\lfloor \log_2 i \rfloor$ (if $i + 1$ is a power of 2) or the two lowest levels $\lfloor \log_2 i \rfloor$ and $\lfloor \log_2 i \rfloor + 1$. Thus $X_i = \lfloor \log_2 i \rfloor + B_i$, where B_i is a Bernoulli random variable that assumes the value 1 with probability

$$\frac{2(i + 1 - 2^{\lfloor \log_2 i \rfloor})}{i + 1},$$

the proportion of external nodes at level $\lfloor \log_2 i \rfloor + 1$, as is easily shown. So

$$\mathbf{E}[X_i] = \frac{2(i + 1 - 2^{\lfloor \log_2 i \rfloor})}{i + 1} + \lfloor \log_2 i \rfloor$$

and

$$\begin{aligned} \mathbf{E}[C_n] &= \sum_{i=1}^{n-1} \frac{2(i + 1 - 2^{\lfloor \log_2 i \rfloor})}{i + 1} + \sum_{i=1}^{n-1} \lfloor \log_2 i \rfloor \\ &= n \log_2 n + O(n). \end{aligned}$$

For binary search, $U_k = 2^{k+1} - 1$, the number of nodes in a complete tree of height k, so $m_k = 2^k$. To compute $s_{U_j}^2$, we may first sum the variances of the

X_i's corresponding to trees of height k, for $k = 1, \ldots, j$, and then sum over k:

$$s_{U_j}^2 = \sum_{k=1}^{j} \sum_{i=0}^{2^k-1} \mathrm{Var}\left[X_{L_k+i}\right].$$

Again using the result about the number of nodes on levels $\lfloor \log_2 i \rfloor$ and $\lfloor \log_2 i \rfloor + 1$,

$$\sum_{i=0}^{2^k-1} \mathrm{Var}\left[X_{L_k+i}\right] = \sum_{i=0}^{2^k-1} \left(\frac{2(i+1)}{2^k+i+1}\right)\left(1 - \frac{2(i+1)}{2^k+i+1}\right).$$

Some straightforward algebraic manipulation shows that

$$\sum_{i=0}^{2^k-1} \mathrm{Var}\left[X_{L_k+i}\right] = 6(2^k)\left[H_{2^{k+1}} - H_{2^k}\right] - 4^{k+1}\left[H^{(2)}_{2^{k+1}} - H^{(2)}_{2^k}\right] - 2^{k+1},$$

where $H_n^{(m)} = \sum_{i=1}^{n} 1/i^m$, the nth harmonic number of order m. (We omit the superscript when it is 1.) To simplify this expression, we use the asymptotic approximations

$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right),$$

where γ is Euler's constant, and

$$H_n^{(2)} = \frac{\pi^2}{6} - \frac{1}{n} + O\left(\frac{1}{n^2}\right).$$

Using these we obtain

$$\sum_{i=0}^{2^k-1} \mathrm{Var}\left[X_{L_k+i}\right] = (6\ln 2 - 4)2^k + O(1).$$

Then

$$\begin{aligned} s_{U_j}^2 &= (6\ln 2 - 4) \sum_{k=1}^{j} 2^k + O(j) \\ &= \Omega(U_j). \end{aligned}$$

Also, if $0 \le l \le 2^j - 1$,

$$\begin{aligned} \sum_{i=0}^{l} \mathrm{Var}\left[X_{L_j+i}\right] &= 4^{j+1}\left(\frac{1}{2^j+l+1} - \frac{1}{2^j}\right) \\ &\quad + 6(2^j)(H_{2^j+l+1} - H_{2^j}) - 2l + O(1). \end{aligned}$$

Thus for general $n = L_j + i$, where $0 \le i \le 2^j - 1$,

$$\begin{aligned} s_{L_j+i}^2 &= s_{U_{j-1}}^2 + 4^{j+1}\left(\frac{1}{2^j+i+1} - \frac{1}{2^j}\right) \\ &\quad + 6(2^j)(H_{2^j+i+1} - H_{2^j}) - 2i + O(1). \end{aligned}$$

