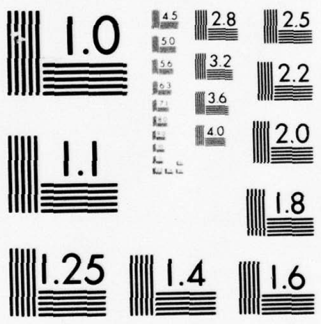


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12
NW

DELETIONS THAT PRESERVE RANDOMNESS

by

Donald E. Knuth

STAN-CS-76-584
DECEMBER 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

See 1473

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(9) Technical rept.,

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. ~~REPORT NUMBER~~
(14) STAN-CS-76-584 ✓

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
(6) DELETIONS THAT PRESERVE RANDOMNESS.

5. TYPE OF REPORT & PERIOD COVERED
technical, December 1976 ✓

6. PERFORMING ORG. REPORT NUMBER
STAN-CS-76-584 ✓

7. AUTHOR(s)
(10) Donald E./Knuth

8. CONTRACT OR GRANT NUMBER(s)
(15) N00014-76-C-0330, NSF-MCS-72-Ø3752

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Stanford University
Computer Science Department ✓
Stanford, Ca. 94305

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
Department of the Navy
Arlington, Va. 22217

12. ~~REPORT DATE~~
(11) Dece[redacted] [redacted]76

13. NUMBER OF PAGES
32 (12) 34p.

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
ONR Representative: Philip Surra
Durand Aeronautics Bldg., Rm. 165
Stanford University
Stanford, Ca. 94305

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Releasable without limitations on dissemination

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

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
analysis of algorithms, binary search trees, data organization, deletions, priority queues

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
→This paper discusses dynamic properties of data structures under insertions and deletions. It is shown that, in certain circumstances, the result of n random insertions and m random deletions will be equivalent to n-m random insertions, under various interpretations of the word "random" and under various constraints on the order of insertions and deletions.←

094120 B

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Deletions That Preserve Randomness

Donald E. Knuth

Computer Science Department
Stanford University
Stanford, California 94305

Abstract

This paper discusses dynamic properties of data structures under insertions and deletions. It is shown that, in certain circumstances, the result of n random insertions and m random deletions will be equivalent to n-m random insertions, under various interpretations of the word 'random' and under various constraints on the order of insertions and deletions.

Index Terms: Analysis of algorithms, binary search trees, data organization, deletions, priority queues.

This research was supported in part by National Science Foundation grant MCS 72-03752 A03, by the Office of Naval Research contract N00014-76-C-0330, and by IBM Corporation. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Deletions that Preserve Randomness

When we try to analyze the average behavior of algorithms that operate on dynamically varying data structures, it has proved to be much easier to deal with structures that merely grow in size than to deal with structures that can both grow and shrink. In other words, the study of insertions into data structures has proved to be much simpler than the study of insertions mixed with deletions. One instance of this phenomenon is described in [5], where what looks like an especially simple problem turns out to require manipulations with Bessel functions, although the data structure being considered never contains more than three elements at a time.

Occasionally an analysis of mixed insertions and deletions turns out to be workable because it is possible to prove some sort of invariance property; if we can show that deletions preserve "randomness" of the structure, in some sense, the analysis reduces to a study of structures built by random insertions. The purpose of this note is to investigate some simple properties which imply various kinds of insensitivity to deletions.

Let us say that a <u>data organization</u> is a class of data structures together with associated algorithms for operating on these structures; for example, consider binary search trees together with algorithms for searching them, inserting into them, and deleting from them. We shall restrict attention to data organizations which depend only on the relative order of the keys of items being inserted and deleted; in other words, if we consider the two data structures formed by the sequence of operations

$$A_1(x_1), A_2(x_2), \ldots, A_n(x_n)$$

$$A_1(y_1), A_2(y_2), \ldots, A_n(y_n)$$

where each $A_k(x)$ means either 'insert an element with key x' or 'delete an element with key x', the two sequences should produce isomorphic data structures if $x_i < x_j$ holds whenever $y_i < y_j$, for all i and j. Such data organization schemes are quite common: Binary search trees with or without height-balancing or weight-balancing [7], 2-3 trees [1], leftist trees [7], and binomial queues [8] all have this property because they operate entirely by making comparisons on keys.

2. Preliminary Definitions.

Let $I(x)$ denote the operation 'insert an element with key x' and let $D(x)$ mean 'delete an element with key x'. We shall be dealing with sequences of operations

$$A_1(x_1), A_2(x_2), \ldots, A_n(x_n)$$

where each A_i is I or D, and where each insertion $I(x_j)$ introduces an element x_j which is distinct from $x_1, \ldots, x_{j-1}$; at least this holds with probability 1 . Furthermore $D(x_j)$ will make sense only if x_j has previously been inserted and not yet deleted; in particular, the number of D's must never exceed the number of I's, counting from left to right.

Since we are assuming that the relative order of keys is all that matters, not their precise value, it suffices to restrict attention to keys $\{1,2,\ldots,n\}$ when there have been n insertions. If $y_1, y_2, \ldots, y_n$ is a sequence of n distinct keys, let

$$\rho(y_1\ y_2\ \ldots\ y_n)$$

be the canonically reordered permutation of $\{1,2,\ldots,n\}$ obtained by mapping the j-th smallest key into the number j . For example, $\rho(8\ \pi\ e\ \sqrt{29}) = 4\ 2\ 1\ 3$.

If $x_1, x_2, \ldots, x_n$ is a permutation of $\{1,2,\ldots,n\}$, we write

$$S(x_1\ x_2\ \ldots\ x_n)$$

for the data structure obtained after the sequence of insertion operations

$$I(x_1)\ I(x_2)\ \ldots\ I(x_n) \ .$$

Finally we write

$$R(x_1\ x_2\ \ldots\ x_n \backslash j) = y_1\ \ldots\ y_{n-1}$$

if the operation of deleting j from $S(x_1\ x_2\ \ldots\ x_n)$ and renumbering yields structure $S(y_1 \ldots y_{n-1})$, where $x_1 x_2 \ldots x_n$ is a permutation of $\{1,2,\ldots,n\}$, $y_1 \ldots y_{n-1}$ is a permutation of $\{1,\ldots,n-1\}$, and $1 \leq j \leq n$.

