An Improved Bound for Random Binary Search Trees with Concurrent Insertions

George Giakkoupis[1] and Philipp Woelfel[2]

1 **INRIA, Rennes, France**
`george.giakkoupis@inria.fr`
2 **University of Calgary, Canada**
`woelfel@cpsc.ucalgary.ca`

Abstract

Recently, Aspnes and Ruppert (DISC 2016) defined the following simple random experiment to determine the impact of concurrency on the performance of binary search trees: n randomly permuted keys arrive one at a time. When a new key arrives, it is first placed into a buffer of size c. Whenever the buffer is full, or when all keys have arrived, an adversary chooses one key from the buffer and inserts it into the binary search tree.

The ability of the adversary to choose the next key to insert among c buffered keys, models a distributed system, where up to c processes try to insert keys concurrently. Aspnes and Ruppert showed that the expected average depth of nodes in the resulting tree is $O(\log n + c)$ for a *comparison-based* adversary, which can only take the relative order of arrived keys into account. We generalize and strengthen this result. In particular, we allow an adversary that knows the actual values of all keys that have arrived, and show that the resulting expected average node depth is $D_{avg}(n) + O(c)$, where $D_{avg}(n) = 2\ln n - \Theta(1)$ is the expected average node depth of a random tree obtained in the standard unbuffered version of this experiment. Extending the bound by Aspnes and Ruppert to this stronger adversary model answers one of their open questions.

1998 ACM Subject Classification E.1 Data Structures

Keywords and phrases Random binary search tree, buffer, average depth, concurrent data structures

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.38

1 Introduction

Consider the following random procedure, in which distinct keys from a totally ordered universe are inserted into an (internal) binary search tree (BST). Multiple keys arrive in random order, and whenever a key arrives, it is first placed into a buffer of fixed size c. Upon each key arrival, an adversary may remove one of the keys from the buffer, and insert it into the BST. (The adversary may also decide not to insert a key, if the buffer is not full.) We are interested in performance affecting properties of the resulting BST, such as its height or average node depth.

This random experiment has recently been studied by Aspnes and Ruppert [1], to understand how concurrency influences the performance of search trees in a distributed setting: In an asynchronous shared memory system, multiple processes may concurrently try to insert keys into a BST data structure.

The order in which concurrent insertions succeed can depend on unpredictable system factors. For example, cache effects or memory locality on NUMA architectures can heavily

influence the speed with which synchronization primitives, and thus insertions take effect. Events such as context switches can delay the operations by some processes, while other processes that may have locked parts of the data structure may be able to execute several operations in rapid succession. Therefore, the order in which concurrent data structure operations succeed can be arbitrary, and is usually analyzed under worst-case assumptions. For randomized algorithms or random inputs, an *adversary* is used to model how this order is influenced by random events. In analogy to the worst-case analysis of sequential algorithms, it is natural to make pessimistic adversary assumptions, i.e., consider the worst conceivable influence that random events can have on the operation order.

A buffered search tree with buffer size c as described above, models such a distributed scenario, where up to c processes insert random keys concurrently. At any point, there are up to c insert operations pending (corresponding to the keys in the buffer), and the adversary decides which of those insert operations succeeds first.

Aspnes and Ruppert [1] analyzed the buffered search tree for a *comparison based* adversary, which can base its decisions only on the relative order of the keys that have arrived so far. They showed that for a random arrival sequence of n distinct keys, the resulting BST has an expected average node depth of $O(c + \log n)$, and that this bound is asymptotically tight. They also proved that there is a comparison based adversary that can achieve an expected height of $\Omega\big(c \cdot \log(n/c)\big)$.

Note that the height of a BST corresponds to the worst-case search time, and the average node depth corresponds to the average-case search time for successful searches.

Contrary to the distributed case (buffered tree), in the sequential setting the expected height and the expected average node depth are known up to constant additive terms. For example, the expected average node depth of a BST obtained by inserting keys $\{1, \ldots, n\}$ in random order is [6]

$$D_{avg}(n) = 2(1 + 1/n)H_n - 2, \text{ where } \ln(n+1) < H_n < (\ln n) + 1.$$

It seems appropriate to also analyze height and average depth as precisely as possible for a distributed setting.

