Deletion Algorithms for Binary Search Trees*

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

April 12, 1994

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.

On the other hand, 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.

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 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 $i+2$ were 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

$$Prob(X=i)=\frac{1}{n} \tag{1}$$

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. (1) states. Let $P(T_1)$ be the probability of the tree T_1 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,

$$P(T)=\frac{P(T_1)\cdot P(T_2)}{n}=\frac{P(T_1)\cdot P(T_2)}{|T|}, \qquad n>0 \tag{2}$$

where T is the tree constructed by attaching a root node, T_1 as the left subtree and T_2 as the right subtree. We will write this as $T = {}_{T_1}\!\overset{\circ}{/\backslash}_{T_2}$.

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 $\mathcal{B}$ and the set of binary trees of size n by $\mathcal{B}_n$.

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$_\Omega$, 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 briefly. Once the key k is found, a random uniformly distributed number δ over the real interval $[0,1]$ is drawn by means of the procedure random_compute, and then it is tested which of the four possible cases schematically shown in Figure 2 are we facing.

The first is a trivial one : simply get rid of the key.

```
function DELETE_Ω(k: key_type; t: BST) returns BST is
  if
    k < key(t) then left_son(t) := DELETE_Ω(k, left_son(t))
    k > key(t) then right_son(t) := DELETE_Ω(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_MAX_β'(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_MIN_γ'(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_MAX_β'(left_son(t))
                                           δ < α then key(t) := smallest_key(right_son(t))
                                                      right_son(t) := DELETE_MIN_γ'(right_son(t))
                                         endif
                      endif
  endif
endfunction
```

Figure 1: Algorithm DELETE_Ω.

Figure 2: Deletion of a root 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 β, 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 β', whose purpose is the same as β's. The introduction of a distinct algorithm, using its own parameter to what to do, is due to our intention of characterizating some classical algorithms which in fact operate in this way.

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

Just now we have introduced four probabilistic parameters. The fourth case gives raise to the fifth parameter, denoted α. When we want to delete a node which has two sons, we must choose 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 α 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$_\Omega$. This has been accomplished by associating probability functions depending on x and t to DELETE$_\Omega$, so the application of the appropiate 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.

Figure 3: Deletion of a maximum key.

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,

$$f : \mathcal{B}_n \mapsto [0,1], \qquad \text{with} \sum_{T \in \mathcal{B}_n} f(T) = 1$$

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 $\mathcal{B}_n$, the support set of the function f, is

$$\sup(f) = \{T \in \mathcal{B}_n \mid f(T) > 0\}$$

Since $\mathcal{B}_n$ is finite, the support of any such function is also finite. Let $\sup(f) = \{T_1, \ldots, T_k\}$ and $f_i = f(T_i), \quad i = 1, \ldots, k$. By the previous convention, we can write f as

$$f = f_1 T_1 + f_2 T_2 + \ldots + f_k T_k$$

function DELETE_MAX$_{\beta'}$(k : key_type; t : BST) **returns** BST **is**
    while $right\_son(t) \neq \Box$ **do** $t := right\_son(t)$ **endwhile;**
    if
      $left\_son(t) = \Box$ **then** $t := \Box$
      $left\_son(t) \neq \Box$ **then** $random\_compute(\delta)$
                              if
                                $\delta \geq \beta'$ **then** $t := left\_son(t)$
                                $\delta < \beta'$ **then** $left\_son(t) :=$ DELETE_MAX$_{\beta'}(left_son(t))$
                              endif
    endif
endfunction

Figure 4: Algorithm DELETE_MAX$_{\beta'}$.

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

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 $\mathcal{B}_n$, denoted by $[af + bg]$ is the PF defined over $\mathcal{B}_n$ given by*

$$[af + bg](T) = a \cdot f(T) + b \cdot g(T) \tag{3}$$

with $0 \leq a, b \leq 1$ and $a + b = 1$.

This operation is meaningful only if both f and g are mappings over $\mathcal{B}_n$, 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 $\mathcal{B}_n$ and $\mathcal{B}_m$ respectively, their tree combination, denoted $[{}_f\overset{\circ}{\wedge}{}_g]$, is a function defined over $\mathcal{B}_{n+m+1}$, such that for trees $T \in \mathcal{B}_{n+m+1}$ with $|T_l| = n$ and $|T_r| = m$ has value*

$$\left[{}_f\overset{\circ}{\wedge}{}_g\right](T) = f(T_l) \cdot g(T_r) \tag{4}$$

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 [16], although we will use them in quite a different way.

Bearing these definitions in mind, the behavior of the deletion algorithm 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 $DEL_\Omega(x,t,T)$ we mean the probability that, given a tree $t \neq \Box$ and a key x, algorithm 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 x ranges between 1 and $|t|$. $DELMAX_{\beta'}(t,T)$ and $DELMIN_{\gamma'}(t,T)$ are defined in the same way.