Since, in the case of binary search, $h_{U_j} = O(\ln U_j)$, there is a positive constant c such that $s_{U_j} = \Omega(U_j^{1/2}) = \Omega(\exp(ch_{U_j}))$. Then for $i = 0, \ldots, m_j - 1$,

$$\begin{aligned} s_{L_j+i} &> s_{U_{j-1}} \\ &= \Omega\Big(\exp\Big(c\Big[h_{L_j+i} - 1\Big]\Big)\Big) \\ &= \Omega\Big(\exp\big(ch_{L_j+i}\big)\Big). \end{aligned}$$

That is, $s_n = \Omega(\exp(ch_n))$, so $h_n = o(s_n)$ and the sufficient condition for normality is satisfied. Writing $n = L_j + i$ and simplifying the above expression for $s^2_{L_j+i}$ gives

$$s_n^2 = n\left[\frac{6(1+f_n)\ln 2 - 6}{2^{f_n}} - 2 + \frac{4}{4^{f_n}}\right] + O(\ln n),$$

where

$$f_n \equiv \log_2 n - \lfloor \log_2 n \rfloor.$$

[The sequence $\{f_n\}_{n=1}^{\infty}$ is dense on the interval $[0, 1)$.] Then

$$\frac{C_n - n\log_2 n}{\sqrt{nA_n}} \to_{\mathscr{D}} N(0,1),$$

where

$$A_n = \frac{6(1+f_n)\ln 2 - 6}{2^{f_n}} - 2 + \frac{4}{4^{f_n}}.$$

The analytic continuation of the function $A_n = A(f_n)$ to the real line, as f_n ranges over the interval $[0, 1)$, attains its minimum value (approximately 0.151911) at the point

$$\frac{W(-8/3e^2) + 2}{\ln 2} - 1 = 0.141469\ldots,$$

where $W(x)$ is the principal branch of the omega function [see Fritsch, Schafer and Crowley (1973)]. It attains its maximum value (approximately 0.174449) at

$$\frac{W(0, -8/3e^2) + 2}{\ln 2} - 1 = 0.707059\ldots,$$

where $W(0, x)$ is the zeroth branch of the omega function. Intuitively, we can think of A_n as an asymptotic average of the variances of $\{X_1, \ldots, X_n\}$. Thus A_n increases in n as long as $\mathrm{Var}[X_n] > A_{n-1}$. This occurs while the numbers of external nodes on the two lowest levels of T_n are close, and thus f_n is close to $\ln(4/3)/\ln 2$ (approximately 0.415037). When one level of T_n has many more external nodes than the other, $\mathrm{Var}[X_n]$ is small and thus A_n tends to decrease. As each level of the tree fills up and f_n takes on increasing values in the interval $[0, 1)$, A_n completes one cycle: it moves from $6\ln 2 - 4$ (approximately 0.158883) down to its minimum (for that cycle), then up to its

maximum (for that cycle) and finally back down to $6 \ln 2 - 4$. With each new cycle, A_n more closely approximates its analytic continuation.

A second commonly used consistent strategy is linear search, which, when searching an array of size l, always selects as a probe the data element of rank l (linear search from the bottom) or the data element of rank 1 (linear search from the top). For linear search it is known [see Gonnet and Baeza-Yates (1991)] that

$$\mathbf{E}[C_n] = \frac{n^2}{4} + O(n)$$

and

$$\mathrm{Var}[C_n] \sim \frac{n^3}{36}.$$

Since $s_n^2 = \Omega(n^3)$ and $h_n = n - 1$, the sufficient condition holds for the linear search strategy; thus

$$\frac{C_n - n^2/4}{n^{3/2}} \to_{\mathscr{D}} N\left(0, \frac{1}{36}\right).$$

6. Normality of consistent strategies. In this section we use the sufficient condition for normality to prove that all consistent strategies are normal. Let $\mathscr{S} = \{S_i\}_{i=1}^{\infty}$ be a consistent strategy represented by the decision trees $\mathscr{T} = \{T_i\}_{i=1}^{\infty}$ and let X_i, C_n, s_n^2, L_k, U_k and m_k be as defined. We begin by establishing some properties of consistent strategies.

