Random Records and Cuttings in Split Trees: Extended Abstract

Cecilia Holmgren[1]

[1] *DEPARTMENT OF MATHEMATICS, UPPSALA UNIVERSITY, PO Box 480, SE- 751 06 UPPSALA, SWEDEN*
`cecilia.holmgren@math.uu.se`

We study the number of records in random split trees on n randomly labelled vertices. Equivalently the number of random cuttings required to eliminate an arbitrary random split tree can be studied. After normalization the distributions are shown to be asymptotically 1-stable. This work is a generalization of our earlier results for the random binary search tree which is one specific case of split trees. Other important examples of split trees include m-ary search trees, quadtrees, median of $(2k+1)$-trees, simplex trees, tries and digital search trees.

Keywords: Random Trees

1 Introduction

1.1 Preliminaries

We study the number of records in random split trees which were introduced by Devroye [2]. This number is equivalent (in distribution) to the number of cuts to eliminate this type of tree as shown by Janson [9].

Given a rooted tree T with n nodes, let each vertex v have a random value λ_v attached to it, and assume that these values are i.i.d. with a continuous distribution. We say that the value λ_v is a *record* if it is the smallest value in the path from the root to v. Let $X_v(T)$ denote the (random) number of records. Alternatively one may attach random variables to the edges and let $X_e(T)$ denote the number of edges with record values. Because only the order relations of the λ_v's are important, the distribution of λ_v does not matter, i.e. one can choose any continuous distribution for λ_v.

The same random variables appear when we consider cuttings of the tree T as introduced by Meir and Moon [14] with the following definition. Make a random cut by choosing one vertex [or edge] at random. Delete this vertex [or edge] so that the tree separates into two parts and keep only the part containing the root. Continue recursively until the root is cut [only the root is left for the edge version]. Then the total (random) number of cuts made is $X_v(T)$ [or $X_e(T)$]. More precisely, cuttings and records give random variables with the same distribution. The proof of this equivalence uses a natural coupling argument as shown in [9].

In [9] the asymptotic distributions for the number of cuts (or the number of records) are found for random trees that can be constructed as conditioned Galton–Watson trees, e.g. labelled trees and random binary trees. There the proof relies on the fact that the method of moments could be used.

For the deterministic (not random) complete binary tree it is however not possible to use the method of moments for this purpose. Therefore Janson [8] introduced another strategy, which is to approximate $X_v(T)$ by a sum of independent random variables derived from λ_v, and then apply a classical limit theorem for triangular arrays. We recently showed that Janson's approach could also be applied for the random binary search tree [6] .

In this paper we consider all types of random split trees defined by Devroye [2], where the binary search tree that we consider in [6] is one example of such trees. Some other important examples of split trees are m-ary search trees, quadtrees, median of $(2k + 1)$-trees, simplex trees, tries and digital search trees. The split trees belong to the family of the so-called $\log n$ trees, that are trees with height (maximal depth) $\mathcal{O}(\log n)$. These have similar properties to the deterministic complete binary tree with height $\lfloor \log_2 n \rfloor$ considered in [8]. In the complete binary tree most vertices are close to $\lfloor \log_2 n \rfloor$ (the height of the tree). In split trees on the other hand most vertices are close to the depth $\sim c \ln n$, where c is a constant (it is natural to use the e-logarithm); for the binary search tree that we investigated in [6] this depth is $\sim 2 \ln n$ (e.g. [3]).

The (random) split trees is a large class of random trees which are recursively generated. Their formal definition is given in the "split tree generating algorithm" below. To facilitate the penetration of this rather complex algorithm we will first provide a brief heuristic description.

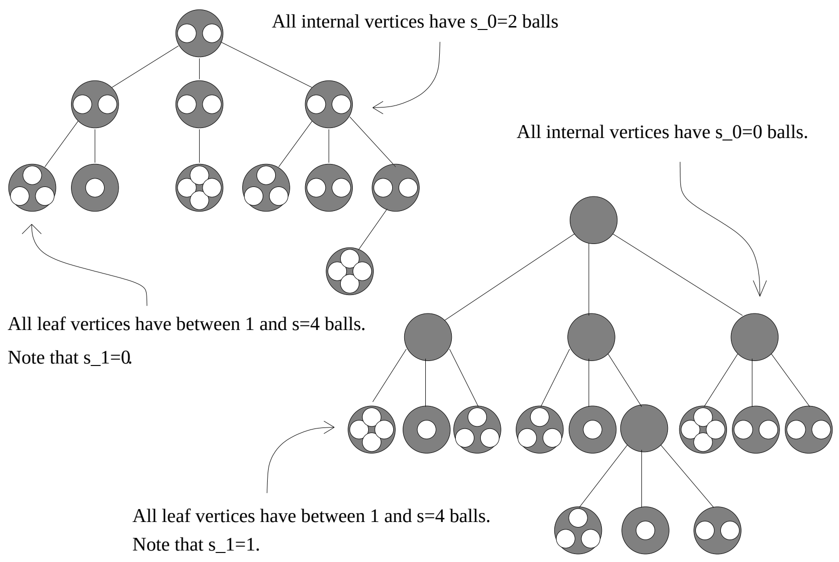
All internal vertices have s_0=2 balls
All internal vertices have s_0=0 balls.
All leaf vertices have between 1 and s=4 balls.
Note that s_1=0.
All leaf vertices have between 1 and s=4 balls.
Note that s_1=1.