It is possible for several input permutations to yield the same structure; in other words we might have $S(x_1\, x_2 \ldots x_n) = S(x_1'\, x_2' \ldots x_n')$ although $x_1\, x_2 \ldots x_n \neq x_1'\, x_2' \ldots x_n'$. In this case the R notation is not uniquely defined, since any permutation $y_1 \ldots y_{n-1}$ yielding $S(y_1 \ldots y_{n-1})$ could be used as the value of $R(x_1 x_2 \ldots x_n \backslash j)$. However, we will assume that a particular $y_1 \ldots y_{n-1}$ has been selected in each case. Thus, if $S(x_1 x_2 \ldots x_n) = S(x_1' x_2' \ldots x_n')$ we might wish to define $R(x_1 x_2 \ldots x_n \backslash j) \neq R(x_1' x_2' \ldots x_n' \backslash j)$ even though it will be true that $S(R(x_1 x_2 \ldots x_n \backslash j)) = S(R(x_1' x_2' \ldots x_n' \backslash j))$. Some of these definitions of R will be better than others; a typical theorem to be proved below states that deletions will preserve a certain kind of randomness if it is possible to define the R function in a certain way.

The R function can be used to define a deletion operation on permutations in the following way. Let π be any permutation of n distinct elements (not necessarily integers), and let u be the j-th smallest element of π. Then we define $\pi\backslash u$ (with respect to a given R function) to be the unique permutation of the elements of π other than u such that

$$\rho(\pi\backslash u) = R(\rho(\pi)\backslash j) .$$

For example, let $\pi = 8325$ and $R(4213\backslash 2) = 132$; then $\pi/3 = 285$.

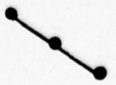
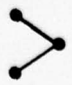
2. Examples.

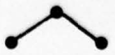
Perhaps the simplest kind of data structure is an unordered linear list, where an insertion is done by simply appending the new element at the right of the list and a deletion is done by deleting the element and closing up the space it occupied. Then $S(x_1 x_2 \ldots x_n)$ is the linear list $\langle x_1, x_2, \ldots, x_n \rangle$, and $R(x_1 x_2 \ldots x_n \setminus j) = \rho(x_1 \ldots x_{k-1} x_{k+1} \ldots x_n)$ where $x_k = j$. In this system the representation preserves all information about the order of insertion.

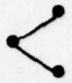
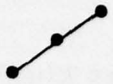
At the other extreme we might consider a sorted linear list, in which $S(x_1 x_2 \ldots x_n) = \langle 1,2,\ldots,n \rangle$ for all permutations $x_1 x_2 \ldots x_n$ of $\{1,2,\ldots,n\}$. In such a system $R(x_1 x_2 \ldots x_n \setminus j)$ can be defined to be any permutation of $\{1,\ldots,n-1\}$ that we wish, as a function of $x_1 x_2 \ldots x_n$ and j.

In between these extremes lie many other regimens, and binary search trees provide a simple but interesting example. This system defines $S(x_1 x_2 \ldots x_n)$ as the empty binary tree if $n = 0$, otherwise $S(x_1 x_2 \ldots x_n)$ consists of a root and two subtrees; the left subtree is $S(\rho(y_1 \ldots y_m))$ and the right subtree is $S(\rho(y'_1 \ldots y'_{n-1+m}))$, where $(y_1 \ldots y_m, y'_1 \ldots y'_{n-1-m})$ are respectively obtained from $x_1 \ldots x_n$ by striking out all elements $(\geq x_1, \leq x_1)$. Furthermore $R(x_1 x_2 \ldots x_n \setminus j) = \rho(x_1 \ldots x_{k-1} x_{k+1} \ldots x_n)$ or $\rho(x_1 \ldots x_{\ell-1} x_{\ell+1} \ldots x_n)$, where $x_k = j$ and $x_\ell = j+1$ if $j < n$; here x_k is deleted if $j = n$ or $\ell < k$, otherwise x_ℓ is deleted.

For example,

$S(123) =$  , $S(132) =$ 

$S(213) = S(231) =$ 

$S(312) =$  , $S(321) =$  ;

and deletion is defined as follows when $n = 3$:

$$R(123\backslash 1) = 12\ , \quad R(123\backslash 2) = 12\ , \quad R(123\backslash 3) = 12\ ;$$
$$R(132\backslash 1) = 12\ , \quad R(132\backslash 2) = 12\ , \quad R(132\backslash 3) = 12\ ;$$
$$R(213\backslash 1) = 12\ , \quad R(213\backslash 2) = 21\ , \quad R(213\backslash 3) = 21\ ;$$
$$R(231\backslash 1) = 12\ , \quad R(231\backslash 2) = 21\ , \quad R(231\backslash 3) = 21\ ;$$
$$R(312\backslash 1) = 21\ , \quad R(312\backslash 2) = 21\ , \quad R(312\backslash 3) = 12\ ;$$
$$R(321\backslash 1) = 21\ , \quad R(321\backslash 2) = 21\ , \quad R(321\backslash 3) = 21\ .$$

3. Deletion Sensitivity and Insensitivity.

From an intuitive standpoint, we wish to show that certain data organizations satisfy theorems of the following general character: "Given a sequence of operations containing n random insertions and m random deletions, in some order, the result is a random data structure on $n-m$ elements; i.e., it has the same probability distribution as we would have obtained by doing $n-m$ random insertions."

The first such theorem was proved by T. N. Hibbard [3], who showed that binary search trees have the following property: create a binary tree by inserting n distinct elements in random order, then delete one of them chosen at random (each being equally likely). Then the resulting tree shapes have the same probability distribution as we would have generated by inserting $n-1$ distinct elements in random order. We shall call this one-step deletion insensitivity, abbreviated " I^*D_r " . (The motivation for this abbreviation will come later; it essentially means "any number of random insertions followed by one random deletion".)

Our definitions make it fairly clear that a data organization has the I^*D_r property if and only if it is possible to define the R function so that, for each n , the $n \cdot n!$ values of $R(x_1 x_2 \ldots x_n \backslash j)$ comprise each of the $(n-1)!$ permutations $y_1 \ldots y_{n-1}$ exactly n^2 times. For example, the tableau above for binary search trees with $n = 3$ shows " 12 " and " 21 " each occurring 9 times.

Proof: The "if" part is obvious. Conversely, consider a data structure σ which is equal to $S(y_1 \ldots y_{n-1})$ for exactly s different permutations $y_1 \ldots y_{n-1}$ of $\{1,\ldots,n-1\}$. The I^*D_r property states that n random insertions followed by one random deletion should produce this data structure with relative frequency s ;

in other words, the $n \cdot n!$ values of $S(R(x_1 x_2 \ldots x_n \backslash j))$ as $x_1 x_2 \ldots x_n$ ranges over the $n!$ essentially different insertions and the n essentially different deletions should include the given structure σ exactly $n^2 s$ times. By redefining $R(x_1 x_2 \ldots x_n \backslash j)$ if necessary we can ensure that each of the s permutations $y_1 \ldots y_{n-1}$ occurs exactly n^2 times.