PROPERTY 1. *For each positive integer i, the decision tree T_i representing an algorithm from a consistent strategy has at least one external node on each unsaturated level.*

The right subtree of the tree in Figure 6, for example, has the following shape characteristic: its left sub-subtree is complete down to level two, while its right sub-subtree has neither internal nor external nodes on level two. The third level of the tree is therefore both unsaturated *and* bereft of external nodes. We will show that this tree cannot be part of a sequence of decision trees representing a consistent strategy.

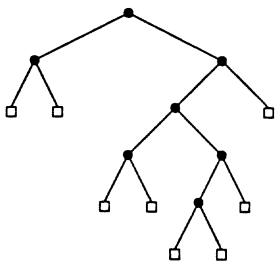

FIG. 6. *An extended decision tree that cannot represent an algorithm from a consistent strategy.*

PROOF OF PROPERTY 1. The first probe specified by S_i, $i = 1, 2, \ldots$, divides an array of size i into two subarrays. Let $g_{\mathscr{S}}(i)$ denote the size of the smaller of the two subarrays and $g'_{\mathscr{S}}(i)$ the size of the larger. [Naturally we define $g_{\mathscr{S}}(0) = g'_{\mathscr{S}}(0) = 0$.] For a cleaner notation, we suppress the subscript $\mathscr{S}$. If T_i had an incomplete level with no external nodes, T_i would contain a subtree whose two sub-subtrees (left and right) had the following property: one sub-subtree would be complete down to some level λ, while the other would have neither internal nor external nodes at level λ. Since S_i is a consistent algorithm, this would imply that for some number $n' \le i$, the size of the subtree,

$$g^{(\lambda)}(n') = 0 \quad \text{and} \quad g^{(\lambda)}(g'(n')) > 0$$

[where $g^{(p)}(i) = g(g(g \cdots (i)))$, the function g composed p times, for a positive integer p]. This is a contradiction, since g must be nondecreasing, but $g'(n') < n'$. □

PROPERTY 2. *For consistent strategies, the sequence* $\{m_k\}_{k=1}^{\infty}$ *is nondecreasing in* k, *so* $U_k/m_k \le k + 1$.

PROOF. Let g and g' be as defined. (Again we suppress the subscript $\mathscr{S}$.) Then for each positive integer i, the larger subtree (rooted at level one) of T_i has the same shape as $T_{g'(i)}$, the decision tree for $S_{g'(i)}$. Since height is monotonic in size, the larger subtree of $T_{L_{k+1}}$—recall that $T_{L_{k+1}}$ is the smallest decision tree in $\mathscr{T}$ having height $k + 1$—has the shape of T_{L_k}, the smallest decision tree in $\mathscr{T}$ whose height is k. So $g'(L_{k+1}) = L_k$. Similarly, $g'(U_{k+1}) = U_k$. Also, by the tree-growing property, if $i_2 > i_1$, then $g'(i_2) \ge g'(i_1)$ and $i_2 - g'(i_2) \ge i_1 - g'(i_1)$. Since for all i, $g'(i) < i$, this implies [taking $i_1 = g'(i)$ and $i_2 = i$] that

$$i - g'(i) \ge g'(i) - g'(g'(i)).$$

Then

$$\begin{aligned} m_{k+1} &= U_{k+1} - U_k \\ &= U_{k+1} - g'(U_{k+1}) \\ &\ge g'(U_{k+1}) - g'(g'(U_{k+1})) \\ &= U_k - U_{k-1} \\ &= m_k. \end{aligned}$$

Since the m_k's are nondecreasing in k,

$$\frac{U_k}{m_k} = \frac{1 + m_1 + m_2 + \cdots + m_k}{m_k} \le k + 1. \qquad \square$$

PROPERTY 3. *For consistent strategies,* $s_n^2 = \Omega(n)$.