Moreover, let $DEL_\Omega(x,t)$ be the probability function defined over the set of binary trees with one node less than t, such that,

$$DEL_\Omega(x,t) \;:\; \begin{array}{ccc} \mathcal{B}_{|t|-1} & \longrightarrow & [0,1] \\ T & \mapsto & DEL_\Omega(x,t,T) \end{array}$$

In other words,

$$DEL_\Omega(x,t,T) = [DEL_\Omega(x,t)](T)$$

For instance,

$$DEL_\Omega\Big(2, \qquad\qquad\qquad \Big) = \alpha \cdot \qquad\qquad + (1-\alpha) \cdot$$

It is obvious that $DEL_\Omega(x,t)$ is not defined for $t = \Box$, and that

$$DEL_\Omega\Big(1, \qquad \Big) = \Box \tag{5}$$

Similarly, $DELMIN_{\gamma'}(t)$ and $DELMAX_{\beta'}(t)$ are the PFs corresponding to deletion of the minimum ($x = 1$) and maximum ($x = |t|$) of a tree t, and Eq. (5) applies to them.

Now, we proceed with a case analysis of $DEL_\Omega(x,t)$, considering the subtrees of t, t_l and t_r, and the value of x (see Figure 6), which leads to the following Proposition :

Proposition 4.1. *The probability function $DEL_\Omega(x,t)$ verifies,*

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

$$DEL_\Omega(x, {}_{t_l}\!\overset{\circ}{\wedge}_{t_r}) = {}_{DEL_\Omega(x,t_l)}\!\overset{\circ}{\wedge}_{t_r} \tag{6}$$

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

$$DEL_\Omega(x, {}_{t_l}\!\overset{\circ}{\wedge}_{t_r}) = {}_{t_l}\!\overset{\circ}{\wedge}_{DEL_\Omega(x-|t_l|-1,t_r)} \tag{7}$$

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

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

$$DEL_\Omega(x, {}_{t_l}\!\overset{\circ}{\wedge}_{t_r}) = (1-\alpha) \cdot {}_{DELMAX_{\beta'}(t_l)}\!\overset{\circ}{\wedge}_{t_r} + \alpha \cdot {}_{t_l}\!\overset{\circ}{\wedge}_{DELMIN_{\gamma'}(t_r)} \tag{8}$$

(c.2) *t_l is not empty, but t_r is : $|t_l| > 0, |t_r| = 0$.*

$$DEL_\Omega(x, {}_{t_l}\!\overset{\circ}{\wedge}_{\Box}) = \beta \cdot {}_{DELMAX_{\beta'}(t_l)}\!\overset{\circ}{\wedge}_{\Box} + (1-\beta)t_l \tag{9}$$

(c.3) *t_r is not empty, but t_l is : $|t_l| = 0, |t_r| > 0$.*

$$DEL_\Omega(x, {}_{\square}\!\overset{\circ}{\wedge}\!{}_{t_r}) = \gamma \cdot {}_{\square}\!\overset{\circ}{\wedge}\!{}_{DELMIN_{\gamma'}(t_r)} + (1-\gamma)t_r \tag{10}$$

(c.4) *both t_l and t_r are empty : $|t_l| = |t_r| = 0$.*

$$DEL_\Omega(x, {}_{\square}\!\overset{\circ}{\wedge}\!{}_{\square}) = \square \tag{11}$$

Note that we shall substitute x by $x - |t_l| - 1$ in case **(b)** because of the implied re-labelling (see Section 2). On the other hand, case **(c.4)** is nothing else than Eq. (5), 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. (8) 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 succesor (the minimum key at its right subtree) leaving the same left subtree with probability α.

$DELMIN_{\gamma'}$ and $DELMAX_{\beta'}$ can be characterized in a similar way. In fact, case (a) applies to $DELMIN_{\gamma'}$ and (b) to $DELMAX_{\beta'}$. Only cases (c.2) and (c.4) could happen when we are actually deleting the maximum key, so Eqs. (9) and (11) work properly for $DELMAX_{\beta'}$, replacing β by β' and DEL_Ω by $DELMAX_{\beta'}$:

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

$$DELMAX_{\beta'}({}_{t_l}\!\overset{\circ}{\wedge}\!{}_{t_r}) = {}_{t_l}\!\overset{\circ}{\wedge}\!{}_{DELMAX_{\beta'}(t_r)} \tag{12}$$

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

(c.2) t_l is not empty : $|t_l| > 0$.

$$DELMAX_{\beta'}({}_{t_l}\!\overset{\circ}{\wedge}\!{}_{\square}) = \beta' \cdot {}_{DELMAX_{\beta'}(t_l)}\!\overset{\circ}{\wedge}\!{}_{\square} + (1-\beta')t_l \tag{13}$$

(c.4) t_l is empty : $|t_l| = 0$.