The adversary considered by Aspnes and Ruppert [1] does not fully reflect a worst-case scenario, as it can make decisions only based on the relative order of elements, but cannot distinguish their absolute values. It is not clear whether adversarial decisions based on, for example, the pairwise differences between consecutive keys in the ordered list of buffered elements, can lead to a significantly increased average node depth. Consequently, Aspnes and Ruppert suggested that "it might be interesting to see whether even stronger malicious adversaries could force the height or average depth of random trees to grow higher by using the actual values of he keys" [1]. They suggested an example for a buffer size of $c = 3$, where keys are drawn uniformly at random from the interval $[0, 1]$: If the first three keys chosen are 0.03, 0.45, and 0.54, then a good strategy for an adversary that can see absolute values would be to insert 0.03 as the root. On the other hand, if the initial three keys in the buffer were 0.46, 0.55, and 0.97, the adversary would be better off by inserting 0.97, first.

Results. We provide a negative answer to the question posed by Aspnes and Ruppert [1], whether an adversary can achieve a significantly increase in the average node depth by using knowledge about the values of keys in the buffer. At the same time, we refine Aspnes and Ruppert's upper bound of $O(\log n + c)$ on the expected average node depth, to $D_{avg}(n) + O(c)$. As a result, a buffer of size $c = o(\log n)$ does not negatively affect the constant factor in the expected average node depth.

We consider a strong adversary, which can at any time base its next insertion decision on the exact values of all keys that have arrived so far. In a distributed system, where the buffer corresponds to pending concurrent insert operations, this translates to the strongest reasonable adversary, namely one which can base its decisions on all past random events, but not future ones.

Consider an arrival sequence of n arbitrary distinct keys chosen from a totally ordered universe and permuted randomly. Let $\Delta(n, c)$ denote the expected average node depth obtained for a buffered BST given the strong adversary, if the buffer has size c. Note that $c = 1$ corresponds to the unbuffered case, as the adversary has to move a key from the buffer whenever the buffer is full. Hence, $\Delta(n, 1) = D_{avg}(n)$ is the expected average node depth in the standard sequential case. We will prove the following result.

▶ **Theorem 1.** *$\Delta(n, c) = \Delta(n, 1) + O(c)$, for any $c \in \{2, \ldots, n\}$.*

Note that the result of Aspnes and Ruppert [1] states that $\Delta'(n, c) = O(\Delta'(n, 1) + c)$, where Δ' is defined as Δ but for the weaker comparison based adversary.

Related Work. Binary search trees are among the most fundamental data structures, and their properties have been studied extensively. It has been known for a long time that the expected average depth of nodes for trees obtained by inserting a random permutation of n distinct elements is $O(\log n)$ [2, 10]. Many additional details, such as higher moments, are known about the distribution of node depths [6]. Other parameters of random BSTs have also been determined precisely. For example, a sequence of works [9, 3, 8] determined the expected height as $(4.311\ldots) \cdot \log n - (1.953\ldots) \cdot \log\log n + O(1)$.

The buffered search tree scenario considered in this paper was introduced by Aspnes and Ruppert [1]. They describe it in terms of a card game, where a player takes c cards from a shuffled deck of n cards, then chooses one of the cards in her hand to insert into the BST, and replaces that card with a new one from the deck. This continues until the deck has been depleted, upon which the player inserts all remaining cards in her hand into the tree. The authors point out that a complementary approach is the smoothed analysis of the expected BST height [7].

Concurrent Data Structure Context. To understand the context of this work, it is helpful to know how efficient concurrent data structures, such as binary search trees, may be implemented. There are several techniques to avoid conflicts, when multiple processes access a search tree (or other data structure) concurrently. The simplest one is to use "coarse grained locking", where a mutual exclusion algorithm (lock) protects the entire data structure. This way, only one process can access the data structure at a time, while other processes have to wait.

But such lock-based solutions are inefficient, and not fault tolerant. The "Read-Copy-Update" technique is an example of a more efficient and fault tolerant solution: The address of the root of the tree is stored in a Compare-And-Swap (CAS) object C. This is a standard shared memory primitive that supports a $C.\texttt{CAS}(old, new)$ operation, which atomically changes the value of C to new, provided its current value is old. To insert a node v, a process first creates a copy of the search path from the root to the node u of which v will become a child. Then the process executes a $C.\texttt{CAS}(old, new)$ operation, where old is the address of the original root, and new is the address of the root on the search path copy. This CAS only succeeds, if no other insertion took effect (i.e., no other CAS succeeded) in the meantime. If it fails, the process repeats its insertion attempt. This method is lock-free, guaranteeing

progress even if processes crash. If the tree only supports insertions and no other update operations (such as deletions), then more efficient implementations based on fine grained synchronization are possible (e.g., using one CAS object or lock for each child pointer).

