Deletion algorithms for binary search trees

Conrado Martinez
Xavier Messeguer

Report LSI-90-39
Deletion Algorithms for Binary Search Trees*

- C. Martínez  X. Meseguer
Dept. Lenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028-Barcelona
e-mail: conrado@lsi.upc.es, meseguer@lsi.upc.es

October 19, 1990

Abstract

The effect of updating (deletions/insertions) on binary search trees has been an interesting research topic for almost three decades, but in the last five years there have been a few contributions, due partially to the intrinsic difficulty of the involved analysis. Since the problem is quite difficult to be solved in a general fashion, we have restricted ourselves to solve a simpler problem, which shall be considered as an important and necessary basis for further developments.

In this paper, we have faced the systematization of the study of deletion algorithms, and deduced the effect that a single random deletion produces in the probability distribution of binary trees of arbitrary size for a wide variety of deletion algorithms. Furthermore, to carry on the analysis, some new tools have been introduced, such as the concepts of strong and weak invariance of probability functions induced by an algorithm. Among others, we have been able to derive interesting results such as an extension of Hibbard’s classical theorem and sufficient conditions under which a complexity measure of main practical importance, the expected search time, does not change.

1 Introduction.

For many years the study of the behavior under dynamical updating of binary search trees has been a topic of interest. One of the first results in the area, and in fact in "pure computer science" was obtained by T.N. Hibbard, who showed that entries could be deleted dynamically without difficulty and that a random deletion in a random tree left a random tree [9]. In 1975 G.D. Knott proved an interesting result close to that of Hibbard [11], but at the same time pointed out a surprising paradox, which came up to be known as Knott’s paradox: if a random insertion is performed in a tree resulting from a random deletion it does not follow that it yields a random tree.

Two years later, the classical paper from A.T. Jonassen and D.E. Knuth [10] presented an exact analysis of the behavior of the probability function of binary search trees after multiple deletion/insertion pairs were applied, for trees of size 3. A similar work is due to R. Baeza-Yates [2], who extended the results to binary search trees of size 4. Both papers show that this analysis is very intricate and difficult, and could hardly be extended to general cases, i.e. arbitrary size and updating patterns.

On the other hand, extensive empirical studies due to J.L. Eppinger [5] and J. Culberson [3] suggested that after a large number of random deletion/insertion pairs were applied to random trees of size \( n \) the average internal path length (IPL) get worse, about \( \Omega(n \log^3 n) \), in front of the expected IPL of random trees of size \( n \), which is \( \Omega(n \log n) \) [12]. These studies contradicted the previous ones done by G.D. Knott, who conjectured that IPL was not degraded by arbitrary updating patterns. Finally, in 1985, Culberson [4]

---

*This research was partially supported by the ESPRIT BRA Program of the EC under contract no. 3075, project ALCOM.
demonstrated that things will get even worse showing that the expected IPL after the application of $O(n^2)$ deletion/insertion pairs to trees of size $n$ was $O(n\sqrt{n})$.

Classical deletion algorithms, such as Hibbard's or its modified version proposed by Knuth, are of asymmetric nature and degrade expected search time. Consequently, interest has been focused towards the behaviour of symmetric algorithms. For instance, the empirical results obtained by Eppinger and Culberson and the theoretical ones of X. Messeguer [14] and Baeza-Yates, suggest that if we use a symmetric algorithm, IPL after updating does not differ substantially from "random" IPL. Nevertheless, few contributions have appeared in the last five years, due partially to the intrinsic difficulty of the involved analysis.

It continues to be an open problem if the symmetric algorithms guarantee the preservation of the fundamental characteristics or if there is any other deletion algorithm which has this property. Since the problem is very difficult we feel that it should be investigated in an step-by-step strategy. The present paper is concerned with the development of the tools and concepts which shall prove useful in order to confront the above mentioned problem.

Since the permutation model here assumed (only relative ordering of inserted keys is considered) is not adequate to analyze the dynamic updating of binary search trees, we cannot investigate other effects but the ones produced by single deletions. Therefore a new model, where the relative ordering of inserted keys and the full updating history are took into account, should be introduced to deal with the more general situation.

Moreover, some interesting results can be derived by means of those tools, even in the limited scope here proposed.

As before said, we restrict ourselves to analyze the effects of a single random deletion, but we do it for a wide variety of deletion algorithms. After a preliminary discussion of the probability model for binary search trees in Section 2, and in order to perform the analysis, we propose a parameterized random algorithm for deletion in Section 3, in the same spirit as in [14]; in this way, most known deletion algorithms are particular cases of it, by just setting the appropriate values for the parameters.

Section 4 is devoted to the analysis of the behaviour of the prior algorithm, deriving a probability function which describes it.

We have been able to derive an exact formula for the probability that a given tree (its shape) has of being produced after a random deletion (Section 5), and to characterize the cases where this probability is the same as the probability after random insertions only (Subsection 6.1). This last result constitutes a generalization of Hibbard's theorem. We also investigate in Subsection 6.2 sufficient conditions to obtain weak invariance, a concept introduced in [14], which guarantees the preservation after random deletions of important measures such as the expected IPL, height, etc.

In Section 7 the expected IPL after random deletions is considered with more detail.

2 The Probabilistic Model for BSTs.

Binary search trees (BSTs, for short) and their balanced variants are well known data structures widely used for insertion, deletion and query operations (dictionary operations). For a general covering on the topic of binary search trees we refer the reader to [1, 8, 12, 15]. Most of the notation used follows that of Knuth.

A binary search tree of size $n$ can be constructed by a sequence of $n$ insertion operations, say $INS(k_1)$, $INS(k_2)$, ..., $INS(k_n)$, where $INS(k_i)$ means "insert key $k_i$". We shall assume that $k_1, \ldots, k_n$ are $n$ distinct keys drawn from a set $K$, and they can be totally ordered: thus, either $k_i < k_j$ or $k_j < k_i$ for $i \neq j$.

Since the characteristics of the resulting BST and the behavior of the operations do not depend on the actual value of the keys but only in its relative ordering [6, 12], the analysis can be carried on by considering as thought BSTs were built from permutations over $\{1, \ldots, n\}$. In this way, the set of ordered $n$-tuples with no repetitions of $K$ is partitioned into classes whose elements exhibit identical behavior from the point of view of insertion, deletion and search; each class consists of the elements having the same
relative ordering. The canonical representative of a class is therefore, the permutation corresponding to
the relative ordering that identifies the class.

This equivalence principle has been applied currently for the analysis of this data structure and its
related algorithms, and in particular, permits to avoid labelling the tree nodes. Given a tree of size \( n \), with
\( i \) keys in its left subtree, and consequently \( n - 1 - i \) keys in the right subtree, the root will be occupied
by the key \( i + 1 \); keys 1 to \( i \) appear in the left subtree and keys \( i + 2 \) to \( n \) in the right subtree. This
scheme can be recursively applied to both subtrees, provided that we relabel the right subtree as thought
key \( i + 2 \) is the first, \( i + 3 \) the second and so on.

If the \( n \) keys to be inserted are drawn independently and without replacement from some uniform
distribution, then the previous permutation model can be used, with each permutation being equally
likely \([6, 7]\) (that is, having probability \( 1/n! \)). This is a common assumption in most analysis, and we
shall adopt it.