Figure 1: *This figure illustrates two split trees. The left one has parameters $b = 3,\ s = 4,\ s_0 = 2$ and $s_1 = 0$, whereas the right one has parameters $b = 3,\ s = 4,\ s_0 = 0$ and $s_1 = 1$.*

A skeleton tree T_b of branch factor b is an infinite rooted tree in which each vertex has exactly b children that are numbered $1, 2, \ldots, b$. A split tree is a finite subtree of a skeleton tree T_b. The split tree is constructed recursively by distributing balls one at a time to a subset of vertices of T_b. We say that the tree has cardinality N if N balls are distributed. There is also a so-called vertex capacity, $s > 0$, which means that each node can hold at most s balls. We say that a vertex v is a leaf in a split tree if the node itself holds at least one ball but no descendants of v hold any balls. The split tree consists of the leaves and all the ancestors to the leaves, in particular the root of T_b, but no descendant to a leaf is included. In this way the definition of leaves in split trees is equivalent to the usual definition of leaves in trees. See figure 1 (taken from [2]), where two examples of split trees are illustrated (the parameters s_0 and s_1 in the figure are introduced in the formal "split tree generating algorithm").

The first ball is placed in the root of T_b. A new ball is added to the tree by starting in the root, and then let the ball fall down to lower levels in the tree until it reaches a leaf. Each vertex v of T_b is given an independent copy of the so-called random splitting vector $\mathcal{V} = (V_1, V_2 \dots, V_b)$ of probabilities, where $\sum_i V_i = 1,\ \ V_i \geq 0$. The splitting vectors control the path that the ball takes until it finally reaches a leaf; when the ball falls down one level from vertex v to one of its children, it chooses the i'th child of v with the probability V_i i.e. the i'th component of the splitting vector associated to v. When a full leaf (i.e. a leaf which already holds s balls) is reached by a new ball it splits. This means that some of the $s+1$ balls are given to its children, leading to new leaves so that more nodes will be included in the tree. When all the N balls have been distributed we end up with a split tree with a finite number of nodes which we denote by the parameter n.

The split tree generating algorithm: The formal, comprehensive "split tree generating algorithm" is as follows with the following introductory notations. The (random) split tree has the parameters b, N, s and $\mathcal{V}$ as we described above, there are also two other parameters: s_0, s_1 (related to the parameter s) that occur in the algorithm below. Let N_v denote the total number of balls that the vertices in the subtree rooted at vertex v hold together, and C_v be the number of balls that is hold by v itself. Note that an equivalent definition of the leaves as the one we give above is to say that a vertex v is a leaf if and only if $C_v = N_v > 0$. Also note that a vertex $v \in T_b$ is included in the split tree if, and only if, $N_v > 0$. If $N_v = 0$, the vertex v is not included and it is called useless.

Below there is a description of the algorithm how the N balls are distributed over the vertices. Initially there are no balls, i.e. $C_v = 0$ for each vertex v. Choose an independent copy $\mathcal{V}_v$ of $\mathcal{V}$ for every vertex $v \in T_b$. Add balls one by one to the root by the following recursive procedure for adding a ball to the subtree rooted at v.

(i) If v is not a leaf, choose child i with probability V_i and recursively add the ball to the subtree rooted at child i, by the rules given in steps (i), (ii) and (iii).

(ii) If v is a leaf and $C_v = N_v < s$, (s is the capacity of the vertex) then add the ball to v and stop. Thus, C_v and N_v increase by 1.

(iii) If v is a leaf and $C_v = N_v = s$, the ball cannot be placed at v since it is occupied by the maximal number of balls it can hold. Then, let $N_v = s+1$ and $C_v = s_0$, by placing $s_0 \leq s$ randomly chosen balls at v and $s+1-s_0$ balls to its children. This is done by first giving s_1 randomly chosen balls to each of the b children . The remaining $s+1-s_0-bs_1$ balls are placed by choosing a child for each ball independently according to the probability vector $\mathcal{V}_v = (V_1, V_2 \dots, V_b)$, and then using the algorithm described in steps (i), (ii) and (iii) to the subtree rooted at the selected child. Note that if $s_0 > 0$ this procedure does not need to be repeated since no child could reach the capacity s, whereas in the case $s_0 = 0$ this procedure may have to be repeated several times.

From (iii) it follows that the integers s_0 and s_1 have to satisfy the inequality

$$0 \leq s_0 \leq s,\ 0 \leq bs_1 \leq s+1-s_0.$$

Note that every nonleaf vertex has $C_v = s_0$ balls and every leaf has $0 < C_v \leq s$ balls. See figure 1 which shows two split trees, one with cardinality 27 and parameters $(b, s, s_0, s_1) = (3, 4, 2, 0)$ and the other with cardinality 26 and parameters $(b, s, s_0, s_1) = (3, 4, 0, 1)$.

We can assume that the components V_i of the splitting vector $\mathcal{V} = (V_1, V_2 \ldots, V_b)$ are identically distributed. (If this would not be the case they can anyway be made identically distributed by using a random permutation.) This gives (because $\sum_i V_i = 1$) that $\mathbf{E}(V_i) = \frac{1}{b}$. We use the notation T_N to denote that we have a split tree with N balls. The only parameters that are important in this work (and in general these parameters are the important ones for most results concerning split trees) are the cardinality N, the branch factor b and the splitting vector $\mathcal{V}$; this is illustrated in Subsubsection 1.2.1. In a binary search tree $b = 2$, the splitting vector $\mathcal{V} = (V_1, V_2)$ is distributed as $(U, 1 - U)$ where U is a uniform $U(0, 1)$ random variable which is equal to a beta $(1, 1)$ random variable. In fact for many important split trees the components in the splitting vector $\mathcal{V}$ are beta-distributed. (The other parameters for the binary search tree considered as a split tree are $s = 1,\ s_0 = 1$ and $s_1 = 0$.) For the binary search tree the number of balls N is the same number as the number of vertices n; this is not true for split trees in general.