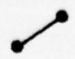
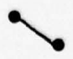
The $I^* D_r$ property might seem to be all that one needs to guarantee insensitivity to any number of deletions, when they are intermixed with insertions in any order. At least, many people (including the present author when writing the first edition of [7]) believed this, and the subtle fallacy in this reasoning was apparently first pointed out by Gary Knott in his thesis [6]. Before we proceed to study stronger forms of deletion insensitivity, it is important to understand why the problem isn't entirely trivial, so we should look at binary trees more closely.

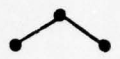
Consider the following process:

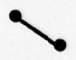
(i) Create a binary search tree by starting with the empty tree and inserting three independent random real numbers, uniformly distributed between 0 and 1 .

(ii) Delete one of these three numbers, selected at random (i.e., each is selected with probability 1/3).

(iii) Insert a fourth independent random real number uniformly distributed between 0 and 1 .

Since the binary search tree organization has the $I^* D_r$ property, we know that the tree remaining after step (ii) will be like a random tree after two insertions; i.e.,  and  will be equally likely. Furthermore it is easy to verify that the element x inserted in step (iii) is equally likely to be smaller than, between, or larger than the two elements remaining after step (ii); for example, the probability that x will be the smallest remaining element is $1/3$. Therefore the insertion in (iii) would seem to behave as the random insertion of a third element into a random two-element tree.

Yet when we analyze carefully what happens after steps (i), (ii), (iii) have been performed, we find that the tree  is obtained with probability $25/72$, not $1/3$. (See [5] for a detailed study of this process.)

The fallacy comes from the fact that the probabilities for the result of step (ii) and the relative position of x in step (iii) are not independent. If we are given the fact that the result of step (ii) was , the conditional probability for element x to be smaller than the two remaining elements turns out to be $13/36$, not $1/3$, since this arises when x is the smallest of all four elements (probability $1/4$) and when x was second smallest but the smallest was deleted (probability $1/4$ times $4/9$, since 4 of the 9 cases where $R(x_1 x_2 x_3 \setminus j) = 12$ have $j = 1$). Therefore, inserting a random element in $[0,1]$ is not equivalent to inserting a number with probability $1/3$ of being smaller than the two remaining, even though a random element in $[0,1]$ does have (unconditional) probability $1/3$ of being smaller than the two remaining.

4. Further Definitions.

The above example shows that deletion insensitivity is not as simple as it may seem at first, so we need to be somewhat careful in our treatment. In this section we shall define several types of insertions and deletions which lead to types of insensitivity that seem to be of importance. The following shorthand notations will prove to be convenient:

I_r stands for insertion of a random real number from some continuous distribution; for example, the distribution might be uniform on the interval $[0,1]$. Each random number inserted is assumed to have the same distribution, and it is to be independent of all previously inserted numbers. Thus, if we look at n such random numbers $x_1, x_2, \ldots, x_n$, the $n!$ possible orderings $\rho(x_1 x_2 \ldots x_n)$ are equally likely, and the particular distribution involved has no effect on the behavior of the data organization.

I_o stands for insertion of a random number by order, in the sense that the new number is equally likely to fall into any of the $d+1$ intervals defined by the d numbers still present as keys after previous insertions and deletions; this is to be independent of the history by which these d numbers were actually obtained. The example in the previous section shows that this is a different concept from I_r ; it is a somewhat artificial kind of random insertion, but it may be a sufficiently good approximation to reality in some applications, and it agrees with I_r before any deletions have taken place.

I_b stands for a "biased" insertion of a random real number obtained as follows: Generate an independent random number X with the exponential distribution, so that $1-e^{-x}$ is the probability that $X \leq x$. Insert the number $X+t$, where t denotes the key most recently deleted (or 0 if there have been no prior deletions). Such insertions arise naturally in priority queue disciplines, where the element with smallest remaining key is always chosen for deletion; the keys can be thought of as specific moments of time when events take place. In this interpretation the exponential deviate X represents a random "waiting time", so that $X+t$ is the time when a newly inserted event will be deleted if the most recent deletion occurred at time t. (Another way to produce biased insertions is to generate a uniform real number X in $[0,1]$ and to multiply it by the most recently deleted key, or by 1 if there were no prior deletions. This corresponds to the above distribution if the largest key is always deleted, since it is essentially isomorphic under the mapping $f(x) = -\log x$.)

D_r stands for a random deletion, in the sense that if d keys are present each is chosen for deletion with probability $1/d$.

D_o stands for a deletion by relative order or rank, in the sense that if d keys are present and if some number j between 1 and d is specified, the j-th smallest element is deleted. Such a j is specified in advance for each deletion.

D_q stands for "priority queue" deletion, the special case of D_o in which j is always equal to 1.

D_a stands for a deletion by relative age, in the sense that if d keys are present and if some number k between 1 and d is specified, the k-th oldest element (the one which has been present k-th longest) is deleted. Such a k is specified in advance for each deletion.

D_f stands for a "fifo" deletion, the special case of D_a in which k is always equal to 1 .

D_ℓ stands for a "lifo" deletion, the special case of D_a in which k is always equal to the current value of d .

Using these abbreviations we shall talk about four different kinds of deletion insensitivity:

I^*D	means any number of insertions followed by one deletion;
I^*D^*	means any number of insertions followed by any number of deletions;
I^*DI^*	means any number of insertions, followed by one deletion, followed by any number of insertions;
$(I,D)^*$	means any number of insertions and deletions, arbitrarily intermixed.

(Of course we also require that deletions never outnumber insertions.) The I 's and D 's will have subscripts to identify their type; for example, $(I_r, D_f)^*$ stands for any number of insertions of random uniform numbers intermixed with fifo deletions. In the first two cases I^*D and I^*D^* , however, no subscript will be given to the I 's since it is easy to see that (I_r, I_o, I_b) are all equivalent until the first deletion has occurred.

We would like to say that a data organization has the $(I_r, D_f)^*$ property if operations $(I_r, D_f)^*$ always produce an essentially random data structure; a data organization might similarly have the I^*D_r property, and so on. These intuitive notions might be formalized as follows, in terms of the R function for that data organization: Consider a sequence of operations

$$A_1(u_1), A_2(u_2), \ldots, A_{m+n}(u_{m+n})$$

which includes n insertions and m deletions; each $A_i(u_i)$ is an insertion or deletion of a given type (e.g., an I_r or a D_f). We define a permutation π_i of the u's remaining after i steps as follows:

π_0 is the null presentation;

if $A_i(u_i)$ is an insertion, π_i is π_{i-1} followed by u_i ;

if $A_i(u_i)$ is a deletion, π_i is the permutation $\pi_{i-1}\backslash u_i$ defined in Section 1 above.

In other words the R function gives us a way to convert deletions on the given data structures to deletions on permutations of the keys. Each permutation π_i has the property that the data structure obtained after i steps is exactly the same as the structure which would be created by inserting the elements of π_i in order from left to right (without deletions).