Observe that two different permutations can give raise to the same BST, so there is no bijection
between the symmetric group of rank \( n \) and the set of binary trees of size \( n \).

Therefore, the probability model induced for the BSTs generated from these equally likely permutations
is not uniform (i.e. trees of the same size may not have the same probability). This probability model has
been classically characterized in the following way: given a tree of size \( n \), formed by two subtrees of sizes
\( X \) and \( n - 1 - X \) respectively, \( X \) is a discrete random variable taking values in the range \([0, \ldots, n - 1]\)
with

\[
\text{Prob}(X = i) = \frac{1}{n} \quad (2.1)
\]

An alternative way of viewing this probabilistic model is to consider the underlying splitting process \([7]\).
We will describe it in an informal way, and derive a useful recursive expression to compute the probability
of a given BST.

Suppose we are said to construct a BST of size \( n > 0 \), and we have at our disposal \( n \) urns, containing
trees of sizes \( 0, \ldots, n - 1 \) and distributed accordingly to the probabilistic model considered. We start
selecting randomly any of the \( n \) urns; for instance, the urn containing trees of size \( i \). From this urn we will
pick a subtree, so the size of this subtree is \( i \) with probability \( 1/n \) as Eq. (2.1) states. Let \( P(T_i) \) be the
probability of the tree \( T_i \) drawn from the \( i \)-th urn. Now, we choose a tree of size \( n - 1 - i \), say \( T_2 \), whose
probability is \( P(T_2) \). If we denote by \(|T|\) the size of a tree \( T \) then we have, assuming the independence
of the three involved events,

\[
P(T) = \frac{P(T_i) \cdot P(T_2)}{n} = \frac{P(T_i) \cdot P(T_2)}{|T|}, \quad n > 0 \quad (2.2)
\]

where \( T \) is the tree constructed by attaching a root node, \( T_i \) as the left subtree and \( T_2 \) as the right subtree.
We will write this as \( T = T_i \bigwedge T_2 \).

For the recursion basis, recalling the fact that there’s only a tree of null size or empty tree, denoted
by \( \Box \), we have \( P(\Box) = 1 \).

In the sequel, we denote the left subtree and right subtree of any tree \( T \) by \( T_l \) and \( T_r \), respectively.
Moreover, the set of binary trees will be denoted \( B \) and the set of binary trees of size \( n \) \( B_n \).

3 A Probabilistic Algorithm for Deletion.

In this section we will define a probabilistic algorithm for deletion that works over non-empty BSTs. To
delete key \( k \) from a non-empty tree \( t \) we use the algorithm DELETE\( t \), assuming w.l.o.g. that \( k \) is actually
in \( t \) and where \( \Omega = < \alpha, \beta, \gamma, \beta', \gamma' > \) is a 5-tuple of probabilistic parameters, with values ranging in the
real interval \([0, 1]\) and whose purpose is to drive the algorithm’s flow (see Figure 1).

We now explain it. Once the key \( k \) is found we face one of the four possible cases schematically shown
in Figure 2, to delete it. Our first step is to draw a random uniformly distributed number \( \delta \) over the real
interval \([0, 1]\), by means of the procedure random\_compute.
function DELETE₀(k: key.type; t: BST) returns BST is
  if
    k < key(t) then left.son(t) := DELETE₀(k, left.son(t))
    k > key(t) then right.son(t) := DELETE₀(k, right.son(t))
    k = key(t) then
      random.compute(δ);
      if
        right.son(t) = [] = left.son(t)
      then t := []
        right.son(t) = [] ≠ left.son(t)
      then if
          δ ≥ β then t := left.son(t)
          δ < β then key(t) := greatest_key(left.son(t))
          left.son(t) := DELETE_MAX₀(left.son(t))
        endif
        right.son(t) ≠ [] = left.son(t)
      then if
          δ ≥ γ then t := right.son(t)
          δ < γ then key(t) := smallest_key(right.son(t))
          right.son(t) := DELETE_MIN₀(right.son(t))
        endif
        right.son(t) ≠ [] ≠ left.son(t)
      then if
          δ ≥ α then key(t) := greatest_key(left.son(t))
          left.son(t) := DELETE_MAX₀(left.son(t))
          δ < α then key(t) := smallest_key(right.son(t))
          right.son(t) := DELETE_MIN₀(right.son(t))
        endif
      endif
  endif
endfunction

Figure 1: Algorithm DELETE₀.
The first is a trivial one: simply get rid of the key.

In the second case, let $j$ and $l$ be respectively the root and the greatest key of the left son of $k$ in $t$, where $k$ is the key to be deleted (see Figure 3). There are two options for deleting key $k$ which are chosen with probabilities $1 - \beta$ and $\beta$, respectively.

(a) get up the left son of $k$, replacing the node $k$ by the node labelled $j$.

(b) find $l$ the greatest key of the left son of $k$, to put it in the root, and delete $l$ from its previous place.

This new deletion has to be made over a tree without right son (because $l$ is the greatest key). Now, we confront only two cases: the trivial one, if $l$ has no sons, or a similar situation to the one here discussed. Therefore, we introduce a new deletion algorithm, DELETE.MAX$_{\beta'}$, given in Figure 4 which decides, if $l$ has a left son, whether to perform the procedure described in the case (a) or to recursively apply the procedure here presented. The election is governed by a new parameter $\beta'$, whose purpose is the same as $\beta$'s. The introduction of two distinct parameters is due to our intention of characterizing some classical algorithms which in fact operate in this way.

The third case of Figure 2 is symmetrical to the second case, with right subtrees playing the rôle of left subtrees, and where there is an algorithm to delete minimum keys, DELETE.MIN$_{\gamma'}$, instead of DELETE.MAX$_{\beta'}$. We denote $\gamma$ and $\gamma'$ the parameters involved, which operate in a similar way to that of $\beta$ and $\beta'$.

Just now we have introduced four probabilistic parameters. The fourth case gives raise to the fifth parameter, denoted $\alpha$. When we want to delete a node which has two sons, we must choose from which subtree shall be searched for a substitute to replace the key in the root: the greatest of the left son or the smallest of the right one. The $\alpha$ parameter suggests the election.

The purpose of each parameter is schematically shown in Figure 5. Most known deletion algorithms are particular cases of this algorithm, each one characterized by giving particular values to the parameters (see Table 1).

### 4 Recursive Characterization of the Deletion Algorithm.

This section is devoted to the characterization of the probability of obtaining a given tree $T$ as the result of deleting a key $x$ from a tree $t$, by means of the algorithm DELETE$_0$. This has been accomplished by associating probability functions depending on $x$ and $t$ to DELETE$_0$, so the application of the appropriate function to a tree $T$ yields the desired probability. This result constitutes the basis for the deduction of the probability distribution that is induced after a single random deletion, in Section 5.

In Subsection 4.1 we introduce the definitions and notation which we need to perform the subsequent analysis, while in Subsection 4.2 the announced characterization is obtained.
4.1 Some Previous Definitions and Notation.