There are several other common solutions, e.g., based on transactional memory. But most of these solutions have in common that the speed of an insertion attempt may depend on the length of the search path. Since "fast" attempts have a higher chance of being successful, it is reasonable to assume that the order of concurrent insertions depends on the values of the involved keys.

2 Analysis

Our analysis works for a slightly stronger adversary, which in addition to the keys in the buffer, also knows the set (but not the order) of all future keys to arrive. In other words, the adversary knows at any point the ranks of the keys that have arrived so far, with respect to the set of all keys in the complete arrival sequence. In that setting we can assume w.l.o.g. that the key arrival sequence is a random permutation over $\{1, \ldots, n\}$. The first $c-1$ of the keys are stored in a buffer. For each following key k that arrives, first k is inserted to the buffer, and then one of the c keys in the buffer is removed and inserted into the tree. (It is obvious that the adversary can gain no advantage by inserting a key from the buffer "early", i.e., when the buffer is not full and more keys are going to arrive.) After all keys have arrived, the c keys remaining in the buffer are inserted into the tree as well.

Proof Overview. We use the fact that the average node depth in any tree is equal to the average number of descendants of all nodes in the tree. Let $k_1, \ldots, k_n$ denote the keys in the order in which they arrive (so, $k_1, \ldots, k_n$ is a random permutation of $\{1, \ldots, n\}$). We bound the expected number of descendants of each key k_i in the final tree. For $i < c$ we use the trivial bound of n on the number of descendants of k_i. Next we focus on the case $i \geq c$. We bound the expected number of descendants of k_i in the final tree, given the current tree and the set of keys in the buffer just before k_i arrives; at that time, the value of k_i is not yet determined, so it is equally likely to be any of the remaining keys. We observe that the number of descendants of k_i in the final tree is maximized if key k_i is inserted to the tree as soon as it arrives, rather than stored in the buffer and inserted at a later time. So, it suffices to bound the expected number of descendants k_i would have, if it were inserted immediately. This expected value depends only on the set of the $i-1$ keys that arrived before k_i, and the subset of them that have been inserted into the tree; in particular, it does not depend on the order in which keys arrived or the order in which keys were inserted into the tree. From this bound on the conditional expectation on the number of descendants of k_i, given the $i-1$ keys that arrived before k_i and the $i-c$ keys already in the tree, we obtain a bound on the corresponding unconditional expectation, by letting the first $i-1$ keys be random, and assuming that an arbitrary (worst-case) $(i-c)$-subset of them has been inserted into the tree. From this bound for each $i \geq c$ (and the trivial bound of n for $i < c$), and from the linearity of expectation, we obtain a bound on the expectation of the average number of descendants of all nodes.

Detailed Proof. We now present the detailed proof of Theorem 1. As already mentioned, $k_1, \ldots, k_n$ is a random permutation of $\{1, \ldots, n\}$ denoting the order in which keys arrive. We assume w.l.o.g. that no key is inserted into the tree, unless the buffer is full or all keys have arrived. I.e., the insertions evolve in rounds as follows: In round $j \in \{1, \ldots, c-1\}$, key

k_j is added to the buffer. In round $j \in \{c, \dots, n\}$, first key k_j is added to the buffer, and then some key ℓ_j is removed from the buffer and inserted into the search tree. Finally, in each round $j \in \{n+1, \dots, n+c-1\}$, one key ℓ_j is removed from the buffer and inserted into the tree.