$$DELMAX_{\beta'}(x, {}_{\square}\!\overset{\circ}{\wedge}\!{}_{\square}) = \square \tag{14}$$

Analogously, Eqs. (6), (10) and (11) cover the case of actual deletion of the minimum, interchanging the rôles of β' and γ', 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 $DEL_\Omega(x, t, T)$, $DELMAX_{\beta'}(t, T)$ and $DELMIN_{\gamma'}(t, T)$. To exemplify how it can be done, we translate the recurrence associated to case **(c.1)** :

(c.1) x is the root of t where $|t_l| > 0$ and $|t_r| > 0$.

$$\begin{aligned} DEL_\Omega(x, t, T) \;&=\; \Big[(1-\alpha)\, DELMAX_{\beta'}(t_l, T_l)\, t_r(T_r) + \\ &\qquad + \alpha\, DELMIN_{\gamma'}(t_r, T_r)\, t_l(T_l)\Big] \end{aligned} \tag{15}$$

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

Figure 6: Decomposition into cases.

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 :

$$P_\Omega(T) = \sum_{\substack{t \in \mathcal{B}_{|T|+1} \\ 1 \leq x \leq |T|+1}} DEL_\Omega(x, t, T)\, P(t)\, Prob(x) \tag{16}$$

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

$$P_\Omega(T) = \frac{1}{|T|+1} \sum_{\substack{t \in \mathcal{B}_{|T|+1} \\ 1 \leq x \leq n+1}} DEL_\Omega(x, t, T)\, P(t)$$

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

$$P_\Omega(T) \;=\; \frac{1}{n+1} \sum_{t \in \mathcal{B}_{n+1}} \sum_{1 \leq x \leq n+1} DEL_\Omega(x, t, T)\, P(t) =$$

$$
\begin{aligned}
&= \frac{1}{|T|+1}\Bigg[\textbf{(a)} \sum_{\substack{t\in\mathcal{B}_{n+1}\\ t_l\neq\square}} \sum_{1\leq x<|t_l|+1} DEL_\Omega(x,t,T)\,P(t) + \\
&+ \textbf{(b)} \sum_{\substack{t\in\mathcal{B}_{n+1}\\ t_r\neq\square}} \sum_{|t_l|+1<x\leq n+1} DEL_\Omega(x,t,T)\,P(t) + \\
&+ \textbf{(c.1)} \sum_{\substack{t\in\mathcal{B}_{n+1}\\ t_l\neq\square\neq t_r}} DEL_\Omega(|t_l|+1,t,T)\,P(t) + \\
&+ \textbf{(c.2)} \sum_{\substack{t\in\mathcal{B}_{n+1}\\ t_l\neq\square= t_r}} DEL_\Omega(|t_l|+1,t,T)\,P(t) + \\
&+ \textbf{(c.3)} \sum_{\substack{t\in\mathcal{B}_{n+1}\\ t_l=\square\neq t_r}} DEL_\Omega(1,t,T)\,P(t) + \\
&+ \textbf{(c.4)} \sum_{\substack{t\in\mathcal{B}_{n+1}\\ t_l=\square= t_r}} DEL_\Omega(1,t,T)\,P(t)\Bigg] \qquad (17)
\end{aligned}
$$

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 $P_\Omega(T_l)$ and $P_\Omega(T_r)$, but cases **(c.1)**, **(c.2)** and **(c.3)** require the introduction of two new functions :

$$
\begin{aligned}
MAX_\rho(T) &= \sum_{t\in\mathcal{B}_{|t|+1}} DELMAX_\rho(t,T)\,P(t) \\
MIN_\rho(T) &= \sum_{t\in\mathcal{B}_{|t|+1}} DELMIN_\rho(t,T)\,P(t) \qquad (18)
\end{aligned}
$$

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

Lemma 5.1. *For any ρ, with $0 \leq \rho \leq 1$,*

$$MAX_\rho(T) = \frac{1}{|T|+1}\Big[P(T_l)\,MAX_\rho(T_r) + \rho\cdot\square(T_r)\cdot MAX_\rho(T_l) + (1-\rho)\,P(T)\Big] \qquad (19)$$

$$MIN_\rho(T) = \frac{1}{|T|+1}\Big[P(T_r)\,MIN_\rho(T_l) + \rho\cdot\square(T_l)\cdot MIN_\rho(T_r) + (1-\rho)\,P(T)\Big] \qquad (20)$$

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

This lemma is left without proof.