In order to perform the analysis of the algorithm discussed in Section 3 we shall make frequent use of probability functions (PFs) defined over binary trees of a given size. For instance,

\[ f : B_n \rightarrow [0, 1], \quad \text{with} \quad \sum_{T \in B_n} f(T) = 1 \]

Even a single tree \( t \) of size \( n \) will denote a PF: \( t \) is the PF such that

\[ t(T) = \begin{cases} 1 & \text{if } T = t \\ 0 & \text{otherwise} \end{cases} \]

Given any PF \( f \) defined over \( B_n \), the support set of the function \( f \), is

\[ \text{sup}(f) = \{ T \in B_n \mid f(T) > 0 \} \]
function \textsc{DELETE.MAX}_{\beta'}(k : key.type; t : BST) returns BST is
\begin{algorithmic}
\State while right.son(t) \neq \emptyset do t := right.son(t) endwhile;
\If{left.son(t) = \emptyset}
\State t := \emptyset
\EndIf
\If{left.son(t) \neq \emptyset}
\State random.compute(\delta)
\If{\delta \geq \beta'}
\State t := left.son(t)
\Else
\State left.son(t) := \textsc{DELETE.MAX}_{\beta'}(left.son(t))
\EndIf
\EndIf
\Endfunction

Figure 4: Algorithm \textsc{DELETE.MAX}_{\beta'}.

![Diagram](image)

Figure 5: Scheme of algorithm \textsc{DELETE}_{\Omega}.

Since \( B_n \) is finite, the support of any such function is also finite. Let \( \text{sup}(f) = \{T_1, \ldots, T_k\} \) and \( f_t = f(T_t), \quad t = 1, \ldots, k \). By the previous convention, we can write \( f \) as
\[ f = f_1 T_1 + f_2 T_2 + \ldots + f_k T_k \]

There are two operations of interest for us which apply to these PFs. The first one is the linear combination of PFs.

**Definition 4.1** The linear combination of two PFs \( f \) and \( g \), both defined over \( B_n \), denoted by \( [af + bg] \) is the PF defined over \( B_n \) given by
\[ [af + bg](T) = a \cdot f(T) + b \cdot g(T) \quad (4.1) \]
with \( 0 \leq a, b \leq 1 \) and \( a + b = 1 \).

This operation is meaningful only if both \( f \) and \( g \) are mappings over \( B_n \), as before said, and it is straightforward to verify that linear combination of PFs gives a new PF.

The other useful operation is the tree combination or tree product of PFs.

**Definition 4.2** Given two PFs \( f \) and \( g \), defined over \( B_n \) and \( B_m \) respectively, their tree combination, denoted \( [f \bigtriangleup g] \), is a function defined over \( B_{n+m+1} \), such that for trees \( T \in B_{n+m+1} \) with \( |T| = n \) and \( |T'| = m \), has value
\[ [f \bigtriangleup g](T) = f(T) \cdot g(T') \quad (4.2) \]
and is null otherwise.

We do not require $f$ and $g$ to be PFs over the same set to define its tree combination, as for the linear combination. Finally, it is easily seen that the tree combination of PFs produces a PF.

Underlying Definitions 4.1 and 4.2, as well as the notation introduced is the concept of formal trees series \([\mathcal{F}]\), although we will use them in quite a different way.

Bearing these definitions in mind, the behavior of the deletion algorithm $\text{DELETE}_\Omega$ can be described by a set of recurrences over PFs, using linear and tree combinations.

### 4.2 Probability Function Associated to the Deletion Algorithm.

By $\text{DEL}_\Omega(z, t, T)$ we mean the probability that, given a tree $t \neq {}$ and a key $z$, algorithm $\text{DELETE}_\Omega$ produces the tree $T$ when this key is deleted. Since we are assuming the permutation model the position of the key in the inorder traversal of $t$ equals its actual value, so $z$ ranges between 1 and $|t|$. $\text{DELMAX}_\Omega(t, T)$ and $\text{DELMIN}_\Omega(t, T)$ are defined in the same way.

Moreover, let $\text{DEL}_\Omega(z, t)$ be the probability function defined over the set of binary trees with one node less than $t$, such that,

$$
\text{DEL}_\Omega(z, t) : \mathcal{B}_{|t|-1} \rightarrow [0, 1]
$$

In other words,

$$
\text{DEL}_\Omega(z, t, T) = \{\text{DEL}_\Omega(z, t)\}(T)
$$

For instance,

$$
\text{DEL}_\Omega(2, \begin{array}{c}
\circ \\
\circ \\
\end{array}) = \alpha \cdot \begin{array}{c}
\circ \\
\circ \\
\end{array} + (1 - \alpha) \cdot \begin{array}{c}
\circ \\
\circ \\
\end{array}
$$

It is obvious that $\text{DEL}_\Omega(z, t)$ is not defined for $t = {}$ and that

$$
\text{DEL}_\Omega(1, \begin{array}{c}
\circ \\
\circ \\
\end{array}) = {}
$$

Similarly, $\text{DELMIN}_\Omega(t)$ and $\text{DELMAX}_\Omega(t)$ are the PFs corresponding to deletion of the minimum ($z = 1$) and maximum ($z = |t|$) of a tree $t$, and Eq. (4.3) applies to them.

Now, we proceed with a case analysis of $\text{DEL}_\Omega(z, t)$, considering the subtrees of $t$, $t_l$ and $t_r$, and the value of $z$ (see Figure 6), which leads to the following Proposition:

**Proposition 4.1** The probability function $\text{DEL}_\Omega(z, t)$ verifies,

(a) $z$ is in the left subtree of $t$ : $z \leq |t_l|$.  

$$
\text{DEL}_\Omega(z, t_l, t_r) = \text{DEL}_\Omega(z, t_l, t_r)
$$

(b) $z$ is in the right subtree of $t$ : $z > |t_l| + 1$.  

$$
\text{DEL}_\Omega(z, t_l, t_r) = t_l \cdot \text{DEL}_\Omega(z - |t_l| - 1, t_r)
$$

(c) $z$ is the root of $t$ : $z = |t_l| + 1$.  

(c.1) both $t_l$ and $t_r$ are non-empty : $|t_l| > 0, |t_r| > 0$.  

$$
\text{DEL}_\Omega(z, t_l, t_r) = (1 - \alpha) \cdot \text{DELMAX}_\Omega(t_l, t_r) + \alpha \cdot t_l \cdot \text{DELMIN}_\Omega(t_r)
$$
(c.2) \( t_i \) is not empty, but \( t_r \) is : \(|t_i| > 0, |t_r| = 0\).

\[
DEL_{\Omega}(z, t_i) = \beta \cdot DELMAX_{\Omega}(t_i) + (1 - \beta)t_i
\]  

(4.7)

(c.3) \( t_r \) is not empty, but \( t_i \) is : \(|t_i| = 0, |t_r| > 0\).

\[
DEL_{\Omega}(z, t_r) = \gamma \cdot \bigwedge DELMIN_{\Omega}(t_r) + (1 - \gamma)t_r
\]  

(4.8)

(c.4) both \( t_i \) and \( t_r \) are empty : \(|t_i| = |t_r| = 0\).

\[
DEL_{\Omega}(z) = \emptyset
\]  

(4.9)

Note that we shall substitute \( z \) by \( z - |t_i| - 1 \) in case (b) because of the implied re-labelling (see Section 2). On the other hand, case (c.4) is nothing else than Eq. (4.3), and constitutes the recursion basis.