PROOF. Suppose $n \geq 100$. First note that, unless T_i (the decision tree illustrating S_i) has at least 95% of its external nodes at a single level, we have $\mathrm{Var}[X_i] \geq 0.0475$. Hence, unless at least 95% of the trees T_i in the subsequence $\{T_i\}_{i=1}^{n}$ have this property, we have $s_{n+1}^2 \geq 0.002375n$.

Now if 95% or more of the trees T_i in $\{T_i\}_{i=1}^{n}$ have this property (that is, at least 95% of their external nodes lie on a single level), then this level, level Λ say, must be the same for all the integers i, $0.10n \leq i \leq n$, for which T_i has this property. This is so because if the level Λ were to shift, say to level $\Lambda + \Delta$ for some integer $\Delta \neq 0$, then $\{T_i\}_{i=1}^{n}$ would contain a subsequence of decision trees having fewer than 95% of their external nodes on any single level, and this subsequence would include more than 5% of the decision trees in $\{T_i\}_{i=1}^{n}$.

Toward a contradiction, suppose T_n has fewer than $0.80n$ external nodes on level Λ. Let

$$i_0 = \max\{i \leq n : T_i \text{ has at least } 0.95i \text{ nodes on level } \Lambda\}.$$

We know that $i_0 \geq 0.95n$, so we may add at most $0.05n$ nodes to T_{i_0} to obtain T_n. If we add all the $n - i_0$ internal nodes to level Λ of T_{i_0} (reducing by as many as possible the number of external nodes on level Λ), the number of external nodes on level Λ of T_n will still be at least $0.8525n$, a contradiction. So T_n must have at least $0.80n$ external nodes on level Λ. Now since $T_{g'(n)}$ is a subtree of T_n and contains at least $0.50n$ external nodes, $T_{g'(n)}$ has at least $0.40g'(n)$ external nodes on level $\Lambda - 1$ (measured from its own root). Let

$$i_1 = \min\{i > g'(n) : T_i \text{ has at least } 0.80i \text{ nodes on level } \Lambda\}.$$

(Note that $i_1 \leq n$ must exist.) Since $T_{g'(n)}$ has at least $0.40g'(n)$ external nodes on level $\Lambda - 1$ and $g'(n) \geq n/2$, the subsequence $\{T_i\}_{i=g'(n)}^{i_1}$ contains more than $0.05n$ trees. This is a contradiction, since all trees in $\{T_i\}_{i=g'(n)}^{i_1}$ have fewer than 95% of their external nodes on level Λ. □

We now present our main result.

THEOREM 1. *All consistent strategies are normal.*

PROOF. If $\lim_{n\to\infty} g(n)/n = 1/2$, the strategy is similar to binary search: the rate of growth in h_n is $O(\ln n)$, while, by Property 3, the rate of growth in s_n^2 is $\Omega(n)$, so the sufficient condition clearly holds. Now assume $\lim_{n\to\infty} g(n)/n = c$ for some constant $c \in [0, 1/2)$. Let h_n be the height and β_n the number of complete levels of a decision tree of size n. We first show that in this case, there are positive constants N_1 and c_1 such that $c_1 < 1$ and for $n > N_1$ we have

$$\beta_n \leq c_1 h_n.$$

If β_n is bounded above by some constant, there is nothing to prove, so assume that $\lim_{n\to\infty} \beta_n = \infty$. (The limit must exist because β_n is nondecreasing in n.) Since $\lim_{n\to\infty} g(n)/n = c < 1/2$, there is a positive constant $c_2 < 1/2$ and an integer N_2 such that for $n > N_2$, $g(n) < c_2 n$. That is, the smaller subtree of a

decision tree of size $n > N_2$ has fewer then $c_2 n$ nodes. Further, each subtree of size $n > N_2$ also has this property; that is, its smaller sub-subtree has at most $c_2 n$ nodes. Thus there is a constant integer c_3 such that for n large enough,

$$g^{(\beta_n - 1)}(n) < c_2^{\beta_n - c_3}\left(\tfrac{1}{2}\right)^{c_3} n,$$

where, by definition, $g^{(k)}(n) = g(g^{(k-1)}(n))$. Since $g^{(\beta_n - 1)}(n) = 1$, we have

$$c_2^{\beta_n - c_3}\left(\tfrac{1}{2}\right)^{c_3} n > 1.$$

Taking logarithms on both sides of the inequality gives

$$(\beta_n - c_3)\ln c_2 - c_3 \ln 2 + \ln n > 0,$$

and, rearranging, we obtain

$$\beta_n < \frac{\ln n}{\ln(1/c_2)} + c_3\left(1 - \frac{\ln 2}{\ln(1/c_2)}\right).$$