Moreover, the following corollary can be established without difficulty :

Corollary 5.1. $MIN_0(T) = MAX_0(T) = P(T)$

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

Then, for each case in the decomposition depicted in Eq. (17) a recursive expresion, in terms of P, P_Ω and, eventually of $MAX_{\beta'}$ and $MIN_{\gamma'}$, is found. The following theorem can thus be stated :

Theorem 5.1. *If $n+1$ keys are randomly inserted in an initially empty tree, and one of them, randomly chosen, is deleted using $DELETE_\Omega$, the probability of the tree shape with n nodes produced is given by*

$$
\begin{aligned}
P_\Omega(T) &= \frac{1}{(|T|+1)^2}\Big[P(T_r)\, P_\Omega(T_l)\, (|T_l|+1) + P(T_l)\, P_\Omega(T_r)\, (|T_r|+1) + \\
&+ (1-\alpha)\, \delta(T_r \neq \Box)\, P(T_r)\, MAX_{\beta'}(T_l) + \alpha\, \delta(T_l \neq \Box)\, P(T_l)\, MIN_{\gamma'}(T_r) + \\
&+ \beta\, \delta(T_r = \Box)\, MAX_{\beta'}(T_l) + (1-\beta)\, P(T) + \\
&+ \gamma\, \delta(T_l = \Box)\, MIN_{\gamma'}(T_r) + (1-\gamma)\, P(T)\Big], \qquad T \neq \Box
\end{aligned} \tag{21}
$$

and $P_\Omega(\Box) = 1$.

where we have introduced, for commodity, the generalized Kronecker delta, whose meaning is

$$
\delta(expr) = \begin{cases} 1 & \text{if } expr \text{ is true} \\ 0 & \text{if } expr \text{ is false or undefined} \end{cases}
$$

In particular, the expressions of the type $t(T)$ have been translated to the equivalent ones $\delta(t = T)$ or $\delta(T = t)$.

Theorem 5.1 is proved by just summing up the recurrences found for each case, as Eq. (17) indicates.

Equation (21) can also be written as

$$
\begin{aligned}
P_\Omega(T) &= \frac{1}{(|T|+1)^2}\Big[P(T_r)\, P_\Omega(T_l)\, (|T_l|+1) + P(T_l)\, P_\Omega(T_r)\, (|T_r|+1) + (2-\beta-\gamma)\, P(T) + \\
&+ MAX_{\beta'}(T_l)\Big((1-\alpha)\, \delta(T_r \neq \Box)\, P(T_r) + \beta\, \delta(T_r = \Box)\Big) + \\
&+ MIN_{\gamma'}(T_r)\Big(\alpha\, \delta(T_l \neq \Box)\, P(T_l) + \gamma\, \delta(T_l = \Box)\Big)\Big], \qquad T \neq \Box
\end{aligned} \tag{22}
$$

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 and P_n 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. $Prob(A(e) = s)$.

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

$$
P_{n,A}(s) = \sum_{e \in E_n} P'_n(e)\, Prob(A(e) = s), \qquad s \in S_n,\ n \geq 0 \tag{23}
$$

In fact, Eq. (16) corresponds to the above equation, for the deletion algorithm DEL_Ω; 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*

$$
P_{n,A}(s) = P_n(s), \qquad \text{for any } s \in S_n, \qquad \text{for any } n \geq 0
$$

or, shortly, if $P = P_A$.

Other important concept is that of weak invariance, introduced in [14]. Let $\equiv$ be any equivalence relation, $S_n/\equiv$ 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*

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

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.

$$\begin{aligned}
\overline{C_n} &= \sum_{s\in S_n} C(s)\,P_n(s) = \sum_{[s]\in S_n/\equiv}\ \sum_{s'\in[s]} C(s')\,P_n(s') = \\
&= \sum_{[s]\in S_n/\equiv} C(s) \sum_{s'\in[s]} P_n(s') = \sum_{[s]\in S_n/\equiv} C(s)\,P_n^{\equiv}([s]) = \\
&= \sum_{[s]\in S_n/\equiv} C(s)\,P_{n,A}^{\equiv}([s]) = \sum_{[s]\in S_n/\equiv} C(s) \sum_{s'\in[s]} P_{n,A}(s') = \\
&= \sum_{[s]\in S_n/\equiv}\ \sum_{s'\in[s]} C(s')\,P_{n,A}(s') = \sum_{s\in S_n} C(s)\,P_{n,A}(s) = \\
&= \overline{C_{n,A}}, \qquad n \geq 0
\end{aligned}$$

For instance, the IPL of a binary tree and its symmetric is the same, and selecting the appropriate values for Ω, we can obtain weak invariance with respect to symmetry for P_Ω. 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, Hibbard'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 $SHAPE$ (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 esence of Knott's paradox.

For the same reason, in our subsequent analysis, it shall be took 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 treatement of strong invariance[1] under deletions of various kinds (not only random) in a wide class of data structures can be found in [13].

6.1 Strong Invariance. Hibbard's Theorem Revisited.

In this section, we will consider which values of α, β, ... lead to an strong invariant induced probability function, i.e. $P_\Omega(T) = P(T)$ for any tree T.

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

[1] It is denominated "insensitivity" in the cited paper.