1.2 Specific Background

1.2.1 Subtrees

For a split tree where the number of balls $N > s$, there are s_0 balls in the root and the cardinalities of the b subtrees are distributed as $(s_1, \ldots, s_1)$ plus a multinomial vector $(N - s_0 - bs_1, V_1, \ldots, V_b)$. Thus, conditioning on the splitting vector $\mathcal{V}_\sigma = (V_1, \ldots, V_b)$ that belongs to the root σ, the subtrees rooted at the children of the root have cardinalities close to $NV_1, \ldots, NV_b$. This is often used in applications of random binary search trees. In particular we used this frequently in [6]. Recall that the total number of balls in the subtree rooted at v is denoted by N_v (which is a random variable). Conditioning on all the splitting vectors of a tree with N balls gives that the subtree size N_v for v at distance k to the root is close to

$$NW_1 W_2 \ldots W_k, \tag{1}$$

where $W_r,\ r \in \{1, \ldots, k\}$ are i.i.d. random variables given by the splitting vectors associated with the nodes in the unique path from vertex v to the root (this means in particular that $W_r \overset{d}{=} V_i$). We note that (1) implies that the N_v's are not independent for different vertices. This follows since the paths to the root for two different vertices consist of some common vertices (at least always the root is common for all of those paths). Thus, it also follows that N_v for vertices that are close to each other are more dependent than for vertices whose last common ancestor is far away.

1.2.2 A strong law and a central limit law for the depth

In [2] Devroye presents a strong law and a central limit law for the depth D_N of the last inserted ball in a split tree with N balls and splitting vector $\mathcal{V}$. (Most vertices in the tree are also close to this depth.)

Let D_N be the depth of the last inserted ball in a random split tree with N balls and splitting vector $\mathcal{V}$. Let

$$\begin{aligned} \mu &:= b\mathbf{E}\Big(V \ln \frac{1}{V}\Big), \\ \sigma^2 &:= b\mathbf{E}\Big(V \ln^2 V\Big) - \mu^2. \end{aligned} \tag{2}$$

If $\mu \neq 0$ and $\mathbf{P}\{V = 1\} = 0$, then

$$\frac{D_N}{\ln N} \xrightarrow{p} \mu^{-1},$$

where $\xrightarrow{p}$ denotes convergence in probability, and

$$\frac{\mathbf{E}(D_N)}{\ln N} \to \mu^{-1}. \tag{3}$$

From (3) it is easy to deduce that also for the average depth $D_N^{'}$, i.e. the sum of all depths of the N balls divided by N, we have

$$\frac{\mathbf{E}(D_N^{'})}{\ln N} \to \mu^{-1}. \tag{4}$$

Furthermore, if $\sigma > 0$, then

$$\frac{D_N - (\ln N)/\mu}{\sqrt{\sigma^2 (\ln N)/\mu^3}} \xrightarrow{d} N(0,1), \tag{5}$$

where $N(0,1)$ denotes the standard Normal distribution and $\xrightarrow{d}$ denotes convergence in distribution. See [2, Theorem 1].

1.3 The Main Theorem

Assumption 1 For technical reasons we assume that there is an $\epsilon > 0$ and a constant α that depends on the type of split tree such that

$$\mathbf{E}(n) = \alpha N + \mathcal{O}\Big(N^{1-\epsilon}\Big), \tag{6}$$

and

$$\mathbf{Var}(n) = \mathcal{O}\Big(N^{2-2\epsilon}\Big);$$

recall that N is the number of balls and n is the number of nodes.

Assumption 1 has previously been shown to hold for example m-ary search trees [13].

Assumption 2 The total path length of a tree T is the sum of all depths of the vertices in T (distances to the root). Since the split tree T_N is a random tree the total path length is a random variable which we denote $\Upsilon(T_N)$. In analogy with [15] we assume that the first moment of the total path length is of the form

$$\mathbf{E}(\Upsilon(T_N)) = \mu^{-1}\alpha N \ln N + \zeta \alpha N + o(N), \tag{7}$$

where α is the constant that occurs in Assumption 1 above and ζ is also a constant that depends on the split tree.

Note that the first asymptotic term in Assumption 2 follows immediately from (4) and (6) above. It is not obvious that the second asymptotic term is of the form $\zeta\alpha N$. Examples of split trees that have been proved to have an expansion of the total path length as in (7) are binary search trees (e.g. [10]), random m-ary search trees [12], quad trees [15] and the random median of a $(2k+1)$-tree [16]. However, the assumption in (7) is not necessary for our aim to prove that the distribution of $X_v(T_N)$ [or $X_e(T_N)$] after normalization is asymptotically 1-stable. If the second asymptotic term in (7) is not of the form $\zeta\alpha N$, the normalization of $X_v(T_N)$ [or $X_e(T_N)$] is slightly different from the one in (8).

Theorem 1.1 *Suppose that $N \to \infty$ and Assumptions 1 and 2 hold. Then*

$$\left(X_v(T_N) - \frac{\alpha N}{\mu^{-1}\ln N} - \frac{\alpha N \ln\ln N}{\mu^{-1}\ln^2 N}\right) \Big/ \frac{\alpha N}{\mu^{-2}\ln^2 N} \xrightarrow{d} -W, \tag{8}$$

where W has an infinitely divisible distribution with characteristic function

$$\mathbf{E}\Big(e^{itW}\Big) = \exp\Big(- \frac{\mu^{-1}}{2}\pi|t| + it(C) - i|t|\mu^{-1}\ln|t|\Big), \tag{9}$$

where μ is the constant in (2) and α is the constant in Assumption 1. The same result holds for $X_e(T_N)$.