Since $h_n \geq \lfloor \log_2 n \rfloor$ and $c_2 < 1/2$, this implies that there are positive constants N_1 and c_1 such that $c_1 < 1$ and for $n > N_1$,

$$\beta_n \leq c_1 h_n.$$

In the notation introduced at the beginning of the previous section, we have shown that there is a constant j_0 such that for $j > j_0$,

$$\beta_{ij} \leq c_1 j,$$

where β_{ij} is the number of complete levels of $T_{L_j + i}$, $i = 0, 1, \ldots, m_j - 1$. By Property 1, $T_{L_j + i}$ has at least one external node on every level below β_{ij}, so for $i = 0, \ldots, m_j - 1$,

$$\begin{aligned}\operatorname{Var}\left[X_{L_j+i}\right] &\geq \frac{1}{L_j + i + 1} \sum_{k=\beta_{ij}}^{j} \left[k - \mathbf{E}\left[X_{L_j+i}\right]\right]^2 \\ &\geq \frac{1}{U_j} \sum_{k'=1}^{j'} [k' - \mu']^2,\end{aligned}$$

where $j' = j - \beta_{ij} + 1$ and $\mu' = \mathbf{E}[X_{L_j+i}] - \beta_{ij} + 1$. Since the sum of squares above may be minimized by setting $\mu' = (j' + 1)/2$, we have

$$\begin{aligned}\operatorname{Var}\left[X_{L_j+i}\right] &\geq \frac{1}{U_j} \sum_{k'=1}^{j'} \left[k' - \frac{j' + 1}{2}\right]^2 \\ &= \frac{1}{U_j}\left\{\frac{j'^3}{12} + O(j'^2)\right\} \\ &= \frac{\Omega(j'^3)}{U_j}.\end{aligned}$$

Recalling that $j' \geq (1 - c_1)j$ for sufficiently large j, we have

$$\mathrm{Var}\left[X_{L_j+i}\right] = \frac{\Omega(j^3)}{U_j}.$$

Then

$$\begin{aligned}\sum_{i=0}^{m_j-1} \mathrm{Var}\left[X_{L_j+i}\right] &= \sum_{i=0}^{m_j-1} \frac{\Omega(j^3)}{U_j} \\ &= \left(\frac{m_j}{U_j}\right)\Omega(j^3) \\ &\geq \left(\frac{1}{j+1}\right)\Omega(j^3) \\ &= \Omega(j^2),\end{aligned}$$

where we have used Property 2 to obtain the inequality. So

$$\begin{aligned}s_{U_j}^2 &= \sum_{k=1}^{j} \sum_{i=0}^{m_k-1} \mathrm{Var}\left[X_{L_k+i}\right] \\ &= \sum_{k=1}^{j} \Omega(k^2) \\ &= \Omega(j^3).\end{aligned}$$

Thus $s_{U_j} = \Omega(j^{3/2})$, which implies that $j = o(s_{L_j})$ or $h_n = o(s_n)$. □

APPENDIX

Normality of other tree-growing search strategies. Properties 1–3, as is clear from their proofs, apply to all tree-growing strategies for which the position of each probe depends only on the length of the data subarray being searched. Here we consider tree-growing strategies which, though not consistent, retain this characteristic and thus possess Properties 1–3. The proof in the previous section clearly works for all such strategies in which the limit $\lim_{n\to\infty} g(n)/n$ does not exist, as long as $\limsup_{n\to\infty} g(n)/n < 1/2$. One example of such a strategy is the following tree-growing implementation of Fibonaccian search. (Not all implementations of Fibonaccian search have the tree-growing property [see Knuth (1973), Section 6.2.1]. We restrict our discussion to tree-growing implementations.) Let the position of the probe to be selected in an array of size n be specified by