Now, we will find sufficient conditions for strong invariance by assuming that $P_\Omega = P$ 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 $P(T_l)P(T_r) = |T| \cdot P(T)$ (Eq. (2)), Eq. (22) becomes,

$$\begin{aligned} P_\Omega(T) = \frac{1}{(|T|+1)^2}\Big[&|T|\,(|T|+1)\,P(T) + P(T)\,(2-\beta-\gamma)+ \\ &+\delta(T_l \neq \Box)\,\alpha\,|T|\,P(T)+ \\ &+\delta(T_r \neq \Box)\,(1-\alpha)\,|T|\,P(T)+ \\ &+\delta(T_r \neq \Box)\,\beta\,P(T_l)+ \\ &+\delta(T_l \neq \Box)\,\gamma\,P(T_r)\Big] \end{aligned} \tag{24}$$

If the four possible cases of subtrees being empty or not are considered then it follows that Eq. (24) equals $P(T)$ if and only if

$$\begin{aligned} \beta + \gamma &= 1 \\ \alpha &= \gamma \end{aligned} \tag{25}$$

Since the induction basis is true, $P_\Omega(\Box) = P(\Box) = 1$, 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 DEL_Ω with Ω satisfying conditions (25) 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 $sym(T)$ is*

$$sym\left({}_{t_l}\!\!\overset{\circ}{\wedge}\!\!{}_{t_r}\right) = {}_{sym(t_r)}\!\!\overset{\circ}{\wedge}\!\!{}_{sym(t_l)}$$

with $sym(\Box) = \Box$.

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

Next, we define the symmetrical probabilistic operator

$$P^{SYM}(\cdot) = P(\cdot) + P(sym(\cdot)) \tag{26}$$

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}_Ω is $P_\Omega(\cdot) + P_\Omega(sym(\cdot))$.

Before we deduce the recurrent relations for P^{SYM}_Ω, we write some trivial identities about the symmetric relation :

(a) $sym(T)_l = sym(T_r)$ and $sym(T)_r = sym(T_l)$.

(b) $|sym(T)| = |T|$.

(c) $P(sym(T)) = P(T)$.

(d) $MAX_\rho(sym(T)) = MIN_\rho(T)$ and $MIN_\rho(sym(T)) = MAX_\rho(T)$, for any $0 \le \rho \le 1$.

Beginning with the Eqs. (16) and (26), and applying the last identities results

$$\begin{aligned}
P^{SYM}_\Omega(T) &= \frac{1}{(|T|+1)^2}\Big[P(T_r)\,P^{SYM}_\Omega(T_l)\,(|T_l|+1) + P(T_l)\,P^{SYM}_\Omega(T_r)\,(|T_r|+1)\\
&+ \delta(T_r \neq \Box)\; P(T_r)\; MAX_{\beta'}(T_l)\\
&+ \delta(T_l \neq \Box)\; P(T_l)\; MIN_{\gamma'}(T_r)\\
&+ \delta(T_r \neq \Box)\big[\beta\, MAX_{\beta'}(T_l) + \gamma\, MAX_{\beta'}(T_l)\big]\\
&+ \delta(T_l \neq \Box)\big[\beta\, MIN_{\gamma'}(T_r) + \gamma\, MIN_{\gamma'}(T_r)\big]\\
&+ 2(2-\beta-\gamma)\; P(T)\Big]
\end{aligned}$$

where the dependence of α has vanished.

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

$$\begin{aligned}
P^{SYM}_\Omega(T) &= \frac{1}{(|T|+1)^2}\Big[P(T_r)\,P^{SYM}_\Omega(T_l)\,(|T_l|+1) + P(T_l)\,P^{SYM}_\Omega(T_r)\,(|T_r|+1)\\
&+ \delta(T_r \neq \Box)\,P(T_r)\,MAX_{\beta'}(T_l) + +\delta(T_l \neq \Box)\,P(T_l)\,MIN_{\gamma'}(T_r)\\
&+ \mu\,\big[\delta(T_r = \Box)\,MAX_{\beta'}(T_l) + +\delta(T_l = \Box)\,MIN_{\gamma'}(T_r)\big]\\
&+ 2(2-\mu)\,P(T)\Big]
\end{aligned}$$

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

$$\begin{aligned}
\mu &= 1\\
\beta' &= \gamma' = 0
\end{aligned}$$

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 $\overline{IPL_n}$ 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,

$$\begin{aligned}
I(z) &= \sum_{n\ge 0} \overline{IPL_n} z^n = \sum_{T\in\mathcal{B}} IPL(T)\,P(T)\,z^{|T|} =\\
&= -2\frac{\ln(1-z)+z}{(1-z)^2}
\end{aligned}$$

and

$$\begin{aligned}
\overline{IPL_n} &= [z^n]I(z) = 2nH_n + 2H_n - 4n\\
&= 2n\ln n - (4-2\hat{\gamma})n + 2\ln n + (2\hat{\gamma}+1) + O\left(\frac{1}{n}\right)
\end{aligned}$$

where $\hat{\gamma} \approx 0.57721\ldots$ is Euler's constant [2] and H_n denotes the n^{th} harmonic number.