Remark 1.1 In the proof of (8) we get

$$\mathbf{E}\Big(e^{itW}\Big) = \exp\left(it\Big(C + \mu^{-1}(\gamma - 1)\Big) + \int_0^\infty (e^{itx} - 1 - itx\mathbf{1}[x<1])d\nu(x)\right), \tag{10}$$

where C is the constant in (9), γ is the Euler constant and the Lévy measure ν is supported on $(0,\infty)$ and has density

$$\frac{d\nu}{dx} = \frac{\mu^{-1}}{x^2}.$$

Recall that the Lévy measure ν gives that W is a weakly 1-stable distribution, see e.g. [5, Section XVII.3]. (The constant C can be expressed as

$$C = -\mu^{-1}\ln\mu^{-1} + 2\mu^{-1} - \mu^{-2}\sigma^2 - \mu^{-1}\gamma - \frac{\sigma^2 - \mu^2}{2\mu^2} + \zeta,$$

where μ and σ^2 are the constants in (2) and ζ is the constant in Assumption 2.) We can simplify the expression in (10) to get (9) above.

Remark 1.2 We note in analogy with [8] and [6] that most records occur close to the depth where most vertices are, i.e. $\sim \mu^{-1}\ln N$ for split trees. Also in analogy with [8] and [6], from Lemma 2.4 and the proof of Theorem 2.1 it follows that most of the random fluctuations of $X_v(T_N)$ can be explained by the values at depths close to $\ln\ln N$.

Remark 1.3 Let $h(v)$ be the height of v (also called the depth of v) i.e. distance to the root. For deterministic rooted trees (see [8]) $\mathbf{E}(X_v(T))=\sum_v \frac{1}{h(v)+1}$ and $\mathbf{E}(X_e(T))=\sum_{v\neq\sigma}\frac{1}{h(v)}$, where σ is the root. For random trees $\mathbf{E}(X_v(T)) = \mathbf{E}\Big(\sum_v \frac{1}{h(v)+1}\Big)$ and $\mathbf{E}(X_e(T))=\mathbf{E}\Big(\sum_{v\neq\sigma}\frac{1}{h(v)}\Big)$. Thus, as we noted for the specific case of the binary search tree [6, Remark 1.3] also for all other split trees

$$\mathbf{E}(X_e(T_N)) - \mathbf{E}(X_v(T_N)) = \mathbf{E}\Big(\sum_{v\neq\sigma}\frac{1}{h(v)(h(v)+1)}\Big) - 1 \sim C_1\frac{N}{\log^2 N},$$

for some constant $C_1 > 0$, while there is no similar difference in the limit distribution; see Theorem 1.1 above. As we noted in [6] (for the binary search tree), this behaviour suggests that it is impossible to use the method of moments to find the asymptotic distribution of $X_v(T_N)$ [or $X_e(T_N)$] for split trees as one could do for the Galton–Watson trees. Instead we generalize the proofs in [6] by using similar methods that Janson used for the complete binary tree [8].

Remark 1.4 Most likely the method that is used here should also work for other trees of logarithmic height and thus the limiting distribution for these trees should also be infinitely divisible and probably also weakly stable. This turns out to be the case for the random recursive tree (that is a logarithmic tree) where the limiting distribution of $X_e(T)$ was recently found to be weakly stable, see [4, Theorem 1.1] and [7, Theorem 1.1]. However, the methods used for the recursive tree in [4, 7] differ completely from our methods. The advantage of studying split trees compared to the whole class of log-n trees is that there is a common definition that describes all split trees and this is the reason why we only consider these trees in the present study.

2 The Method of Proof of the Main Theorem

The structure of the proof of the Main Theorem follows from the lemmas and Theorem 2.1 which are presented below. However, in this extended abstract we have excluded the details of the proofs (which are quite technical).

2.1 Notations

Most of our notations are similar to the ones that are used in [6], where the binary search tree is considered.

We use the standard notations $\log_b$ for the b-logarithm (recall that a split tree with parameter b is a b-ary tree) and $\ln$ for the e-logarithm. In the proof of Theorem 1.1 we treat the case $X_v(T_N)$ in detail and then indicate why the same result also holds for $X_e(T_N)$. From now on, since it is clear that we consider the vertex model we just write $X(T_N)$.

First recall from Subsection 1.1 that the λ_v's are i.i.d. random values associated to the vertices in T_N and that λ_v is a record if it is the smallest value in the path from v to the root.

Let $X(T_N)_y$ be $X(T_N) - 1$ conditioned on the root label $\lambda_\sigma = y$. Recall that we denote the total path length (the sum of all depths or equivalently the sum of the distances to the root) for all nodes in T_N by $\Upsilon(T_N)$.

We say that, $Y_n = o_p(a_n)$ if a_n is a positive number and Y_n is a random variable such that $Y_n/a_n \xrightarrow{p} 0$ as $n \to \infty$. We say that, $Y_n = \mathcal{O}_{L^p}(a_n)$ if a_n is a positive number and Y_n is a random variable such that $(\mathbf{E}(Y_n{}^p))^{\frac{1}{p}} \leq K a_n$ for some constant K.

In the sequel we write T instead of T_N. For a vertex $v \in T$, we let T_v be the subtree of T rooted at v. Recall that N_v is the number of balls in T_v. We denote the number of nodes in T_v by n_v. We can without loss of generality assume that the labels λ_v have an exponential distribution Exp(1). As mentioned in Subsection 1.1 this does not affect the distribution of $X(T_N)$.