This proposition is the direct translation of the behavior of the algorithm DELETE_\( \Omega \) to recurrences over the associated probability functions. For instance, Eq. (4.6) just states that the result of deleting the key at the root of a tree having both subtrees, is a tree which results when this key is replaced by the maximum key at its left subtree and its right subtree remains unchanged with probability \( 1 - \alpha \), or a tree which results from the replacement of the key by its successor (the minimum key at its right subtree) leaving the same left subtree with probability \( \alpha \).

\( DELMIN_{\Omega} \) and \( DELMAX_{\Omega} \) can be characterized in a similar way. In fact, case (a) applies to \( DELMIN_{\Omega} \) and (b) to \( DELMAX_{\Omega} \). Only case (c.2) and (c.4) could happen when we are actually deleting the maximum key, so Eqs. (4.7) and (4.9) work properly for \( DELMAX_{\Omega} \), replacing \( \beta \) by \( \beta' \) and \( DEL_{\Omega} \) by \( DELMAX_{\Omega} \).

(b) the maximum occurs in the right subtree of \( t \): \(|t| > |t_i| + 1\).

\[
DELMAX_{\Omega}(t_i, t_r) = t_i \bigwedge DELMAX_{\Omega}(t_r)
\]  

(4.10)

(c) the maximum is the root of \( t \): \(|t| = |t_i| + 1 \) or equivalently, \(|t_r| = 0\).

(c.2) \( t_i \) is not empty : \(|t_i| > 0\).

\[
DELMAX_{\Omega}(t_i, t_r) = \beta' \cdot DELMAX_{\Omega}(t_i) + (1 - \beta')t_i
\]  

(4.11)

(c.4) \( t_i \) is empty : \(|t_i| = 0\).

\[
DELMAX_{\Omega}(t_i, t_r) = \emptyset
\]  

(4.12)

Analogously, Eqs. (4.4), (4.8) and (4.9) cover the case of actual deletion of the minimum, interchanging the rôles of \( \beta' \) and \( \gamma' \), and the rôles of left and right subtrees.

A simple translation of the previous equations, gives us recursive expressions for the value of the probability function when applied to a given tree, that is, for \( DEL_{\Omega}(z, t, T) \), \( DELMAX_{\Omega}(t, T) \) and \( DELMIN_{\Omega}(t, T) \). To exemplify how it can be done, we translate the recurrence associated to case (c.1):

(c.1) \( z \) is the root of \( t \) where \(|t_i| > 0 \) and \(|t_r| > 0\).

\[
DEL_{\Omega}(z, t, T) = \left[ (1 - \alpha) DELMAX_{\Omega}(t_i, T_i) t_r(T_r) + \alpha DELMIN_{\Omega}(t_i, T_i) T_t(T_i) \right]
\]  

(4.13)
5 The Probabilistic Distribution After a Random Deletion.

We are interested in the probabilistic distribution of binary trees of arbitrary size after deletion of one key randomly chosen. This probability, for a given tree $T$, is the sum, for all trees with one more node than $T$, and for all possible keys, of the probability of obtaining $T$ from the tree and the key considered, multiplied by the probability of choosing them:

$$P_{\Omega}(T) = \sum_{\lambda \in \mathcal{B}_{[|T|]+1}^{[|T|+1]}} \Delta E L_\Omega(z, t, T) P(t) \text{Prob}(z) \quad (5.1)$$

where $P(t)$ and $\text{Prob}(z)$ are the probability of $t$ and the probability of deleting key $z$. Note that the joint probability of selecting $(z, t)$ as input to DELETE$\Omega$ has been split into a product of probabilities, because we assume that they are independently drawn. Furthermore, the probability of selecting a particular $z$ is the same for all values, so $\text{Prob}(z) = \frac{1}{|T|+1}$, and we have

$$P_{\Omega}(T) = \frac{1}{|T|+1} \sum_{\lambda \in \mathcal{B}_{[|T|]+1}^{[|T|+1]}} \Delta E L_\Omega(z, t, T) P(t)$$

As we have seen in the preceding section, the probability function $\Delta E L_\Omega(z, t)$ is recursively defined depending on the shape of the tree $t$, so the first thing we must do is to split the sum according to the
above criterion:

\[ P_n(T) = \frac{1}{n+1} \sum_{i \in B_{n+1}} \sum_{1 \leq z \leq n+1} DEL_0(z,t,T) P(t) = \]
\[ = \frac{1}{|T|+1} \left[ (a) \sum_{i \in B_{n+1}} \sum_{1 \leq z < |T_i|+1} DEL_0(z,t,T) P(t) + \right. \]
\[ + \sum_{i \in B_{n+1}} \sum_{1 \leq z < |T_i|+1} DEL_0(z,t,T) P(t) + \]
\[ + (c.1) \sum_{i \in B_{n+1}} \sum_{1 \leq z < |T_i|+1} DEL_0(|T_i|+1,t,T) P(t) + \]
\[ + (c.2) \sum_{i \in B_{n+1}} \sum_{1 \leq z < |T_i|+1} DEL_0(|T_i|+1,t,T) P(t) + \]
\[ + (c.3) \sum_{i \in B_{n+1}} \sum_{1 \leq z < |T_i|+1} DEL_0(1,t,T) P(t) + \]
\[ + (c.4) \sum_{i \in B_{n+1}} \sum_{1 \leq z < |T_i|+1} DEL_0(1,t,T) P(t) \] \quad (5.2)

where \( n = |T| \).

We have denoted all terms in the splitting using the case decomposition already seen in Subsection 4.2 for clarity. Moreover, this splitting has been represented in Figure 6.

Cases (a) and (b) are defined recursively in terms of \( P_0(T) \) and \( P_0(T_i) \), but cases (c.1), (c.2) and (c.3) require the introduction of two new functions:

\[ MAX_\rho(T) = \sum_{i \in B_{n+1}} DELMAX_\rho(t,T) P(t) \]
\[ MIN_\rho(T) = \sum_{i \in B_{n+1}} DELMIN_\rho(t,T) P(t) \] \quad (5.3)

where \( 0 \leq \rho \leq 1 \) and whose recursive form is given by the following Lemma:

**Lemma 5.1** For any \( \rho \), with \( 0 \leq \rho \leq 1 \),

\[ MAX_\rho(T) = \frac{1}{|T|+1} \left[ P(T) \ast MAX_\rho(T) + \rho \square(T) \ast MAX_\rho(T) + (1 - \rho) P(T) \right] \] \quad (5.4)
\[ MIN_\rho(T) = \frac{1}{|T|+1} \left[ P(T) \ast MIN_\rho(T) + \rho \square(T) \ast MIN_\rho(T) + (1 - \rho) P(T) \right] \] \quad (5.5)

with \( MIN_\rho(\square) = MAX_\rho(\square) = 1 \).

The proof of this lemma appears in Appendix 8.

Moreover, the following corollary can be established without difficulty:

**Corollary 5.1** \( MIN_\rho(T) = MAX_\rho(T) = P(T) \)

It can be readily proved by induction on the size of the tree.