[2] The accent over this constant avoids confusion with the parameter of DEL_Ω.

If Ω is such that P_Ω 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_Ω in a random tree of size $n+1$. We define

$$I_\Omega(z) = \sum_{n\geq 0} \overline{IPL_{\Omega,n}} z^n = \sum_{T\in\mathcal{B}} IPL(T)\, P_\Omega(T)\, z^{|T|} \tag{27}$$

Eqs. (2),(21),(19), (20) and the definition of $IPL(T)$ lead to

$$\begin{aligned}
\vartheta^2 I_\Omega(z) &= \frac{2z}{1-z}\vartheta I_\Omega(z) + \frac{3-z}{(1-z)^2} I(z) + \\
&\quad +2(2-\mu)\int_0^z \frac{I(t)}{1-t}\,dt + \vartheta^2\left[\frac{z^2}{(1-z)^2}\right] + \\
&\quad + \left[\beta + (1-\alpha)\frac{z}{1-z}\right] K_{\beta'}(z) + \left[\gamma + \alpha\frac{z}{1-z}\right] K_{\gamma'}(z)
\end{aligned} \tag{28}$$

where $\vartheta F = \frac{d}{dz}(z \cdot F)$, $\mu = \beta + \gamma$ and $K_\nu(z)$ is the solution of

$$\begin{aligned}
\frac{dK_\nu}{dz} - \left(\frac{1}{1-z} + \nu\right) K_\nu(z) &= \left(\frac{1}{1-z} - \nu\right) I(z) + \nu\frac{z^2}{(1-z)^2} + \frac{2z^2}{(1-z)^3} \\
K_\nu(0) &= 0
\end{aligned}$$

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

Since Eq. (28) is a linear first order differential equation in $\vartheta I_\Omega(z)$, we can solve it by any standard method[3], assuming $\beta' = \gamma' = 0$, thus yielding,

$$\vartheta I_\Omega(z) = 4\frac{-\ln(1-z)}{z(1-z)^3} + (\mu - 5)\frac{-\ln(1-z)}{z(1-z)^2} + \dots$$

And taking into account that $[z^n] I_\Omega(z) = \frac{1}{n+1}[z^n]\vartheta\, I_\Omega(z)$, we have

$$\begin{aligned}
&\overline{IPL_{\Omega,n}} = \\
&\quad = 2nH_n - 4n + 6H_n + (7/6\mu - 55/6) + 6\frac{H_n}{n} + O\left(\frac{1}{n}\right) \\
&\quad = 2n\ln n + (2\hat{\gamma} - 4)n + 6\ln n + (6\hat{\gamma} + 7/6\mu - 49/6) + 6\frac{\ln n}{n} + O\left(\frac{1}{n}\right)
\end{aligned}$$

It should be pointed out, that it does not depend of α and that depends of μ, neither of β nor of γ 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,

$$\overline{IPL_{\Omega,n}} - \overline{IPL_n} = 4\ln n + (4\gamma + 7/6\mu - 55/6) + 6\frac{\ln n}{n} + O\left(\frac{1}{n}\right)$$

In fact, the last two formulas are true only if $\mu \neq 1$; otherwise $\overline{IPL_n} = \overline{IPL_{\Omega,n}}$, since $\mu = 1$ and $\beta' = \gamma' = 0$ make P_Ω weak invariant, as discussed in Subsection 6.2.

[3] A symbolic manipulation system, such as MAPLE, is almost indispensable to carry out the calculations.

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 α 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/insertions 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 refused).
- 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 Rafael Casas for his early interest, helping to debug our first draft, and making useful comments and suggestions.

Proof of Lemma 5.1

$MAX_{\beta'}$ and $MIN_{\gamma'}$ can be readed 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 ρ in $[0,1]$, but for the first function only. We shall split the summation as in Eq. (17), and apply the recurrent equations that $DELMAX_\rho$ satisfies :

$$\begin{aligned}
MAX_\rho(T) &= \sum_{t\in\mathcal{B}_{|T|+1}} DELMAX_\rho(t,T)\,P(t) = \\
&= \textbf{(b)} \sum_{t_r\neq\square} DELMAX_\rho(t_r,T_r)\,t_l(T_l)\,\frac{P(t_l)\cdot P(t_r)}{|T|+1} + \\
&+ \textbf{(c.2)} \sum_{t_l\neq\square=t_r} \Big[\rho\,DELMAX_\rho(t_l,T_l)\,\delta(T_r=\square)+(1-\rho)\,t_l(T)\Big]\frac{P(t_l)}{|T|+1} + \\
&+ \textbf{(c.4)} \sum_{t_l=\square=t_r} 1
\end{aligned}$$