Recall from Remark 1.3 that $h(v)$ is the height of v (also called the depth of v) i.e. distance to the root. Let $L := \lfloor \beta \log_b \ln N \rfloor$ for some constant β. (Below we choose $\beta > \frac{1}{-\log_b(\mathbf{E}V_i^2)-1}$.) Let Λ_{v_i} be the minimum of λ_v along the path $P(v_i) = \sigma, \ldots, v_i$ from the root σ of T to v_i, $1 \leq i \leq b^L$, where v_i are the vertices at height L. Thus, the definition of Λ_{v_i} and the assumption $\lambda_v \in$ Exp(1) give $\Lambda_{v_i} \in$ Exp$(\frac{1}{L+1})$.

For simplicity we write $T_i := T_{v_i}$, $N_i := N_{v_i}$, $n_i := n_{v_i}$ and $\Lambda_i := \Lambda_{v_i}$.

We denote $h_i(v) := h(v) - L$. This is the height in the subtree $T_i,\ i \in \{i, \ldots, b^L\}$ of a vertex $v \in T_i$ i.e. the distance from v to the root v_i of T_i.

In Lemma 2.1 and Lemma 2.2 we use the notation $X(T_i)_{\Lambda_i}$ which we can think of as $X(T_i) - 1$ conditioned on the root label $\lambda_{v_i} = \Lambda_i$.

The conditional expected value of a random variable Z given the number of balls N_i of T_i is denoted $\mathbf{E}_{N_i}(Z) := \mathbf{E}(Z \mid N_i)$.

We denote $\xi_v := \frac{N_v \mu^{-1} \ln N}{N} \cdot e^{-\lambda_v \mu^{-1} \ln N}$ that is used in the later part of the proof when we consider triangular arrays.

Finally we use the notation Ω_L for the σ-field generated by $\{N_v,\ h(v) \leq L\}$.

2.2 Some Lemmas

One of the main ideas of the Main Theorem (that is used frequently in the proofs of the first two lemmas below) is that most vertices in a split tree T_N are close to the level $\mu^{-1} \ln N$ and those vertices that are not can be ignored since they are few enough. Recall that there is a central limit theorem for the depth of nodes in (5) so that “most” nodes lie at $\mu^{-1} \ln N + \mathcal{O}\Big(\sqrt{\ln N}\Big)$. We say a vertex v in a split tree T_N is ”good” if

$$\mu^{-1} \ln N - \ln^{0.6} N \leq h(v) \leq \mu^{-1} \ln N + \ln^{0.6} N,$$

and bad otherwise. In particular a vertex v in the subtree T_i is ”good” if

$$\mu^{-1} \ln N_i - \ln^{0.6} N_i \leq h_i(v) \leq \mu^{-1} \ln N_i + \ln^{0.6} N_i; \tag{11}$$

recall that $h_i(v)$ is the distance from v to the root v_i of the subtree T_i.

For the specific case of the binary search tree that we investigated in [6] there are detailed results for the profile of the nodes in [1] that imply that the bad nodes are bounded by a small error term. For split trees in general there are no results of this type, instead we use large deviations to show that the number of bad nodes in a split tree with N balls is bounded by a small enough error term. Thus, we only have to consider the good vertices.

Recall that $L = \lfloor \beta \log_b \ln N \rfloor$, and Λ_i is the minimum of λ_v in the path from the root σ of T to v_i at height L. Also recall that N_i is the number of balls and n_i is the number of nodes in the subtree T_i rooted at v_i.

Lemma 2.1 *For all subtrees T_i rooted at v_i with $h(v_i) = L$, conditioned on N_i,*

$$\mathbf{E}(X(T_i)_{\Lambda_i} \mid T_i, \Lambda_i) = \frac{n_i}{\mu^{-1} \ln N_i}(1 - e^{-(\mu^{-1} \ln N_i)\Lambda_i}) - \frac{\Upsilon(T_i) - \mu^{-1} n_i \ln N_i}{\mu^{-2} \ln^2 N_i} + \sum_{good\ v \in T_i} \frac{(h_i(v) - \mu^{-1} \ln N_i)^2}{\mu^{-3} \ln^3 N_i} + \mathcal{O}_{L^1}\Big(\frac{N_i}{\ln^{2.2} N_i}\Big),$$

where $\Upsilon(T_i)$ is the total path length of the subtree T_i.

Lemma 2.2 *For all vertices v_i with $h(v_i) = L$, conditioned on N_i,*

$$\mathbf{E}_{N_i}\Big(\mathbf{Var}(X(T_i)_{\Lambda_i} \mid T_i, \Lambda_i)\Big) = \mathcal{O}\Big(\frac{N_i^2}{\ln^3 N_i}\Big).$$

The estimate in Lemma 2.2 is used in the proof of the following result.

Lemma 2.3 *In a split tree with N balls let $v_i, 1 \leq i \leq b^L$, be the vertices at height $L = \lfloor \beta \log_b \ln N \rfloor$ choosing $\beta > \frac{1}{-\log_b \mathbf{E}(V_i^2)-1}$. Then*

$$X(T_N) = \sum_{i=1}^{b^L} \mathbf{E}(X(T_i)_{\Lambda_i} \mid T_i, \Lambda_i) + o_p\Big(\frac{N}{\ln^2 N}\Big).$$

We show by using Chebyshev's inequality that

$$\sum_{i=1}^{b^L} \sum_{good\ v \in T_i} \frac{(h_i(v) - \mu^{-1} \ln N_i)^2}{\mu^{-3} \ln^3 N_i} = \frac{\sigma^2 n}{\ln^2 N} + o_p\Big(\frac{N}{\ln^2 N}\Big). \tag{12}$$