For example, consider the operation sequence

$$I(0.5), I(0.2), I(0.6), D(0.5), I(0.4)$$

on binary search trees; then

$$\pi_3 = 0.5\ 0.2\ 0.6 \ , \quad \pi_4 = 0.6\ 0.2 \ , \quad \pi_5 = 0.6\ 0.2\ 0.4 \ ,$$

since $\rho(\pi_4) = R(2\,1\,3 \backslash 2) = 21$.

After n insertions and m deletions we will obtain some permutation π_{m+n} of the remaining $n-m$ elements. An R function will be called <u>insensitive</u> to deletions if the elements of π_{m+n} are in random order after such a sequence of random insertions and deletions, i.e., if the resulting permutations $\rho(\pi_{m+n})$ of $\{1,\ldots,n-m\}$ are uniformly distributed.

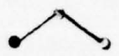
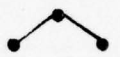
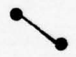
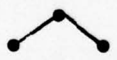
According to this formal definition, it should be clear for example that a data organization has the I^*D_r property if and only if it has an R function which is I^*D_r deletion-insensitive. On the other hand our definition uses only the R function, not the S function, so we are actually distinguishing between different permutations which might yield the same data structure after insertion; we are therefore talking about rather strong forms of deletion insensitivity. This aspect of our model is discussed further in Section 9 below.

5. Reducing the Number of Cases.

With three types of insertions, six types of deletions, and four ways to combine them, we have defined $6+6+18+18=48$ types of deletion insensitivity, namely I^*D_r, I^*D_o, ... ; $I^*D_r^*$, $I^*D_o^*$, ... ; $I_r^*D_rI_r^*$, ... , $I_b^*D_\ell I_b^*$; $(I_r,D_r)^*$, $(I_r,D_o)^*$, ... , $(I_b,D_\ell)^*$. But many of these are uninteresting, since (for example) biased insertions are probably meaningful only in connection with priority queue deletions, type D_q .

It is well-known that the exponential distribution is "memoryless", in the sense that if X is an exponential deviate and if we are told that $X \geq x_0$, the conditional distribution of $X-x_0$ given this knowledge is again exponential. This suggests that $(I_b,D_q)^*$ is actually equivalent to $(I_o,D_q)^*$, a fact which was first proved rigorously by Jonassen and Dahl in their study of priority queue algorithms [4]. Therefore we need not consider I_b any further.

A sequence of random operations with insertions of real numbers can be converted to a discrete probability space in a simple way, because our data organization is assumed to depend only on comparisons between distinct keys. If there are n insertion operations of type I_r , we can assume without loss of generality that the numbers inserted are the integers $\{1,2,\ldots,n\}$ in some order, and that each of the $n!$ permutations is equally likely. On the other hand, suppose that the insertions are of type I_o , where the structure contains respectively $d_1,d_2,\ldots,d_n$ elements just before each insertion; then the number of equiprobable cases to consider is $(d_1+1)(d_2+1)\ldots(d_n+1)$. Furthermore if we are doing m random deletions of type D_r , with respectively $d_1',\ldots,d_m'$ elements present before the deletions, we should multiply the number of equally probable types of insertions by $d_1' \ldots d_m'$, the number of different ways to specify the deleted keys.

For example, consider again the sequence of operations $I_r I_r I_r D_r I_r$ discussed in Section 3; we have $n = 4$, $m = 1$, $d_1' = 3$, so the number of equally probable ways the algorithm might behave is $n! \times d_1' \ldots d_m' = 24 \times 3 = 72$. In 25 of these ways the resulting data structure is  , agreeing with our claim that the probability of this tree is 25/72 . Furthermore it turns out that the final permutation $\rho(\pi_5)$ will be 231 in 13 cases; this confirms that the probability of obtaining  , given that the tree after the deletion was  (i.e., given the 36 cases with $\rho(\pi_4) = 12$), is 13/36 . On the other hand if the operations had been $I_o I_o I_o D_r I_o$, we have $n = 4$, $d_1 d_2 d_3 d_4 = 0122$, $m = 1$, $d_1' = 3$, so the number of equally probable ways the algorithm might behave is $1 \cdot 2 \cdot 3 \cdot 3 \times d_1' \ldots d_m' = 54$. Under this model (which corresponds to the fallacy discussed in Section 3), the tree  occurs with probability 1/3 ; in fact, it is not difficult to prove that the R function given for binary search trees is $(I_o, D_r)^*$ deletion insensitive, by writing down the reasoning which might have led us to believe (fallaciously) that it was $(I_r, D_r)^*$ deletion insensitive.

It is impossible for an R function to be $I_r^* D_q I_r^*$ deletion insensitive. In fact, this is obvious, for the operations $I_r I_r D_q I_r$ lead to six equiprobable values of π_4 (namely 23 , 32 , 23 , 31 , 32 , and 31), hence $\rho(\pi_4) = 12$ with probability 1/3 and $\rho(\pi_4) = 21$ with probability 2/3 . This holds for all R functions, since $R(x_1 x_2 \backslash j)$ is forced to equal 1 . Since D_q is a special case of D_o , no R function can be $I_r^* D_o I_r^*$ deletion insensitive either, much less $(I_r, D_o)^*$. Fortunately such types of insensitivity do not seem to be very important in applications.

The fact that I^*D specifies a smaller class of operations than I^*D^* or I^*DI^*, and that these in turn are smaller than $(I,D)^*$, means for example that

$$(I_r,D_r)^* \Rightarrow I^*D_r^* \Rightarrow I^*D_r \quad ;$$

any R function which is $(I_r,D_r)^*$ insensitive is also $I_r^*D_r^*$, etc. Similarly the fact that D_q is a special case of D_o means that insensitivity under D_o implies insensitivity under D_q ; and insensitivity under D_a implies insensitivity under both D_f and D_ℓ. Furthermore, insensitivity under either D_o or D_a implies insensitivity under D_r, since D_r corresponds to a sum of disjoint cases with the j_i's or k_i's varying in all $d_i!$ possible ways.

Thus there are many obvious implications between the various types of deletion insensitivity which remain to be investigated; and we will find that many of these are actually equivalent to each other.