$$\xi_1(n) = \begin{cases} n, & \text{if } n \leq 3, \\ n - F_{k(n)-1} + 1, & \text{if } F_{k(n)+1} \leq n \leq F_{k(n)+1} + F_{k(n)-1} - 1, \\ F_{k(n)+1}, & \text{if } F_{k(n)+1} + F_{k(n)-1} \leq n \leq F_{k(n)+2} - 1, \end{cases}$$

where, for $n \geq 3$, $k(n)$ is such that

$$F_{k(n)+1} \leq n < F_{k(n)+2}$$

and F_i is the ith Fibonacci number.

Using this strategy, we "grow" the Fibonacci tree of size $F_{k+1} - 1$ into a Fibonacci tree of size $F_{k+2} - 1$ in the following way. We first add F_{k-1} nodes to the larger subtree of $T_{F_{k+1}-1}$, bringing the size of this subtree to $F_{k+1} - 1$. Then we add F_{k-2} nodes to the smaller subtree, bringing its size to $F_k - 1$. The new Fibonacci tree then has size $F_{k+1} - 1 + F_{k-1} + F_{k-2} = F_{k+2} - 1$. With this strategy we have, using our previous notation,

$$\begin{aligned}
\limsup_{n\to\infty} \frac{g(n)}{n} &= \lim_{k\to\infty} \frac{F_{k-1} - 1}{F_{k+1} - 1} \\
&= \lim_{k\to\infty} \frac{(1/\sqrt{5})\phi^{k-1}}{(1/\sqrt{5})\phi^{k+1}} \\
&= \frac{1}{\phi^2} \\
&= 0.381966\ldots,
\end{aligned}$$

where

$$\phi = \frac{1 + \sqrt{5}}{2},$$

the golden ratio [as in Knuth (1973), page 416]. This strategy is not consistent, because

$$\begin{aligned}
\liminf_{n\to\infty} \frac{g(n)}{n} &= \lim_{k\to\infty} \frac{F_{k-1} - 1}{F_{k+1} + F_{k-1} - 1} \\
&= \lim_{k\to\infty} \frac{\phi^{k-1}}{\phi^{k+1} + \phi^{k-1}} \\
&= \frac{1}{\phi^2 + 1} \\
&= 0.276393\ldots.
\end{aligned}$$

Since we do, however, have $\limsup_{n\to\infty} g(n)/n < 1/2$, this strategy is normal. Note that we may draw this conclusion from our general result, avoiding a somewhat complicated variance calculation.

The following proposition further expands the class of strategies for which we can prove normality.

PROPOSITION 1. *Any search strategy S for which $\liminf_{n\to\infty} g(n)/n = k_1 \in (0, 1/2)$ is normal.*

PROOF. Let $g'(n)$ be as defined. Since $\liminf_{n\to\infty} g(n)/n = k_1 > 0$, there are constants j_0 and k_2 such that $k_2 < 1$ and for $j > j_0$ we have $g'(U_j) \le k_2 U_j$. Then for $j > j_0$,

$$g'^{(j-j_0)}(U_j) \le k_2^{j-j_0} U_j.$$

For example, for $j = j_0 + 2$ we have

$$g'^{(2)}(U_{j_0+2}) = g'\big(g'(U_{j_0+2})\big) = g'(U_{j_0+1}) \le k_2 U_{j_0+1} = k_2 g'(U_{j_0+2}) \le k_2^2 U_{j_0+2}.$$