For each case we find a recurrence relation, in terms of \( P \), \( P_{i1} \) and, eventually of \( MAX_\rho \) and \( MIN_\rho \). Proof sketches can also be found in Appendix 8
(a) \[ \frac{1}{(|T| + 1)^2} P(T_r) P_{n}(T_l) (|T| + 1) \] \hspace{1cm} (5.6)

(b) \[ \frac{1}{(|T| + 1)^2} P(T_l) P_{o}(T_r) (|T| + 1) \] \hspace{1cm} (5.7)

(c.1) \[ \frac{1}{(|T| + 1)^2} \left[ (1 - \alpha) (1 - \Box(T_r)) P(T_r) \text{MAX}_{\beta'}(T_l) + \alpha (1 - \Box(T_l)) P(T_l) \text{MIN}_{\gamma'}(T_r) \right] \] \hspace{1cm} (5.8)

(c.2) \[ \frac{1}{(|T| + 1)^2} \left[ \beta \Box(T_r) \text{MAX}_{\beta'}(T_l) + (1 - \beta) P(T) \right] \] \hspace{1cm} (5.9)

(c.3) \[ \frac{1}{(|T| + 1)^2} \left[ \gamma \Box(T_l) \text{MIN}_{\gamma'}(T_r) + (1 - \gamma) P(T) \right] \] \hspace{1cm} (5.10)

(c.4) \[ \Box(T) \] \hspace{1cm} (5.11)

Now, we can state the following theorem:

**Theorem 5.1** If \( n + 1 \) keys are randomly inserted in an initially empty tree, and one of them, randomly chosen, is deleted using \( \text{DELET}_{E_{ni}} \), the probability of the tree shape with \( n \) nodes produced is given by

\[
P_{n}(T) = \frac{1}{(|T| + 1)^2} \left[ P(T_r) P_{n}(T_l) (|T| + 1) + P(T_l) P_{o}(T_r) (|T| + 1) + (1 - \alpha) (1 - \Box(T_r)) P(T_r) \text{MAX}_{\beta'}(T_l) + \alpha (1 - \Box(T_l)) P(T_l) \text{MIN}_{\gamma'}(T_r) + \beta \Box(T_r) \text{MAX}_{\beta'}(T_l) + (1 - \beta) P(T) + \gamma \Box(T_l) \text{MIN}_{\gamma'}(T_r) + (1 - \gamma) P(T) \right], \quad T \neq \Box
\] \hspace{1cm} (5.12)

and \( P_{n}(\Box) = 1 \).

The theorem is proved by just summing up Eqs. (5.6) to (5.11), as Eq. (5.2) indicates. The above equation can also be written as

\[
P_{n}(T) = \frac{1}{(|T| + 1)^2} \left[ P(T_r) P_{n}(T_l) (|T| + 1) + P(T_l) P_{o}(T_r) (|T| + 1) + (2 - \beta - \gamma) P(T) + \text{MAX}_{\beta'}(T_l) \left( (1 - \alpha) (1 - \Box(T_r)) P(T_r) + \beta \Box(T_l) \right) + \text{MIN}_{\gamma'}(T_r) \left( \alpha (1 - \Box(T_l)) P(T_l) + \gamma \Box(T_r) \right) \right], \quad T \neq \Box
\] \hspace{1cm} (5.13)

6 Invariance Theorems.

Consider two families of sets \( E = \{ E_n \}_{n \geq 0} \) and \( S = \{ S_n \}_{n \geq 0} \) and an algorithm \( A \) which given an input \( e \) from \( E_n \) produces an output \( s \in S_n \). Let \( P_{n}^e \) and \( P_{n}^s \) be probability functions defined over \( E_n \) and \( S_n \). If \( A \) is deterministic we shall write \( A(e) = s \), but if not, we ought characterize the probability of any element in \( S \) of being produced by \( A \), i.e. \( \text{Prob}(A(e) = s) \).

Then, algorithm \( A \) induces a probability distribution over \( S_n \), denoted \( P_{n,A} \), which is

\[
P_{n,A}(s) = \sum_{e \in E_n} P_{n}^e(e) \text{Prob}(A(e) = s), \quad s \in S_n, \ n \geq 0
\] \hspace{1cm} (6.1)

12
In fact, Eq. (5.1) corresponds to the above equation, for the deletion algorithm \( \text{DE}L_0 \); an input consists of a tree and a position ranging from 1 to the tree's size, and output of a tree with one less node. The probability model \( P \) has been discussed in Section 2, while \( P' \) can be derived from the independence and randomness assumptions, and \( P \).

Let \( P = \{P_n\}_{n \geq 0} \) and analogously define \( P_A = \{P_{n,A}\}_{n \geq 0} \).

Definition 6.1 A family of induced probability functions \( P_A \) is strong invariant or equivalently, \( A \) induces strong invariance over \( P \) if and only if

\[
P_{n,A}(s) = P_n(s), \quad \text{for any } s \in S_n, \quad \text{for any } n \geq 0
\]

or, shortly, if \( P = P_A \).

Other important concept is that of weak invariance, introduced in [14]. Let \( \equiv \) be any equivalence relation, \( S_n/\equiv \equiv \) the set of classes defined by the equivalence relation over set \( S_n \), and \( [s] \) denote the class to which \( s \) belongs, where \( s \) is any element of \( S_n \).

Next, define \( P_n^\equiv([s]) \) to be \( \sum_{s' \in [s]} P(s') \) and \( P_{n,A}^\equiv([s]) \) to be the sum of the induced probabilities.

Definition 6.2 A family of induced probability functions \( P_A \) is weak invariant with respect to \( \equiv \) (also, \( \equiv \)-weak invariant) if and only if

\[
P_{n,A}^\equiv([s]) = P_n^\equiv([s]), \quad \text{for any } [s] \in S_n/\equiv, \quad \text{for any } n \geq 0
\]

Strong invariance is a particular form of weak invariance, taking \( \equiv \) to be the identity. Obviously, strong invariance implies any form of weak invariance, and guarantees that average properties of the elements of \( S \) after algorithm application don't vary from the original ones.

On the other hand, if \( C \) is a complexity measure defined over \( S_n \) such that \( C(s) = C(s') \) for any two elements \( s \) and \( s' \) in a class of equivalence, and algorithm \( A \) induces weak invariance for the same equivalence criterion, then the expected value of \( C \), after \( A \)'s application is preserved.

\[
\overline{C}_n = \sum_{s \in S_n} C(s) P_n(s) = \sum_{[s] \in S_n/\equiv} \sum_{s' \in [s]} C(s') P_n(s') =
\]

\[
= \sum_{[s] \in S_n/\equiv} \sum_{s' \in [s]} P_n(s') = \sum_{[s] \in S_n/\equiv} C(s) P_n^\equiv([s]) =
\]

\[
= \sum_{[s] \in S_n/\equiv} C(s) P_{n,A}^\equiv([s]) = \sum_{[s] \in S_n/\equiv} \sum_{s' \in [s]} P_{n,A}(s') =
\]

\[
= \sum_{[s] \in S_n/\equiv} \sum_{s' \in [s]} C(s') P_{n,A}(s') = \sum_{s \in S_n} C(s) P_{n,A}(s) =
\]

\[
= \overline{C}_{n,A}, \quad n \geq 0
\]

For instance, the IPL of a binary tree and its symmetric is the same, and selecting the appropriate values for \( \Omega \), we can obtain weak invariance with respect to symmetry for \( \text{P}_0 \). Therefore, a deletion satisfying the w.i. conditions would not modify the average IPL. This also holds for other measures such as height, number of nodes in the fringe, etc.