The first equivalence result is, in fact, obvious:

<u>Lemma 1</u>. Let D be D_r, D_o or D_q. Then

$$(I_o,D)^* \Leftrightarrow I_o^*DI_o^* \Leftrightarrow I^*D^* \Leftrightarrow I^*D \ .$$

<u>Proof</u>. Since $(I_o,D)^* \Rightarrow I^*D^* \Rightarrow I^*D$, and $(I_o,D)^* \Rightarrow I_o^*DI_o^* \Rightarrow I^*D$, it remains to show that $I_o^*D \Rightarrow (I_o,D)^*$; i.e., we need only prove that one-step deletion sensitivity implies full $(I_o,D)^*$ insensitivity. But this is obvious since we can prove that $\rho(\pi_i)$ is uniformly distributed after an I_o insertion if $\rho(\pi_{i-1})$ was, and one-step insensitivity implies that $\rho(\pi_i)$ is uniformly distributed after deletion if $\rho(\pi_{i-1})$ was. Thus $\rho(\pi_i)$ is uniformly distributed for all i . □

Similarly when D is D_a, D_f, or D_ℓ we have $I^*D \Leftrightarrow I_o^*DI_o^*$; but I^*D^* need not be equivalent to these, since the first deletion might "confuse" the ages of the remaining elements. For example, the reader should have little difficulty constructing R functions which are I^*D_ℓ insensitive but not $I^*D_\ell^*$ insensitive.

6. Necessary and Sufficient Conditions.

We can make further progress in understanding deletion insensitivity if we convert the definitions into properties of the R function. Let P_n be the set of all permutations on n elements; and if x is such a permutation, let x_k be its k-th element, from left to right, for $1 \leq k \leq n$. If $x \in P_n$ and $y \in P_{n-1}$, we write

$$[x\backslash j = y]$$

for the function of x , j , and y which is 1 if $R(x\backslash j) = y$, otherwise 0 . In terms of this notation, the following lemma is an immediate consequence of the definitions:

Lemma 2. An R function is

(a) I^*D_r insensitive if and only if $\sum_{\substack{x \in P_n \\ 1 \leq j \leq n}} [x\backslash j = y] = n^2$, for all $y \in P_{n-1}$ and $n \geq 1$;

(b) I^*D_o insensitive if and only if $\sum_{x \in P_n} [x\backslash j = y] = n$, for all $y \in P_{n-1}$ and $n \geq j \geq 1$;

(c) I^*D_q insensitive if and only if $\sum_{x \in P_n} [x\backslash 1 = y] = n$, for all $y \in P_{n-1}$ and $n \geq 1$;

(d) I^*D_a insensitive if and only if $\sum_{x \in P_n} [x\backslash x_k = y] = n$, for all $y \in P_{n-1}$ and $n \geq k \geq 1$;

(e) I^*D_f insensitive if and only if $\sum_{x \in P_n} [x \backslash x_1 = y] = n$, for all $y \in P_{n-1}$ and $n \geq 1$;

(f) I^*D_ℓ insensitive if and only if $\sum_{x \in P_n} [x \backslash x_n = y] = n$, for all $y \in P_{n-1}$ and $n \geq 1$. $\square$

Clearly (b) ⇒ (c), (d) ⇒ (e) and (f), and (b) or (d) ⇒ (a), as we already knew. Furthermore it is easy to see that these are the only implications between the six types; we might have (e) and (f) and (c) but not (a), etc.

The next result is less obvious, possibly even surprising, since it states that a comparatively weak form of deletion insensitivity is equivalent to a comparatively strong property.

Theorem 1. $I^*D_o \Leftrightarrow I_r^*D_rI_r^* \Leftrightarrow (I_r, D_r)^*$.

Proof. Since $(I_r, D_r)^* \Rightarrow I_r^*D_rI_r^*$, we must only show that $I_r^*D_rI_r^* \Rightarrow I^*D_o$ and $I^*D_o \Rightarrow (I_r, D_r)^*$.

Assume first that a given R function is $I_r^*D_rI_r^*$ deletion insensitive, and consider the sequence of operations $I_r^nD_rI_r$ for some fixed n . Any of the $n\cdot(n+1)!$ equally probable realizations of such operations defines a sequence of permutations $\pi_1, \ldots, \pi_{n+2}$ such that $\rho(\pi_{n+2})$ is a uniformly distributed permutation on $\{1,2,\ldots,n\}$; hence every possible permutation $\rho(\pi_{n+2})$ occurs $n(n+1)$ times. Let $\rho(\pi_{n+2}) = y_1 \ldots y_{n-1}y_n$ and $\pi_{n+2} = y_1' \ldots y_{n-1}'y_n'$, where $y_n = j$, and suppose that t is the element missing from $y_1' \ldots y_n'$, where $1 \leq t \leq n+1$; then $y_i = y_i'$ or $y_i'-1$ according as $y_i' < t$ or $y_i' > t$. The number of ways to obtain $\rho(\pi_{n+2}) = y_1 \ldots y_{n-1}y_n$ is

$$\sum_{1\le t\le n+1}\left(\left(\sum_{\substack{x\in P_n\\ t>j}}[x\backslash(t-1)=\rho(y_1\ldots y_{n-1})]\right)+\left(\sum_{\substack{x\in P_n\\ t\le j}}[x\backslash t=\rho(y_1\ldots y_{n-1})]\right)\right)$$

$$=\sum_{\substack{x\in P_n\\ 1\le t\le n}}[x\backslash t=\rho(y_1\ldots y_{n-1})]+\sum_{x\in P_n}[x\backslash j=\rho(y_1\ldots y_{n-1})]$$

$$=n^2+\sum_{x\in P_n}[x\backslash j=\rho(y_1\ldots y_{n-1})]$$

by Lemma 2(a), since R is I^*D_r deletion insensitive; and this equals $n(n+1)$ by assumption. Therefore R is I^*D_o deletion insensitive by Lemma 2(b).

Now assume that a given R function is I^*D_o deletion insensitive, and consider any given sequence of operations $A_1(u_1),\ldots,A_{m+n}(u_{m+n})$ corresponding to n I_r's and m D_r's, where there are respectively $d_1',\ldots,d_m'$ elements present before the deletions. Any of the $n!d_1'\ldots d_m'$ equally probable realizations of such operations defines a sequence of permutations $\pi_1,\ldots,\pi_{m+n}$, where the keys inserted are $\{1,2,\ldots,n\}$ in some order, and we wish to prove that each of the $(n-m)!$ possible values of $\rho(\pi_{m+n})$ occurs $n!d_1'\ldots d_m'/(n-m)!$ times. Let $z_1,\ldots,z_m$ be the elements deleted, so that $\pi_{m+n}z_1\ldots z_m$ is a permutation of the n elements inserted. We will prove that each of these $n!$ permutations occurs exactly $d_1'\ldots d_m'$ times.