We apply Lemma 2.1, Lemma 2.3 and (12), (and use the Markov inequality) to show that for $\beta > \frac{1}{-\log_b \mathbf{E}V_i{}^2-1}$ (recall this is the constant in L that appears in Lemma 2.3),

$$X(T_N) = \sum_{i=1}^{b^L} \frac{2n_i}{\mu^{-1} \ln N_i} - \sum_{i=1}^{b^L} \frac{\Upsilon(T_i)}{\mu^{-2} \ln^2 N_i} - \frac{1}{\mu^{-1} \ln N} \sum_{i=1}^{b^L} n_i e^{-(\mu^{-1} \ln N)\Lambda_i} + \frac{\sigma^2 n}{\ln^2 N} + o_p\Big(\frac{N}{\ln^2 N}\Big). \tag{13}$$

Lemma 2.4 *Choosing $L = \lfloor \beta \log_b \ln N \rfloor$ for some constant β gives,*

$$\sum_{i=1}^{b^L} n_i e^{-(\mu^{-1} \ln N)\Lambda_i} = \sum_{h(v) \leq L} n_v e^{-(\mu^{-1} \ln N)\lambda_v} + o_p\Big(\frac{N}{\ln^2 N}\Big).$$

Thus, choosing $\beta > \frac{1}{-\log_b \mathbf{E}V_i{}^2-1}$ from (13)

$$X(T_n) = \sum_{i=1}^{b^L} \frac{2n_i}{\mu^{-1} \ln N_i} - \sum_{i=1}^{b^L} \frac{\Upsilon(T_i)}{\mu^{-2} \ln^2 N_i} - \frac{1}{\mu^{-1} \ln N} \sum_{h(v) \leq L} n_v e^{-(\mu^{-1} \ln N)\lambda_v} + \frac{\sigma^2 n}{\ln^2 N} + o_p\Big(\frac{N}{\ln^2 N}\Big). \tag{14}$$

We can simplify the expression of $X(T_N)$ in (14) to (15) below. For simplicity we also change the notation $n_i,\ 1 \leq i \leq b^L$, to $n_v,\ h(v) = L$ and similarly for N_i. Thus, from (14) choosing the constant $\beta > \frac{1}{-\log_b \mathbf{E}(V_i^2)-1}$,

$$X(T_N) = \sum_{h(v)=L} \frac{n_v}{\mu^{-1} \ln N_v} - \frac{1}{\mu^{-1} \ln N} \sum_{h(v) \leq L} n_v e^{-(\mu^{-1} \ln N)\lambda_v} + \frac{n(\mu^{-2}\sigma^2 - \zeta)}{\mu^{-2} \ln^2 N} + o_p\Big(\frac{N}{\ln^2 N}\Big); \tag{15}$$

recall that ζ is the constant in Assumption 2.

2.3 The Main Theorem 1.1 is implied by Theorem 2.1

As in [8] and [6] the proof of Theorem 1.1 will be completed by a classical theorem for convergence of sums of triangular null arrays to infinitely divisible distributions, see e.g. [11, Theorem 15.28]. First we recall the definition of $\xi_v := \frac{N_v\mu^{-1}\ln N}{N}\cdot e^{-\lambda_v\mu^{-1}\ln N}$ above. Because of (15) and the notation of ξ_v we get,

$$\frac{\mu^{-2}\ln^2 N}{\alpha N}\left(X(T_N)-\frac{\alpha N}{\mu^{-1}\ln N}-\frac{\alpha N\ln\ln N}{\mu^{-1}\ln^2 N}\right)$$
$$=-\sum_{h(v)\leq L}\xi_v+\frac{\mu^{-2}\ln^2 N}{\alpha N}\sum_{h(v)=L}\frac{\alpha N_v}{\mu^{-1}\ln N_v}-\mu^{-1}\ln\ln N-\mu^{-1}\ln N+\mu^{-2}\sigma^2-\zeta+o_p(1). \tag{16}$$

Since the N_v's in the sums in (16) are not independent (as we explained in Sub-subsection 1.2.1), the ξ_v's are not independent and thus $\{\xi_v\ ,\ h(v)\leq L\}$ is not a triangular array. For the sake of independence we condition on the N_v's in the sums in (16) and show that(16) converges in distribution to $-W$, where W has an infinitely divisible distribution, that is not depending on the N_v's we conditioned on. Then it follows in the same way as we show for the specific case of the binary search tree in [6] that the normalized $X(T_N)$ in (16) also unconditioned converges in distribution to $-W$.

The next theorem implies Theorem 1.1 above. Recall that the σ-field generated by $\{N_v,\ h(v)\leq L\}$ is denoted Ω_L .

Theorem 2.1 *Suppose that* $N\to\infty$ *and choose any constant* $c>0$. *Conditioning on the* σ*-field* Ω_L *defined above (where* $L=\lfloor\beta\log_b\ln N\rfloor,\ \beta\geq\frac{1}{-\log_b\mathbf{E}(V_i^2)-1}$*), the following hold*

(i) $\sup_v \mathbf{P}\big(\xi_v>x\big|\Omega_L\big)\to 0$ *for every* $x>0$,

(ii) $\sum_{h(v)\leq L}\mathbf{P}\big(\xi_v>x\big|\Omega_L\big)\xrightarrow{p}\nu(x,\infty)=\frac{\mu^{-1}}{x}$ *for every* $x>0$,

(iii) $$\sum_{h(v)\leq L}\mathbf{E}\big(\xi_v\mathbf{1}[\xi_v\leq c]\big|\Omega_L\big)-\frac{\mu^{-2}\ln^2 N}{\alpha N}\sum_{h(v)=L}\frac{\alpha N_v}{\mu^{-1}\ln N_v}$$
$$+\mu^{-1}\ln\ln N+\mu^{-1}\ln N-\mu^{-2}\sigma^2+\zeta\xrightarrow{p}-\mu^{-1}\ln\mu^{-1}+\mu^{-1}-\mu^{-2}\sigma^2-\frac{\sigma^2-\mu^2}{2\mu^2}+\zeta+\mu^{-1}\ln c,$$