It is interesting to point out, that from this point of view, Hildbrand's theorem stated that his proposed algorithm induces strong invariance. Nevertheless, the input model was unaware of the keys actually inserted, so in fact, the theorem only states weak invariance with respect to the equivalence relation \( \text{SHAPE} \) (all trees with the same shape are equivalent), if the input model considered cares of shapes and contents as should do. We feel that this qualitative reasoning captures the essence of Knott's paradox.

For the same reason, in our subsequent analysis, it shall be taken into account that the induced probability calculated in the previous section is for tree shapes, and consequently the theorem to be presented in Subsection 6.1 also suffers Knott's paradox.

A rather different but complete treatment of strong invariance\(^1\) under deletions of various kinds (not only random) in a wide class of data structures can be found in [13].

\(^1\)It is denominated "insensitivity" in the cited paper.
6.1 Strong Invariance. Hibbard’s Theorem Revisited.

In this section, we will consider which values of $\alpha, \beta, \ldots$ lead to an strong invariant induced probability function, i.e. $P(n)(T) = P(T)$ for any tree $T$.

The key observation is that if $DELMIN_{\beta'}$ and $DELMAX_{\gamma'}$ make no recursive calls ($\beta' = \gamma' = 0$) then we have $MIN_{\gamma'}(T) = MAX_{\beta'}(T) = P(T)$ (Lemma 5.1).

Now, we will find sufficient conditions for strong invariance by assuming that $P(1) = P$ for trees up to a given size $n - 1$, as the induction hypothesis. Let $T$ be any tree of size $n > 0$. Suppose, moreover, that $\beta' = \gamma' = 0$, and recalling that $P(T)P(T') = |T| \cdot P(T)$ (Eq. (2.2)), Eq. (5.13) becomes,

$$
P(n)(T) = \frac{1}{(|T| + 1)^2} \left[ |T| \cdot |T| + 1 \right] P(T) + P(T)(2 - \beta - \gamma) +
\begin{align*}
  &+ (1 - \square(T)) \alpha \cdot |T| \cdot P(T) + \\
  &+ (1 - \square(T)) \cdot (1 - \alpha) \cdot |T| \cdot P(T) + \\
  &+ (1 - \square(T)) \cdot \beta \cdot P(T) + \\
  &+ (1 - \square(T)) \cdot \gamma \cdot P(T)
\end{align*}
$$

(6.2)

If the four possible cases of subtrees being empty or not are considered then it follows that Eq. (6.2) equals $P(T)$ if and only if

$$
\beta + \gamma = 1 \\
\alpha = \gamma
$$

(6.3)

Since the induction basis is true, $P(1)(\square) = P(\square) = 1$, we have demonstrated the following theorem

Theorem 6.1 If $n + 1$ items are inserted into an initially empty binary tree, in random order, and if one of these, selected at random, is deleted using $DELM$ with $\Omega$ satisfying conditions (6.3) and $\beta' = \gamma' = 0$, then the probability that the resulting binary tree has a given shape is the same as the probability that this tree shape would be produced by inserting $n$ items into an initially empty tree in random order.

Theorem 6.1 is a generalization of Hibbard’s classical result [9]; in fact, the deletion algorithm proposed by Hibbard can be characterized by setting $\alpha = \gamma = 1$ and $\beta = \beta' = \gamma' = 0$ thus satisfying the conditions of our theorem.

6.2 Weak Invariance.

We shall study weak invariance for the symmetry equivalence relation. We start defining which is the symmetric tree of a tree.

Definition 6.3 Given any tree $T$, its symmetric, denoted $\text{sym}(T)$ is

$$
\text{sym}(t_i \oslash t_j) = \text{sym}(t_j \oslash t_i)
$$

with $\text{sym}(\square) = \square$.

This enables us to define a equivalence relation, called $SYM$, where each class consists in a tree and its symmetric (only one tree if $T = \text{sym}(T)$).

Next, we define the symmetrical probabilistic operator

$$
P_{SYM}(\cdot) = P(\cdot) + P(\text{sym}(\cdot))
$$

(6.4)

corresponding to sum of probabilities of the elements in a class or twice the probability for singleton classes (so it is not exactly what we have defined in our previous discussion on weak invariance) and similarly $P_{SYM}(\cdot)$ is $P(n)(\cdot) + P(n)(\text{sym}(\cdot))$.

Before we deduce the recurrent relations for $P_{SYM}$, we write some trivial identities about the symmetric relation:
(a) $\text{sym}(T)_i = \text{sym}(T_r) \mathbf{and} \text{sym}(T)_r = \text{sym}(T_l)$.  

(b) $|\text{sym}(T)| = |T|$.  

(c) $P(\text{sym}(T)) = P(T)$.  

(d) $MAX_\rho(\text{sym}(T)) = MIN_\rho(T)$ and $MIN_\rho(\text{sym}(T)) = MAX_\rho(T)$, for any $0 \leq \rho \leq 1$.

Beginning with the Eqs. (5.1) and (6.4), and applying the last identities results

$$P_{\Omega}^{SYM}(T) = \frac{1}{(|T|+1)^2} [ P(T_r) P_{\Omega}^{SYM}(T_l)(|T_l|+1) + P(T_l) P_{\Omega}^{SYM}(T_r)(|T_r|+1) + (1-\square(T_l)) P(T_r) MAX_{\rho'}(T_l) + (1-\square(T_l)) P(T_l) MIN_{\gamma'}(T_r) + (1-\square(T_l))(\beta MAX_{\rho'}(T_l) + \gamma MAX_{\rho'}(T_l)) + (1-\square(T_l))(\beta MIN_{\gamma'}(T_r) + \gamma MIN_{\gamma'}(T_r)) + 2(2-\beta-\gamma) P(T) ]$$

where the dependence of $\alpha$ has vanished.

To simplify the last equation, we make $\mu = \beta + \gamma$ and impose $\beta' = \gamma'$:

$$P_{\Omega}^{SYM}(T) = \frac{1}{(|T|+1)^2} [ P(T_r) P_{\Omega}^{SYM}(T_l)(|T_l|+1) + P(T_l) P_{\Omega}^{SYM}(T_r)(|T_r|+1) + (1-\square(T_l)) P(T_r) MAX_{\rho'}(T_l) + (1-\square(T_l)) P(T_l) MIN_{\gamma'}(T_r) + \mu \square(T_l) MAX_{\rho'}(T_l) + \gamma \square(T_l) MIN_{\gamma'}(T_r) + 2(2-\mu) P(T) ]$$

As in the preceding subsection, making $\beta' = \gamma' = 0$ makes $MAX_{\rho'} = MIN_{\gamma'} = P$. By an inductive proof it's easy to see that if $\mu = 1$ then $P_{\Omega}^{SYM} = P^{SYM}$, so

$$\mu = 1$$
$$\beta' = \gamma' = 0$$

are sufficient conditions for the SYM-weak invariance of the deletion algorithm to hold.