In order to avoid cumbersom notations, a single example should suffice to explain the basic idea. Suppose the sequence is

$$I_r\,I_r\,I_r\,I_r\,I_r\,D_r\,I_r\,D_r\,D_r\,I_r\,I_r$$

so that $n = 8$, $m = 3$, $d_1'd_2'd_3' = 554$, and suppose we want to count how many realizations will yield $\pi_{11} = 51637$ and $z_1z_2z_3 = 824$. Working backwards, we must have $\pi_9 = 516$, π_8 = a permutation on $\{1,4,5,6\}$ such that $\pi_8\backslash 4 = \pi_9$, π_7 = a permutation $x_1x_2x_3x_4x_5$ on $\{1,2,4,5,6\}$ such that $\pi_7\backslash 2 = \pi_8$, and π_5 = a permutation on $\{x_1,x_2,x_3,x_4,8\}$ such that $\pi_5\backslash 8 = x_1x_2x_3x_4$. By Lemma 2(b) the number of choices for π_8 is 4 , and for each π_8 there are 5 suitable π_7 's, and for each π_7 there are 5 suitable π_5 's, hence there are $d_1'd_2'd_3'$ solutions. It should be clear that this method of proof is completely general. $\square$

This completes a characterization of deletion insensitivity involving D_r , D_o , and D_q : We have three classes

$$\begin{tikzcd}
(I_o,D_r)^* \arrow[r, Leftrightarrow] & I_o^*D_rI_o^* \arrow[r, Leftrightarrow] & I^*D_r^* \arrow[r, Leftrightarrow] & I^*D_r & & \\
(I_o,D_o)^* \arrow[r, Leftrightarrow] & I_o^*D_oI_o^* \arrow[r, Leftrightarrow] & I^*D_o^* \arrow[r, Leftrightarrow] & I^*D_o \arrow[u, Rightarrow] \arrow[d, Rightarrow] \arrow[r, Leftrightarrow] & I_r^*D_rI_r^* \arrow[r, Leftrightarrow] & (I_r,D_r)^* \\
(I_o,D_q)^* \arrow[r, Leftrightarrow] & I_o^*D_qI_o^* \arrow[r, Leftrightarrow] & I^*D_q^* \arrow[r, Leftrightarrow] & I^*D_q & &
\end{tikzcd}$$

and $(I_r,D_o)^*$, $(I_r,D_q)^*$, $I_r^*D_oI_r^*$, $I_r^*D_qI_r^*$ are impossible.

7. Age-sensitive Deletions.

Let us now consider D_a more closely.

Theorem 2. An R function is $I_r^* D_a I_r^*$ deletion insensitive if and only if, for $1 \leq j,k \leq n$ and all $y \in P_{n-1}$, there exists a unique $x \in P_n$ such that $x_k = j$ and $R(x\backslash j) = y$.

Assume first that a given R function is $I_r^* D_a I_r^*$ deletion insensitive, and consider the sequence of operations $I_r^n D_a I_r$ for some fixed n, where the deletion operation removes the k-th element inserted. Any of the $(n+1)!$ equally probable realizations of such operations defines a sequence of permutations $\pi_1, \ldots, \pi_{n+2}$ such that $\rho(\pi_{n+2})$ is a uniformly distributed permutation on $\{1,2,\ldots,n\}$, hence every possible permutation $\rho(\pi_{n+2})$ occurs $n+1$ times. Now argue as in Theorem 1 with the extra restriction that $x \in P_n$ is such that x_k is the element being deleted; using Lemma 2(d) we find that the number of ways to obtain $\rho(\pi_{n+2}) = y_1 \ldots y_{n-1} y_n$ is

$$n + \sum_{\substack{x \in P_n \\ x_k = j}} [x\backslash j = \rho(y_1 \ldots y_{n-1})]$$

when $y_n = j$, hence the condition in Theorem 2 is necessary.

Conversely, assume that the stated condition holds, and consider a given sequence of operations $I_r^p D_a I_r^{n-p}$ where D_a deletes the k-th element inserted. For example, the sequence might be $I_r I_r I_r I_r I_r D_a I_r I_r$ with $n = 7$, $p = 5$, and $k = 4$. The number of realizations which yield $\pi_8 = 314572$ is the number of permutations x of $\{1,3,4,5,6\}$ such that $x_4 = 6$ and $x\backslash 6 = 3145$; and by hypothesis there is just one such x. There are seven choices of π_8 with $\rho(\pi_8) = 314562$, and each such choice occurs once. This argument clearly generalizes to prove that R is $I_r^* D_a I_r^*$ deletion insensitive. □

<u>Corollary</u>. $I_r^* D_a I_r^* \Rightarrow (I_r, D_r)^*$.

<u>Proof</u>. The condition of Theorem 2 is much stronger than the condition for $I^* D_o$ in Lemma 2(b), since the latter requires only that the equation $R(x \backslash j) = y$ have exactly n solutions when j and y are given. Now apply Theorem 1. □

The condition of Theorem 2 is not strong enough to prove $(I_r, D_a)^*$ insensitivity, which seems to be very strong property indeed. The author has been unable to construct any R functions which are $(I_r, D_a)^*$ insensitive except those which satisfy the following strong requirement:

<u>Condition Q</u>. For each $1 \leq k \leq n$ there exists a permutation $q_1 \ldots q_{n-1}$ of $\{1, \ldots, k-1, k+1, \ldots, n\}$ such that $R(x_1 x_2 \ldots x_n \backslash x_k) = \rho(x_{q_1} \ldots x_{q_{n-1}})$. In other words, deletion of the k-th element inserted will permute the other elements in a way depending only on k , not on their values.

This condition may not be necessary, but it is at least sufficient to prove what we want:

<u>Theorem 3</u>. An R function which satisfies Condition Q is $(I_r, D_a)^*$ deletion insensitive.

<u>Proof</u>. Consider the operation sequence

$$I_r\, I_r\, I_r\, I_r\, I_r\, D_a\, I_r\, D_a\, D_a\, I_r\, I_r$$

where the three D_a's respectively have $k = 2, 3, 4$, and let us count how many realizations will yield $\pi_{11} = 5\,1\,6\,3\,7$ after deleting the elements

824 in this order. Suppose the eight insertions are $z_1z_2z_3z_4z_5z_6z_7z_8$ respectively, a permutation of $\{1,2,\ldots,8\}$; we will show that the z's are uniquely determined by these assumptions. For concreteness, let us suppose that some of the permutations implied by Condition Q are $x_1x_2x_3x_4x_5\backslash x_2 = x_3x_1x_5x_4$, $x_1x_2x_3x_4x_5\backslash x_4 = x_2x_5x_3x_1$, and $x_1x_2x_3x_4\backslash x_2 = x_3x_1x_4$. Then we know that $\pi_5 = z_1z_2z_3z_4z_5$, $z_2 = 8$, $\pi_6 = z_3z_1z_5z_4$, $\pi_7 = z_3z_1z_5z_4z_6$, $z_4 = 2$, $\pi_8 = z_1z_6z_5z_3$, $z_6 = 4$, $\pi_9 = z_5z_1z_3$, $\pi_{11} = z_5z_1z_3z_7z_8 = 51637$; hence $z_1 \ldots z_8 = 18625437$. This argument clearly generalizes to prove the theorem, since there is always a unique realization for each choice of π_{m+n} and the sequence of elements deleted, for each choice of k's in the D_a operations. $\square$