If $T \neq \square$, taking out of the sum some factors yields

$$\begin{aligned} MAX_\rho(T) &= \frac{1}{|T|+1} P(T_l) \sum_{t_r \in \mathcal{B}_{|T_r|+1}} DELMAX_\rho(t_r, T_r)\, P(T_r) \\ &+ \frac{1}{|T|+1} \left[\rho \cdot \delta(T_r = \square) \sum_{t_l \in \mathcal{B}_{|T_l|+1}} DELMAX_\rho(t_l, T_l)\, P(t_l) + (1-\rho)\, P(T) \right] \end{aligned}$$

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_Ω has been splitted (see Eq. (17)).

(a) We have for this case

$$\frac{1}{|T|+1} \sum_{\substack{t \in \mathcal{B}_{n+1} \\ t_l \neq \square}} \sum_{1 \leq x < |t_l|+1} DEL_\Omega(x, t, T)\, P(t)$$

which by Eq. (6) equals

$$\frac{1}{|T|+1} \sum_{\substack{t \in \mathcal{B}_{n+1} \\ t_l \neq \square}} \sum_{1 \leq x < |t_l|+1} DEL_\Omega(x, t_l, T_l)\, t_r(T_r)\, \frac{P(t_l) \cdot P(t_r)}{|t|}$$

Pulling out the sums all constant factors,

$$\frac{1}{(|T|+1)^2} \sum_{\substack{t \in \mathcal{B}_{n+1} \\ t_l \neq \square}} t_r(T_r)\, P(t_r) \sum_{1 \leq x < |t_l|+1} DEL_\Omega(x, t_l, T_l)\, P(t_l) \tag{29}$$

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

$$\begin{aligned} \sum_{\substack{t \in \mathcal{B}_{n+1} \\ t_l \neq \square}} t_r(T_r)\, P(t_r) \sum_{1 \leq x \leq |t_l|+1} \cdots &= P(T_r) \cdot \sum_{\substack{t_l \neq \square \\ t_r = T_r}} \sum_{1 \leq x \leq |t_l|+1} \cdots = \\ &= P(T_r) \cdot \sum_{t_l \in \mathcal{B}_{|t|-1-|t_r|}} \sum_{1 \leq x \leq |t_l|+1} \cdots = \\ &= P(T_r) \cdot \sum_{t_l \in \mathcal{B}_{|T_l|+1}} \sum_{1 \leq x \leq |t_l|+1} \cdots \end{aligned} \tag{30}$$

so we have by substituting Eq. (30) in Eq. (29)

$$\frac{1}{(|T|+1)^2} P(T_r) \sum_{t \in \mathcal{B}_{|T_l|+1}} \sum_{1 \leq x \leq |T_l|+1} DEL_\Omega(x, t_l, T_l)\, P(t_l)$$

Remembering the definition of $P_\Omega(T)$ for the left son T_l, given by Eq (16)

$$P_\Omega(T_l) = \frac{1}{|T_l|+1} \sum_{\substack{t \in \mathcal{B}_{|T_l|+1} \\ 1 \leq x \leq |T_l|+1}} DEL_\Omega(x, t, T_l)\, P(t_l) \tag{31}$$

we have, finally, that Eq. (30) and the previous yield

$$\frac{1}{(|T|+1)^2} P(T_r)\, P_\Omega(T_l)\, (|T_l|+1) \tag{32}$$

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

$$\frac{1}{(|T|+1)^2} P(T_l)\, P_\Omega(T_r)\, (|T_r|+1) \tag{33}$$

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

$$\frac{1}{|T|+1} \sum_{t_l \neq \Box \neq t_r} DEL_\Omega(|t_l|+1, t, T)\, P(t)$$

Plugging Eq. (8) turns into

$$\begin{aligned} \frac{1}{|T|+1} \sum_{t_l \neq \Box \neq t_r} \Big[& (1-\alpha)\, DELMAX_{\beta'}(t_l, T_l)\, t_r(T_r) + \\ + \quad & \alpha\, DELMIN_{\gamma'}(t_r, T_r)\, t_l(T_l) \Big] \frac{P(t_l) \cdot P(t_r)}{|t|} \end{aligned}$$

that equals

$$\begin{aligned} \frac{1}{|T|+1} \Big[& \sum_{t_l \neq \Box \neq t_r} (1-\alpha)\, DELMAX_{\beta'}(t_l, T_l)\, t_r(T_r) \frac{P(t_l) \cdot P(t_r)}{|t|} \\ + \quad & \sum_{t_l \neq \Box \neq t_r} \alpha\, DELMIN_{\gamma'}(t_r, T_r)\, t_l(T_l) \frac{P(t_l) \cdot P(t_r)}{|t|} \Big] \end{aligned}$$