Since $g'^{(j-j_0)}(U_j) \ge 1$, this implies that

$$1 \le k_2^{j-j_0} U_j.$$

Thus there is some constant $k_3 > 0$ such that for $j > j_0$, $U_j > \exp(k_3 j)$. Writing $j = h_n$ and noting that $U_{j-1} < n$, we have, by Property 3,

$$s_n^2 = \Omega(n) = \Omega\big(\exp(k_3 h_n)\big),$$

so $h_n = o(s_n)$. □

Another implementation of Fibonaccian search provides a practical example of a strategy for which the previous proposition applies. Let the position of the probe to be selected in an array of size n be determined by the function

$$\xi_2(n) = \begin{cases} n, & \text{if } n \le 2, \\ F_{k(n)}, & \text{if } F_{k(n)+1} \le n \le F_{k(n)+1} + F_{k(n)-2} - 1, \\ n - F_{k(n)} + 1, & \text{if } F_{k(n)+1} + F_{k(n)-2} \le n \le F_{k(n)+2} - 1, \end{cases}$$

where $F_{k(n)}$ is as previously defined. This strategy is similar to the one determined by ξ_1 above, except we add nodes to the smaller subtree first, then to the larger. With this strategy we have, using our previous notation,

$$\begin{aligned} \limsup_{n\to\infty} \frac{g(n)}{n} &= \lim_{k\to\infty} \frac{F_k - 1}{F_{k+1} + F_{k-2} - 1} \\ &= \frac{1}{2}. \end{aligned}$$

Here, again, we do not have consistency, since

$$\begin{aligned} \liminf_{n\to\infty} \frac{g(n)}{n} &= \lim_{k\to\infty} \frac{F_{k-1} - 1}{F_{k+1} - 1} \\ &= \lim_{k\to\infty} \frac{\phi^{k-1}}{\phi^{k+1}} \\ &= \frac{1}{\phi^2}, \end{aligned}$$

but the normality of this Fibonaccian strategy follows from Proposition 1.

Many other tree-growing strategies are also normal. Alternating linear search, for example, is clearly normal and shares the asymptotic mean and variance of linear search from the bottom (or top). Intuitively, if a strategy

corresponds to a set of decision trees $\mathscr{T} = \{T_i\}_{i=1}^{\infty}$ for which the height of T_i grows at a steady rate as i grows, the strategy is likely to be normal. In fact, we can prove the normality of all search strategies $\mathscr{S}$ whose corresponding decision trees $\mathscr{T}$ satisfy one of the following sets of conditions:

1. (a) $m_k = O(k^{1+\varepsilon})$ for some ε in the interval $(0, 1)$, (b) $U_k/m_k = O(k^{1+\delta})$ for some δ in the interval $(0, 1)$ and (c) the tree T_i is complete down to a certain level; below this level, T_i has at least one external node on every cth level for some constant c.
2. (a) $m_k = \Omega(k^{1+\varepsilon})$ for some $\varepsilon > 0$ and (b) $\mathrm{Var}(X_n) = \Omega(1)$.

Acknowledgments. The authors gratefully acknowledge the contribution of Sudip Bose of the George Washington University, whose excellent suggestions substantially improved this paper. In addition, the anonymous referees who reviewed the paper offered many helpful comments. The authors are particularly thankful for the proof of Property 3 and the example of a tree-growing strategy that violates the sufficient condition for normality, which one of the referees provided.

REFERENCES

BILLINGSLEY, P. (1986). *Probability and Measure*. Wiley, New York.

DOBERKAT, E. (1982). Asymptotic estimates for the higher moments of the expected behavior of straight insertion sort. *Inform. Process. Lett.* **14** 179–182.

FRITSCH, F., SCHAFER, R. and CROWLEY, W. (1973). Solution of the transcendental equation $we^{w} = x$. *Comm. ACM* **16** 123–124.

GONNET, G. and BAEZA-YATES, R. (1991). *Handbook of Algorithms and Data Structures*, 2nd ed. Addison-Wesley, Reading, MA.

HU, T. and WACHS, M. (1987). Binary search on a tape. *SIAM J. Comput.* **16** 573–590.

KNUTH, D. (1973). *The Art of Computer Programming* **3**. Addison-Wesley, Reading, MA.

PANNY, W. (1986). A note on the higher moments of the expected behavior of straight insertion sort. *Inform. Process. Lett.* **22** 175–177.

DEPARTMENT OF STATISTICS
GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC 20052
E-MAIL: lent_j@bls.gov
hosam@gwis2.circ.gwu.edu