There is an interesting way to weaken Condition Q to obtain a somewhat weaker kind of deletion insensitivity, yet one which is stronger than that of Lemma 2(d):

<u>Condition Q_0</u>. For each $1 \leq k \leq n$ there exists a sequence of n permutations $(q_{1,1} \ldots q_{1,n-1}), \ldots, (q_{n,1} \ldots q_{n,n-1})$ of $\{1,\ldots,k-1,k+1,\ldots,n\}$ with the following property: For all $y \in P_{n-1}$ there exists a permutation $p_1 \ldots p_n$ of $\{1,\ldots,n\}$, possibly depending on y , such that

$$\rho(x_1 \ldots x_{k-1}x_{k+1} \ldots x_n) = y \text{ and } x_k = j \text{ implies}$$

$$R(x_1 \ldots x_n \backslash x_k) = \rho(x_{q_{p_j},1} \ldots x_{q_{p_j},n-1}) .$$

In other words, when we delete the k-th element inserted the result is one of n specified permutations of the remaining elements; and if the remaining elements are held fixed, while x_k runs through all n possible values relative to them, the results run through these n specified permutations in some order. Condition Q implies Condition Q_o , since the n specified permutations might be identical.

This rather peculiar condition seems to be just what is needed to prove the following slightly weakened form of Theorem 3.

Theorem 4. An R function which satisfies Condition Q_o is $(I_o, D_a)^*$ deletion insensitive.

Proof. As in the previous proofs, it is most convenient to consider a more-or-less random example which is sufficiently general to be convincing without the introduction of elaborate notation. Consider the operation sequence

$$I_o\, I_o\, I_o\, I_o\, I_o\, D_a\, I_o\, D_a\, D_a\, I_o\, I_o$$

where the three D_a's have $k = 2, 4, 4$, respectively. There are $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 5 \cdot 4 \cdot 5$ realizations of this sequence; we will show that $5 \cdot 5 \cdot 4$ of them will yield any given value of $\rho(\pi_{11})$. For example, suppose $\rho(\pi_{11}) = 3\,1\,4\,2\,5$. There are 5 choices for the q-permutation in the first deletion; let us choose one of these, and assume for example that the deletion takes $\pi_5 = z_1 z_2 z_3 z_4 z_5$ into $\pi_6 = z_1' z_2' z_3' z_4' = z_3 z_1 z_5 z_4$. In other words, one of the q-permutations for $n = 5$ and $k = 2$ is assumed to be $3\,1\,5\,4$. (If $3\,1\,5\,4$ occurs as two or more of the q-permutations we also choose the subscript j such that $(3,1,5,4) = (q_{j,1}, q_{j,2}, q_{j,3}, q_{j,4})$; thus, there are 5 distinct choices possible even when the q-permutations are not distinct.) Then if $\pi_7 = z_1' z_2' z_3' z_4' z_5'$ we must delete the 4-th oldest element, which is z_3' (since it equals z_5); again we have 5 q-permutations to choose from, and let us suppose that the q-permutation for the second deletion yields

$\pi_8 = z_1''z_2''z_3''z_4'' = z_4'z_1'z_5'z_2'$; we similarly shall choose one of the 4 q-permutations now available for the last deletion and suppose that it yields $\pi_9 = z_1'''z_2'''z_3''' = z_2''z_4''z_1''$.

To make $\rho(\pi_{11}) = 31425$ we now can work backwards and identify the relative sizes of various elements: Since $\rho(\pi_9) = 213$, we know that $\rho(z_1''z_2''z_4'') = 321$. This value of y together with Condition Q_o allows us to determine $\rho(\pi_8)$, since each possible value of z_3'' relative to z_1'', z_2'', z_4'' corresponds to one of the predetermined choices of q-permutation once $\rho(z_1''z_2''z_4'')$ is known. In our case $\rho(\pi_8)$ must be 4312 , 4321 , 4231 , or 3241 , and our choice of q-permutation subscript tells us which of these occurs, say 4231 ; then $\rho(z_1'z_2'z_4'z_5') = 2143$ and we can similarly reconstruct $\rho(\pi_7)$, which might be 21354 . In the same way $\rho(\pi_6) = 2134$ implies that $\rho(z_1z_3z_4z_5) = 1243$; and we can use this knowledge to find $\rho(\pi_5)$, say 21354 . Each I_o insertion has now been characterized, thus each of our $5 \cdot 5 \cdot 4$ choices has led to a unique realization such that $\rho(\pi_{11}) = 31425$. □

Although I_o is a somewhat artificial type of random insertion, Theorem 4 is interesting because $(I_o,D_a)^*$ insensitivity implies $I^*D_a^*$ insensitivity, and this special case is not artificial.

Let us conclude our theoretical investigations by considering briefly the fifo and lifo deletion types, D_f and D_ℓ . If the R function satisfies

$$R(x_1x_2 \dots x_n \backslash x_1) = \rho(x_2 \dots x_n)$$

it is obviously $(I_r,D_f)^*$ insensitive; note that this condition might hold even though neither Condition Q nor Q_o are satisfied, in fact the weak condition of Lemma 2 (a) might not even hold. On the other hand when the R function does not satisfy the above formula, there appear to be no interesting conditions which guarantee $I^{*} D_{f}^{*}$ insensitivity, other than the condition Q_o we have already discussed. (We might have, say, $R(x_1 x_2 \ldots x_n \backslash x_1) = \rho(x_3 x_2 x_4 \ldots x_n)$ and $R(x_1 x_2 \ldots x_n \backslash x_2) = \rho(x_1 x_3 \ldots x_n)$; these conditions lead to $(I_r, D_f)^{*}$ insensitivity without the full generality of Condition Q, but they don't seem to be very interesting.) Essentially the same remarks hold also for lifo-deletions, if R does or does not satisfy

$$R(x_1 \ldots x_{n-1} x_n \backslash x_n) = \rho(x_1 \ldots x_{n-1}) \ .$$

8. Applications.

Let us finally apply these theorems to some important data organizations. Sorted and unsorted linear lists have every possible type of insensitivity to deletions, but this is obvious without the above theory.

Binary search trees provide what is perhaps the most interesting application. We have already mentioned that Hibbard [3] originated this theory by essentially proving that the R function defined in Section 2 above is I^*D_r insensitive. Knuth [7, answer to exercise 6.2.2-13] observed that it is in fact I^*D_a insensitive, and then Knott [6] went much further, proving that Hibbard's R function is $(I_o, D_a)^*$ insensitive. In particular, if we do n random insertions, followed by $m < n$ fifo-deletions, the resulting tree has the shape distribution of a binary tree after $n-m$ random insertions. This is a difficult theorem to prove, perhaps the "deepest" result about a data structure which had been obtained by anyone before 1975.