(iv) $\sum_{h(v)\leq L}\mathbf{Var}\big(\xi_v\mathbf{1}[\xi_v\leq c]\big|\Omega_L\big)\xrightarrow{p}\mu^{-1}c.$

We show how this theorem will imply Theorem 1.1 above. Let

$$D=\frac{\mu^{-2}\ln^2 N}{\alpha N}\sum_{h(v)=L}\frac{\alpha N_v}{\mu^{-1}\ln N_v}-\mu^{-1}\ln\ln N-\mu^{-1}\ln N+\mu^{-2}\sigma^2-\zeta.$$

We apply [11, Theorem 15.28] to $\sum_{h(v)\leq L}\xi_v+\sum_{i=1}^{N}\xi_i^{'}$ conditioned on Ω_L with $\xi_i^{'}=\frac{-D}{N}$ deterministic. Note that $\frac{D}{N}\to 0$, thus because of (*i*), $\{\xi_v,\ h(v)\leq L\}\bigcup\{\xi_i^{'},\ i\in\{1,\dots,N\}\}$ conditioned on Ω_L is a triangular null array. From (ii) in Theorem 2.1 we have $\frac{d\nu}{dx} = \frac{\mu^{-1}}{x^2}$, hence

$$\int_c^1 x d\nu(x) = \int_c^1 \frac{\mu^{-1}}{x} dx = -\mu^{-1} \ln c \;\text{ and }\; \int_0^c x^2 d\nu(x) = \int_0^c \mu^{-1} dx = \mu^{-1} c.$$

Thus, the right hand sides of (*iii*) and (*iv*) are $b - \int_c^1 x d\nu(x)$ and $\int_0^c x^2 d\nu(x)$ respectively, where b is the constant $b = -\mu^{-1}\ln\mu^{-1} + \mu^{-1} - \mu^{-2}\sigma^2 - \frac{\sigma^2-\mu^2}{2\mu^2} + \zeta$.

The convergence in Theorem 2.1 is in probability, while [11, Theorem 15.28] requires usual convergence. However, if the convergence instead was a.s. in Theorem 2.1, then it would have been easy to see from this theorem that conditionally on Ω_L, the conditions of [11, Theorem 15.28] are fulfilled for $\sum_{h(v)\le L} \xi_v + \sum_{i=1}^N \xi_i^{'}$. Thus, assuming a.s. convergence in Theorem 2.1 instead of convergence in probability, [11, Theorem 15.28] implies that conditioned on Ω_L (and let $N \to \infty$)

$$\sum_{h(v)\le L} \xi_v + \sum_{i=1}^N \xi_i^{'} \xrightarrow{d} W, \tag{17}$$

where W has an infinitely divisible distribution (in particular a weakly 1-stable distribution in this case) with characteristic function

$$\mathbf{E}\Big(e^{itW}\Big) = \exp\left(it(b) + \int_0^\infty (e^{itx} - 1 - itx\mathbf{1}[x<1])d\nu(x)\right);$$

this is (10) in Remark 1.1 (since $b = C + \mu^{-1}(\gamma - 1)$) which can be simplified to (9) in Theorem 1.1. It follows from (17) that conditioning on Ω_L has no influence on the distributional convergence of $\sum_{h(v)\le L} \xi_v + \sum_{i=1}^N \xi_i^{'}$ (unconditioned), since for any continuous bounded function $g : \Re \to \Re$,

$$\mathbf{E}\bigg(g\Big(\sum_{h(v)\le L} \xi_v + \sum_{i=1}^N \xi_i^{'}\Big)\Big|\Omega_L\bigg) \overset{N\to\infty}{\longrightarrow} \mathbf{E}\Big(g(W)\Big).$$

Thus, taking expectation by dominated convergence

$$\mathbf{E}\bigg(g\Big(\sum_{h(v)\le L} \xi_v + \sum_{i=1}^N \xi_i^{'}\Big)\bigg) \overset{N\to\infty}{\longrightarrow} \mathbf{E}\Big(g(W)\Big).$$

This shows that also unconditioned $\sum_{h(v)\le L} \xi_v + \sum_{i=1}^N \xi_i^{'} \xrightarrow{d} W$. Thus, unconditioned $(16) \xrightarrow{d} -W$.

It remains to show that convergence in probability which is the type of convergence in Theorem 2.1 actually is sufficient to get Theorem 1.1 from this theorem. In [6] we proved this fact for the binary search tree in two ways one by using subsequences and the other by using Skorohod's coupling theorem, see e.g. [11, Theorem 3.30]. By analogy these proofs also work for general split trees. Thus, the proof of Theorem 1.1 for $X_v(T)$ is completed. Now it follows easily that the result also holds for $X_e(T)$. One way to see this is to consider $\widehat{T}$ as the tree T with the root deleted. Then there is a natural 1-1 correspondence between edges of T and vertices of $\widehat{T}$ and this correspondence also preserves the record (and cutting) operations. Since it is very unlikely that the root value would decide if values at high levels are records or not it follows that asymptotically $X_e(T)$ and $X_v(T)$ have the same distribution. Thus, the proof of Theorem 1.1 is completed.

2.4 The Method of Proof of Theorem 2.1

This proof is the most technical part of this work (including many lengthy calculations). The idea of the proof of Theorem 2.1 is as for the binary search tree [6, Theorem 2.1] to use the well-known Chebyshev inequality for proving (ii), (iii) and (iv); (i) is very easy to prove.(Recall that Chebyshev's inequality is a useful tool for proving that a random variable is sharply concentrated about its mean value.)