7 Expected Behavior of Internal Path Length After a Random Deletion.

Let $IPL_n$ denote the average internal path length of a BST of size $n$ built after $n$ random insertions, and $IPL(T)$ the internal path length of $T$. Let $B$ be the set of binary trees. Then,

$$I(z) = \sum_{n \geq 0} IPL_n z^n = \sum_{T \in B} IPL(T) P(T) z^{|T|} = -2 \ln(1-z) + z \frac{1}{(1-z)^2}$$
and
\[
\overline{IPL_n} = [z^n]I(z) = 2nH_n + 2H_n - 4n = 2n \ln n - (4 - 2\gamma)n + 2\ln n + (2\gamma + 1) + O \left( \frac{1}{n} \right)
\]
where \(\gamma \approx 0.57721\ldots\) is Euler's constant \(^2\) and \(H_n\) denotes the \(n^{th}\) harmonic number.

If \(\Omega\) is such that \(P_\Omega\) is strong or weak invariant for symmetry (see Section 6) then average IPL remains unchanged, but if it is not there will be subtle variations of the IPL. Of course, removing a single key from a large tree will not modify substantially its IPL, so the average IPL after a random deletion should be \(O(n \ln n)\).

Let \(\overline{IPL_{\Omega,n}}\) the expected IPL for a tree of size \(n\) produced by a random deletion using \(DEL_\Omega\) in a random tree of size \(n + 1\). We define
\[
I_\Omega(z) = \sum_{n \geq 0} \overline{IPL_{\Omega,n}} z^n = \sum_{T \in B} IPL(T) P_\Omega(T) z^{|T|} 
\tag{7.1}
\]

Eqs. (2.2), (5.12), (5.4), (5.5) and the definition of \(IPL(T)\) lead to
\[
\theta^2 I_\Omega(z) = \frac{2z}{1-z} \theta I_\Omega(z) + \frac{3-z}{(1-z)^2} I(z) + \frac{2(2-\mu)}{1-t} \int_0^t \frac{I(t)}{1-t} dt + \theta^2 \left[ \frac{z^2}{(1-z)^2} \right] + \left[ \beta + (1 - \alpha) \frac{z}{1-z} \right] K_\beta(z) + \left[ \gamma + \alpha \frac{z}{1-z} \right] K_\gamma(z) 
\tag{7.2}
\]

where \(\theta F = \frac{d}{dz}(z \cdot F)\), \(\mu = \beta + \gamma\) and \(K_\nu(z)\) is the solution of
\[
\frac{dK_\nu}{dz} - \left( \frac{1}{1-z} + \nu \right) K_\nu(z) = \left( \frac{1}{1-z} - \nu \right) I(z) + \nu \frac{z^2}{(1-z)^2} + \frac{2z^2}{(1-z)^3} 
\]

It turns out that for \(0 < \nu < 1\), \(K_\nu(z)\) is a complex function involving the exponential integral \(Ei(\nu(1-z))\), but for \(\nu = 0\) it has a simpler solution (it is not a particular case not covered by the general solution of \(K_\nu\) : \(K_0(z) = z \cdot I(z)\).

Since Eq. (7.2) is a linear first order differential equation in \(\theta I_\Omega(z)\), we can solve it by any standard method\(^3\), assuming \(\beta' = \gamma' = 0\), thus yielding,
\[
\theta I_\Omega(z) = 4 \ln(1-z) + \left( \mu - 5 \right) \frac{\ln(1-z)}{(1-z)^3} + \ldots
\]

And taking into account that \([z^n]I_\Omega(z) = \frac{1}{n+1}[z^n]\theta I_\Omega(z)\), we have
\[
\overline{IPL_{\Omega,n}} - \overline{IPL_n} = 2nH_n - 4n + 6H_n + (7/6\mu - 55/6) + 6H_n + O \left( \frac{1}{n} \right) = 2n \ln n + (2\gamma - 4)n + 6\ln n + (6\gamma + 7/6\mu - 49/6) + 6\ln n + O \left( \frac{1}{n} \right)
\]

It should be pointed out, that it does not depend of \(\alpha\) and that depends of \(\mu\), neither of \(\beta\) nor of \(\gamma\) alone. As suspected, average IPL continues being \(\Omega(n \ln n)\) and the difference between \(\overline{IPL_{\Omega,n}}\) and \(\overline{IPL_n}\) is only of logarithmic magnitude,
\[
\overline{IPL_{\Omega,n}} - \overline{IPL_n} = 4\ln n + (4\gamma + 7/6\mu - 55/6) + 6\ln n + O \left( \frac{1}{n} \right)
\]
\(^2\)The accent over this constant avoids confusion with the parameter of \(DEL_\Omega\).
\(^3\)A symbolic manipulation system, such as MAPLE, is almost indispensable to carry out the calculations.
In fact, the last two formulas are true only if $\mu \neq 1$; otherwise $IPL_n = IPL_{\Omega,n}$, since $\mu = 1$ and $\beta' = \gamma' = 0$ make $P_{\Omega}$ weak invariant, as discussed in Subsection 6.2.

8 Conclusions. Further Research

There are some points of interest worth mentioning,

- analysis of deletion algorithms has been done in a systematic manner, introducing a set of probabilistic parameters which guide the deletion strategy; these parameters allow the covering of many different deletion algorithms.

- complete description of the behavior is possible for arbitrary sizes; nevertheless, extensions of these results to arbitrary updating patterns should be done using an extended model with deals with shapes and key values.

- the tool which made possible this description, can surpass the limited area of application, and prove useful in other scenarios; in particular, the recursive characterization of induced probability models seems to be powerful enough to investigate the effects of deletion/insertion operations.

- the concept of weak invariance, which provides more flexible conditions under which updating could be well-behaved; for instance, no condition over $\alpha$ need to be imposed to achieve preservation of IPL after a random deletion.

- moreover, some conjectures could have an explanation in terms of weak invariance: for example, it's plausible that patterns of the type $(DI)^k$ ($k$ deletion/insertion pairs) with symmetric algorithms do not degrade expected IPL and search time because they induce a form of weak invariant probability.

Further research lines include

- the extension of the results using a model suitable for the analysis of more general updating patterns (the permutation model should be refined).

- the determination of equivalence relations and the conditions that should be imposed to achieve weak invariance for the main complexity measures.

- given an algorithm which induces weak invariance, the derivation useful properties that the induced probability after the application of the algorithm $k$ times verifies, if possible.

Acknowledgements

We wish to thank Prof. Rafael Casas for his early interest, helping to debug our first draft, and making useful comments and suggestions.