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

$$\begin{aligned} \frac{1}{(|T|+1)^2} \Big[& \delta(T_r \neq \Box)\, (1-\alpha)\, P(T_r) \sum_{t_l \in \mathcal{B}_{|T_l|+1}} DELMAX_{\beta'}(t_l, T_l)\, P(t_l) \\ + \quad & \delta(T_l \neq \Box)\, \alpha\, P(T_l) \sum_{t_r \in \mathcal{B}_{|T_r|+1}} DELMIN_{\gamma'}(t_r, T_r)\, P(t_r) \Big] \end{aligned}$$

we can conclude that the third pair of sums of Eq. (16) equals

$$\frac{1}{(|T|+1)^2} \Big[(1-\alpha)\, \delta(T_r \neq \Box)\, P(T_r)\, MAX_{\beta'}(T_l) + \alpha\, \delta(T_l \neq \Box)\, P(T_l)\, MIN_{\gamma'}(T_r) \Big] \tag{34}$$

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

$$\frac{1}{|T|+1} \sum_{t_l \neq \Box = t_r} DEL_\Omega(|T_l|+1, t, T)\, P(t)$$

Taking into account Eq. (9), yields

$$\frac{1}{|T|+1} \sum_{t_l \neq \Box = t_r} \Big[\beta\, DELMAX_{\beta'}(t_l, T_l)\, \delta(T_r = \Box) + (1-\beta)\, t_l(T_l) \Big] \frac{P(t_l) \cdot P(t_r)}{|T|+1}$$

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

$$\frac{1}{(|T|+1)^2}\left[\beta\,\delta(T_r=\square)\,MAX_{\beta'}(T_l)+(1-\beta)\,P(T)\right] \tag{35}$$

(c.3) The sum to evaluated is

$$\frac{1}{|T|+1}\sum_{t_l=\square\neq t_r} DEL_\Omega(1,t,T)\,P(t)$$

that is (c.2)'s symmetric case, changing left by right, $MAX_{\beta'}$ by $MIN_{\gamma'}$ and so on. Hence, the final result is

$$\frac{1}{(|T|+1)^2}\left[\gamma\,\delta(T_l=\square)\,MIN_{\gamma'}(T_r)+(1-\gamma)\,P(T)\right] \tag{36}$$

(c.4) This a trivial case since it is just

$$\frac{1}{|T|+1}\,DEL_\Omega(1,{}_\square\!\wedge_\square,T)=\delta(T=\square) \tag{37}$$

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1976.

2. R. Baeza-Yates. A trivial algorithm whose analysis isn't: A continuation. Technical Report CS-86-67, Dept. of Computer Science,University of Waterloo, December 1986.

3. J. Culberson. Updating binary trees. Master's thesis, Dept. of Computer Science, University of Waterloo, March 1984.

4. J. Culberson. The effect of updates in binary search trees. In *ACM Symposium on the Theory of Computation*, pages 205–212, 1985.

5. J.L. Eppinger. An empirical study of insertion and deletion in binary search trees. *C.ACM*, 26(9):663–669, September 1983.

6. Ph. Flajolet. Mathematical methods in the analysis of algorithms and data structures. In Egon Börger, editor, *Trends in Theoretical Computer Science*, chapter 6, pages 225–304. Computer Science Press, 1988.

7. Ph. Flajolet. Random tree models in the analysis of algorithms. In *Performance'87*, pages 171–187, 1988.

8. G.H. Gonnet. *Handbook of Algorithms and Data Structures.* Addison-Wesley, London, 1984.

9. T.N. Hibbard. Some combinatorial properties of certain trees with applications to searching and sorting. *J. ACM*, 9(1):13–18, January 1962.

10. A.T. Jonassen and D.E. Knuth. A trivial algorithm whose analysis isn't. *JCSS*, 16(3):301–322, June 1978.

11. G.D. Knott. *Deletions in Binary Storage Trees.* PhD thesis, Computer Science Dept., Stanford University, May 1975.

12. D.E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, 1973.

13. D.E. Knuth. Deletions that preserve randomness. *IEEE Trans. Software Engineering*, 3:351–359, 1977.

14. X. Messeguer. Probabilistic deletion algorithms for BSTs. Technical Report 22, LSI-UPC, 1990.

15. R. Sedgewick. Mathematical analysis of combinatorial algorithms. In *Probability Theory and Computer Science*, pages 124–205. Academic Press, 1983.

16. J.-M. Steyaert. *Structure et Complexité des Algorithmes*. PhD thesis, Université Paris 7, April 1984.

Deletion algorithm	Ω
Hibbard's [9]	(1,0,1,0,0)
Knuth's [12]	(1,0,0,0,0)
Symmetric [5]	($\frac{1}{2}$,1,1,0,0)
Symmetric [2]	($\frac{1}{2}$,0,0,0,0)

Table 1: Some deletion algorithms and its characteristic Ω.