The important observation is that the random variables in (ii), (iii) and (iv) in Theorem 2.1 only depend on the (random) subtree sizes $\{N_v,\ h(v) \le L\}$ since we can express

$$\sum_{h(v)\le L} \mathbf{P}\big(\xi_v > x\big|\Omega_L\big) = (1+o(1))\sum_{k=1}^{L}\sum_{h(v)=k}\frac{1}{m}\ln_+\frac{mN_v}{Nx},$$

$$\sum_{h(v)\le L} \mathbf{E}\big(\xi_v \mathbf{1}[\xi_v \le c]\big|\Omega_L\big) = \sum_{h(v)\le L}\frac{mN_v}{N(m+1)}e^{-\frac{m+1}{m}\ln_+(\frac{mN_v}{Nc})},$$

$$\sum_{h(v)\le L} \mathbf{Var}\big(\xi_v \mathbf{1}[\xi_v \le c]\big|\Omega_L\big) = \sum_{h(v)\le L}\frac{m^2 {N_v}^2}{2mN^2}e^{-\frac{2m+1}{m}\ln_+(\frac{mN_v}{Nc})} + o(1).$$

As we briefly explained in (1) in Sub-subsection 1.2.1 the subtree size N_v for v at height k, is close to $NW_1W_2\dots W_k$, where $W_r,\ r\in\{1,\dots k\}$ are independent random variables distributed as the components V_i in the splitting vector $\mathcal{V}$. Now let $Y_k := -\sum_{r=1}^{k}\ln W_r$ and note that $NW_1W_2\dots W_k = Ne^{-Y_k}$. Recall that in a binary search tree, the splitting vector $\mathcal{V} = (V_1, V_2)$ is distributed as $(U, 1-U)$ where U is a uniform $U(0,1)$ random variable. For this specific case of split tree we frequently used in [6, Theorem 2.1] that the sum Y_k, (where $W_r,\ r\in\{1,\dots k\}$ in this case are i.i.d. uniform $U(0,1)$ random variables) is distributed as a $\Gamma(k,1)$ random variable. For general split trees there is usually no simple distribution function for Y_k; instead we use renewal theory. We define the renewal function

$$U(t) = \sum_{k=1}^{\infty} b^k\mathbf{P}(Y_k \le t) = \sum_{k=1}^{\infty} F_k(t), \tag{18}$$

and also denote $F(t) := F_1(t) = b\mathbf{P}(W_i \le t)$. For $U(t)$ we obtain the following renewal equation

$$U(t) = F(t) + \sum_{k=1}^{\infty}(F_k * F)(t) = F(t) + (U*F)(t).$$

The solution of this equation is given in the following lemma.

Lemma 2.5 *Suppose that* $t\to\infty$ *then the renewal equation* $U(t)$ *in (18) has the solution*

$$U(t) = (\mu^{-1} + o(1))e^t. \tag{19}$$

The result in (19) is then frequently used in the proof of Theorem 2.1

Acknowledgements

I gratefully acknowledge the help and support of Professor Svante Janson, both for introducing me to this problem area and for stimulating discussions and guidance throughout the work.

References

[1] B. Chauvin, M. Drmota, and J. Jabbour Hattab, The profile of binary search trees. *Ann. Applied Probab. 11,* 2001, 1042–1062.

[2] L. Devroye, Universal limit laws for depths in random trees. *Siam J. Comput.* Vol 28, no 2, 1998, 409–432.

[3] L. Devroye, Applications of Stein's method in the analysis of random binary search trees. *Steins method and Applications,* ed. Chen and Barbour, Institute for Mathematical Sciences Lecture Notes Series, Vol. 5, World Scientific Press, Singapore, 2005, 247–297.

[4] M. Drmota, A. Iksanov, M. Moehle and U. Roesler, A limiting distribution for the number of cuts needed to isolate the root of a random recursive tree. *Preprint,* 2006.

[5] W. Feller, *An Introduction to Probability Theory and Its Applications.* Volume II 2nd edition, Wiley, New York, 1971.

[6] C. Holmgren, Random records and cuttings in binary search trees. Submitted 2007.

[7] A. Iksanov, M. Moehle, A probabilistic proof of a weak limit law for the number of cuts needed to isolate the root of a random recursive tree. *Preprint,* 2006.

[8] S. Janson, Random records and cuttings in complete binary trees. *Mathematics and Computer Science III, Algorithms, Trees, Combinatorics and Probabilities* (Vienna, 2004), Birkhäuser, Basel/Switzerland, 2004, 241–253.

[9] S. Janson, Random cutting and records in deterministic and random trees. *Random Struct. Alg. 29* , 2006, 139–179.

[10] J. A. Fill and S. Janson, Quicksort asymptotics. *J. Algorithms 44*, 2002, 4–28.

[11] O. Kallenberg, *Foundations of Modern Probability.* 2nd edition, Springer Verlag, Reading, Mass., 2002.

[12] H. Mahmoud, On the average internal path length of m-ary search trees. *Acta Inform 23*, 1986, 111–117.

[13] H. Mahmoud, B. Pittel, Analysis of the space of search trees under the random insertion algorithm. *J. Algorithms 10,* 1989, no. 1, 52–75.

[14] A. Meir and J. W. Moon, Cutting down random trees. *J. Australian Math. Soc. 11*, 1970, 313–324.

[15] R. Neininger and L. Rüschendorf, On the internal pathlength of d-dimensional quad trees. *Random Structures Algorithms 15*, 1999, no. 1, 25–41.

[16] U. Roesler, On the analysis of stochastic divide and conquer algorithms. Average-case analysis of algorithms (Princeton, NJ, 1998), *Algorithmica 29*, 2001, no. 1-2, 238–261.