It is possible to establish Knott's theorem using the above theory; in fact, much of that theory was motivated by what he did. We want to show that the binary search tree organization satisfies Condition Q_o. Let $k \le n$ be given, and for $1 \le \ell \le n$ let

$$q_{\ell,1} \ldots q_{\ell,n-1} = \begin{cases} 1 \ldots (k-1)(k+1) \ldots n \; , & \text{if } \ell \le k \; ; \\ 1 \ldots (k-1)\ell(k+1) \ldots (\ell-1)(\ell+1) \ldots n \; , & \text{if } \ell > k \; . \end{cases}$$

Let $y = y_1 \ldots y_{n-1} \in P_n$ be given, and let $z_1 \ldots z_{n-1}$ be the inverse permutation, so that $y_{z_j} = j$. It is not difficult to verify that Condition Q_o holds with the permutation defined by

$$p_j = \begin{cases} z_j \; , & \text{if } z_j < k \; ; \\ z_j+1 \; , & \text{if } z_j \ge k \; ; \\ k \; , & \text{if } j = n \; . \end{cases}$$

Thus binary search trees are $(I_o, D_a)^*$ insensitive to deletions using Hibbard's method. On the other hand we have seen that they are not $(I_r, D_r)^*$ insensitive, so by Theorem 1 they are not even $I^* D_o$ insensitive.

Suppose we define deletion in a different way, essentially by interchanging left and right in Hibbard's method: Let $R(x_1 x_2 \ldots x_n \setminus j) = \rho(x_1 \ldots x_{k-1} x_{k+1} \ldots x_n)$ or $\rho(x_1 \ldots x_{\ell-1} x_{\ell+1} \ldots x_n)$ where $x_k = j$ and $x_\ell = j-1$ if $j > 1$, and where x_k is deleted if $j = 1$ or $\ell < k$, otherwise x_ℓ is deleted. (For example, this changes $R(1\,3\,2\setminus 1)$, $R(3\,1\,2\setminus 3)$ to 21 and $R(2\,1\,3\setminus 2)$, $R(2\,3\,1\setminus 2)$ to 12 in the table of Section 2.) This function is $(I_o, D_a)^*$ insensitive to deletions, and it also satisfies Lemma 2(c) so it is $(I_b, D_q)^*$ insensitive as well. Furthermore, like Hibbard's function it possesses $(I_r, D_\ell)^*$ insensitivity. We can also verify $(I_b, D_q, D_\ell)^*$ insensitivity, if the I_b insertions are biased by the most recent D_q (not D_ℓ) deletion. (Is it $(I_b, D_q, D_a)^*$ insensitive in this sense?)

Jean Vuillemin [8] has recently defined a useful type of data organization which he calls binomial queues, and Mark Brown [2] has shown that they are highly insensitive to deletions. In fact, Brown proved that the corresponding R function satisfies Condition Q, hence it is $(I_r, D_a)^*$ and $(I_r, D_r)^*$ insensitive.

The leftist tree structures developed in 1971 by Clark Crane (see [7, Section 5.2.3]) unfortunately do not share such nice properties. In fact, the corresponding function $R(x_1 x_2 x_3 x_4 \setminus j)$ has a pronounced bias towards $3\,2\,1$ and $2\,3\,1$ except when $j = 1$, and the function $R(x_1 x_2 x_3 x_4 x_5 \setminus 1)$ is extremely biased. Therefore leftist trees are quite sensitive to deletions, and it will probably be very difficult to analyze them. In fact, the analysis for pure insertions is already very formidable. Similar remarks apply to balanced trees.

9. Degeneracy.

We have defined deletion insensitivity only in terms of the R function, but when many different permutations lead to the same data structure (i.e., if they yield the same S value) it might be possible to have deletion insensitivity that cannot be carried back to any R function for the organization. For example, when the data structure consists of a sorted linear list, the S function is essentially constant, so we trivially have $(I_r, D_o)^*$ insensitivity; but we have observed that no R function can have this property.

In other words the conditions we have derived in Theorems 1 and 2 are sufficient but not necessarily necessary for insensitivity. An example can be given of a data organization which is $I_r^* D_r I_r^*$ insensitive when the S equivalences are considered, yet it is not $I^* D_o$ insensitive: Let $R(x_1 \ldots x_n \backslash x_k) = \rho(x_1 \ldots x_{k-1} x_{k+1} \ldots x_n)$ for $n \neq 3$, $R(x_1x_2x_3\backslash 1) = 12$, $R(x_1x_2x_3\backslash 2) = 21$, $R(123\backslash 3) = R(132\backslash 3) = R(231\backslash 3) = 12$, $R(213\backslash 3) = R(312\backslash 3) = R(321\backslash 3) = 21$; and let $S(x_1 \ldots x_n) = S(y_1 \ldots y_n)$ if and only if $x_1 \ldots x_n = y_1 \ldots y_n$ or $n \geq 3$ and $x_4 \ldots x_n = y_4 \ldots y_n$ and $S(\rho(x_1x_2x_3)) = S(\rho(y_1y_2y_3))$, where $S(132) = S(231)$ and $S(312) = S(321)$. The operations $IIID_q$ leave a nonrandom result; but $I_r^n D_r I_r^m$ clearly produces a random structure when $n \neq 3$, and this can be verified also for $n = 3$. Thus Theorem 1 is not true when we take the S equivalences into account.

It appears unlikely that any conditions weaker than those discussed in the above lemmas and theorems will be useful for proving deletion insensitivity in practice. Furthermore it is not difficult to see that the existence of an R function satisfying the six respective conditions in Lemma 2 is, in fact, both necessary and sufficient for the six corresponding kinds of I^*D insensitivity. (We proved this for I^*D_r in Section 3.)

[1] A. V. Aho, J. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, (Reading, Mass.: Addison-Wesley, 1974), x + 470 pp.

[2] Mark R. Brown, "Implementation and analysis of binomial queue algorithms," submitted for publication.

[3] Thomas N. Hibbard, "Some combinatorial properties of certain trees with applications to searching and sorting," J.ACM 9 (1962), 13-28.

[4] Arne Jonassen and Ole-Johan Dahl, "Analysis of an algorithm for priority queue administration," Math. Inst., Univ. of Oslo (1975).

[5] Arne Jonassen and Donald E. Knuth, "A trivial algorithm whose analysis isn't," submitted for publication.

[6] Gary D. Knott, "Deletion in binary storage trees," Ph.D. thesis, Computer Science Department, Stanford University (May 1975), 93 pp.

[7] Donald E. Knuth, The Art of Computer Programming, Vol. 3, Sorting and Searching, (Reading, Mass.: Addison-Wesley, 1973).

[8] Jean Vuillemin, "A data structure for manipulating priority queues," Comm. ACM, to appear.