Proof of Lemma 5.1

$MAX_{\rho}$ and $MIN_{\rho}$, can be read as the probabilities to obtain the tree $T$ deleting from $t$ the greatest and smallest keys, respectively. Due to their symmetry (changing left by right and conversely everywhere) we shall deduce a recursive expression for any value $\rho$ in $[0, 1]$, but for the first function only. We shall split the summation as in Eq. (5.2), and apply the recurrent equations that $DELMAX_{\rho}$ satisfies:

\[
MAX_{\rho}(T) = \sum_{t \in B_{|T|+1}} DELMAX_{\rho}(t, T) P(t) =
\]

\[
= (b) \sum_{t, \rho \neq \emptyset} DELMAX_{\rho}(t_r, T_r) t_r(T_r) \frac{P(t_r) \cdot P(t_r)}{|T_r| + 1} +
\]

17
\[
+ (c.2) \sum_{t_i \neq \Box, t_i = t_\nu} \left[ \rho \, DELM A X_\rho(t_i, T_i) \, \square(T_i) + (1 - \rho) \, t_i(T) \right] \frac{P(t_i)}{|T| + 1} + \\
+ (c.4) \sum_{t_i = \Box} 1
\]

If \( T \neq \Box \), taking out of the sum some factors yields

\[
MAX_\rho(T) = \frac{1}{|T| + 1} \sum_{t_r \in B_{|T|+1}} DELM A X_\rho(t_r, T_r) \, P(T_r) \\
+ \frac{1}{|T| + 1} \left[ \rho \, \square(T_r) \sum_{t_i \in B_{|T|+1}} DELM A X_\rho(t_i, T_i) \, P(t_i) + (1 - \rho) \, P(T) \right]
\]

and some final elementary manipulations yield the desired result.

On the other hand, an analogous recurrence is obtained, by the above mentioned symmetry.

Proof of Theorem 5.1

We shall demonstrate each of the cases into which \( P_{11} \) has been split (see Eq. (5.2)).

(a) We have for this case

\[
\frac{1}{|T| + 1} \sum_{t_r \in B_{|T|+1}} \left( \sum_{1 \leq z < |t_i| + 1} DEL_\Omega(z, t_i, T) \right) P(t)
\]

which by Eq. (4.4) equals

\[
\frac{1}{|T| + 1} \sum_{t_r \in B_{|T|+1}} \left( \sum_{1 \leq z < |t_i| + 1} DEL_\Omega(z, t_i, T_i) t_r(T_r) \frac{P(t_i) \cdot P(t_r)}{|t|} \right)
\]

Pulling out the sums all constant factors,

\[
\frac{1}{(|T| + 1)^2} \sum_{t_r \in B_{|T|+1}} t_r(T_r) P(t_r) \sum_{1 \leq z < |t_i| + 1} DEL_\Omega(z, t_i, T_i) P(t_i)
\]

Realizing that \( t(T) \) is one if and only if \( T = t \), we can change the limits of the sum

\[
\sum_{t_r \in B_{|T|+1}} t_r(T_r) P(t_r) = P(T_r) \cdot \sum_{t_r = \Box} P(T_r) \cdot \sum_{t_r \in B_{|T|+1}} P(T_r) \cdot \sum_{t_r \in B_{|T|+1}}
\]

so we have by substituting Eq. (2.2) in Eq. (1.1)

\[
\frac{1}{(|T| + 1)^2} P(T_r) \sum_{t_i \in B_{|T|+1}} \left( \sum_{1 \leq z \leq |T_i| + 1} DEL_\Omega(z, t_i, T_i) \right) P(t_i)
\]
Remembering the definition of $P_{i1}(T)$ for the left son $T_i$, given by Eq (5.1)

$$
P_{i1}(T) = \frac{1}{|T| + 1} \sum_{t_i \in \mathcal{B}_{1}^{T_1+1}}^{T_1+1} \text{DEL}_{i1}(z_i, t_i) \cdot P(t_i)
$$

we have, finally, that Eq. (5.2) and the previous yield the stated result.

(b) This case is analogous to the first case, by changing left by right and vice-versa everywhere.

(c.1) This case considers only one sum, so we write it as

$$
\frac{1}{|T| + 1} \sum_{t_i \notin \mathcal{S}_{1}} \text{DEL}_{i1}(|t_i| + 1, t_i, T) \cdot P(t_i)
$$

Plugging Eq. (4.6) turns into

$$
\frac{1}{|T| + 1} \sum_{t_i \notin \mathcal{S}_{1}} \left[ (1 - \alpha) \text{DELMAX}_{i1}(t_i, T_i) \cdot t_i(T_i) + \alpha \text{DELMIN}_{i1}(t_i, T_i) \cdot t_i(T_i) \right] \cdot \frac{P(t_i) \cdot P(t_{i1})}{|t_i|}
$$

that equals

$$
\frac{1}{|T| + 1} \left[ \sum_{t_i \notin \mathcal{S}_{1}} (1 - \alpha) \text{DELMAX}_{i1}(t_i, T_i) \cdot t_i(T_i) \cdot \frac{P(t_i) \cdot P(t_{i1})}{|t_i|} + \sum_{t_i \notin \mathcal{S}_{1}} \alpha \text{DELMIN}_{i1}(t_i, T_i) \cdot t_i(T_i) \cdot \frac{P(t_i) \cdot P(t_{i1})}{|t_i|} \right]
$$

And extracting out of the sum some factors like in the preceding case results

$$
\frac{1}{(|T| + 1)^2} \left[ (1 - \mathbb{S}(T_i)) (1 - \alpha) \cdot P(T_i) \sum_{t_i \in \mathcal{B}_{1}^{T_1+1}} \text{DELMAX}_{i1}(t_i, T_i) \cdot P(t_i) + (1 - \mathbb{S}(T_i)) \cdot \alpha \cdot P(T_i) \sum_{t_i \in \mathcal{B}_{1}^{T_1+1}} \text{DELMIN}_{i1}(t_i, T_i) \cdot P(t_i) \right]
$$

we can conclude that the third pair of sums of Eq. (5.1) equals the given result.

(c.2) As in the above case, it is constituted by only one sum :

$$
\frac{1}{|T| + 1} \sum_{t_i \notin \mathcal{S}_{1}} \text{DEL}_{i1}(|t_i| + 1, t_i, T) \cdot P(t_i)
$$

Taking into account Eq. (4.7), yields

$$
\frac{1}{|T| + 1} \sum_{t_i \notin \mathcal{S}_{1}} \left[ \beta \cdot \text{DELMAX}_{i1}(t_i, T_i) \cdot \mathbb{S}(T_i) + (1 - \beta) \cdot t_i(T_i) \right] \cdot \frac{P(t_i) \cdot P(t_{i1})}{|T| + 1}
$$

Applying the same manipulations as in case (c.1) yields the expression into its definitive form.

19
(c.3) The sum to evaluated is
\[
\frac{1}{|T| + 1} \sum_{t=1}^{\text{max}} D_E L_0(1,t,T) P(t)
\]
that is (c.2)'s symmetric case, changing left by right, \( \text{MAX}_L \) by \( \text{MIN}_R \) and so on.

(c.4) This a trivial case since it is just
\[
\frac{1}{|T| + 1} \sum_{t=1}^{\text{max}} D_E L_0(t,T) = \sum_{t=1}^{\text{max}} D(t)
\]

References


<table>
<thead>
<tr>
<th>Deletion algorithm</th>
<th>$\Omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hibbard's [9]</td>
<td>(1,0,1,0,0)</td>
</tr>
<tr>
<td>Knuth's [12]</td>
<td>(1,0,0,0,0)</td>
</tr>
<tr>
<td>Symmetric [5]</td>
<td>($\frac{1}{2}$,1,1,0,0)</td>
</tr>
<tr>
<td>Symmetric [2]</td>
<td>($\frac{1}{2}$,0,0,0,0)</td>
</tr>
</tbody>
</table>

Table 1: Some deletion algorithms and its characteristic $\Omega$. 