For $i \in \{0, \dots, n+c-1\}$, let X_i be the set of keys that have arrived by the end of round i, and let Y_i be the set of keys that the tree contains at the end of round i ($X_0 = Y_0 = \emptyset$). Then $X_i = \{k_1, \dots, k_i\}$ for $i \in \{0, \dots, n\}$, and $X_i = \{1, \dots, n\}$ for $i > n$. Moreover, $Y_i = \emptyset$ for $i \in \{1, \dots, c-1\}$, and $Y_i = \{\ell_c, \dots, \ell_i\}$ for $i \in \{c, \dots, n+c-1\}$. At the end of round i, the set of keys in the buffer is $X_i \setminus Y_i$. This set contains i elements at the end of round $i \in \{0, \dots, c-1\}$, $c-1$ elements at the end of round $i \in \{c, \dots, n\}$, and $n+c-i-1$ elements at the end of round $i \in \{n+1, \dots, n+c-1\}$. Hence,

$$|X_i \setminus Y_i| = \min\{i, c-1, n+c-i-1\}. \tag{1}$$

Let $x_i^1, \dots, x_i^i$ denote the elements of X_i ordered by increasing key values. For convenience we also define $x_i^0 = 0$ and $x_i^{i+1} = n+1$. Further, let $Y_i' = Y_i \cup \{0, n+1\}$.

We are interested in bounding the number of descendants of each key in the final tree. For $i \in \{0, \dots, n-1\}$, define the random variable

$$\delta_i^c = \min\{y \in Y_i' \mid y > k_{i+1}\} - \max\{y \in Y_i' \mid y < k_{i+1}\}.$$

As we will show below in the proof of Claim 2, the number of descendants of k_{i+1} in the search tree with buffer size c is at most $\delta_i^c - 2$. Moreover, for $c = 1$, the number of descendants is exactly $\delta_i^1 - 2$. Since the average node depth of a search tree equals the average number of descendants of each node, we can upper bound the expected average node depth as the expected average of all values $\delta_i^c - 2$. And for $c = 1$, these two quantities match.

▶ **Claim 2.**

(a) $\Delta(n,1) = \dfrac{1}{n} \cdot \sum_{i=0}^{n-1} \mathbf{E}[\delta_i^1] - 2$; *and*

(b) $\Delta(n,c) \le \dfrac{1}{n} \cdot \sum_{i=0}^{n-1} \mathbf{E}[\delta_i^c] - 2$, *for any* $c \ge 2$.

Proof. For $i \in \{0, \dots, n-1\}$ let $desc(i)$ denote the set of descendants of key k_i in the final search tree, and let $depth(j)$ denote the depth of node k_j in the final tree. Then the average node depth of the final tree is

$$\frac{1}{n} \cdot \sum_{j=1}^{n} depth(j) = \frac{1}{n} \cdot \sum_{j=1}^{n} \big|\{i \in \{1, \dots, n\} : k_j \in desc(i)\big| = \frac{1}{n} \cdot \sum_{j=1}^{n} |desc(j)|.$$

We will now show for any $i \in \{0, \dots, n-1\}$ that $|desc(i+1)| \le \delta_i^c - 2$, and moreover, for a buffer of size $c = 1$, $|desc(i+1)| = \delta_i^1 - 2$. The claim follows immediately from that.

Key k_{i+1} arrives in the buffer in round $i+1$. Suppose it is moved into the tree in round $j+1 \ge i+1$. Then at that point the tree contains the elements in Y_j. When that happens, the largest element in the tree that is smaller than k_{i+1} (if it exists) is $a_j = \max\{y \in Y_j \mid y < k_{i+1}\}$, and the smallest element in the tree that is larger than k_{i+1} (if it exists) is $b_j = \min\{y \in Y_j \mid y > k_{i+1}\}$. The search path to the position where k_{i+1} gets inserted goes through a_j and b_j. Hence, in the final tree, the search path to any descendant of k_{i+1} also goes through those two nodes. Moreover, any key that is larger than a_j and smaller than b_j will follow the search path to k_{i+1} and become a descendant of k_{i+1}. Define

$a_j = 0$ respectively $b_j = n+1$ if Y_j contains no element smaller respectively larger than k_{i+1}. Then we obtain $desc(i+1) = \{a_j + 1, \ldots, b_j - 1\} \setminus \{k_{i+1}\}$, and thus

$$|desc(i+1)| = b_j - a_j - 2.$$

Now observe that if the buffer has size $c = 1$, then $j = i$, because key k_i gets inserted into the tree in the same round as it arrives. Since $a_i = \max\{y \in Y'_i \mid y < k_{i+1}\}$ and $b_i = \min\{y \in Y'_i \mid y > k_{i+1}\}$ (recall that $Y'_i = Y_i \cup \{0, n+1\}$) we obtain $\delta_i^1 = b_i - a_i$, and so $|desc(i+1)| = \delta_i^1 - 2$.

For $c > 1$ we may have $j > i$, but in any case $Y_i \subseteq Y_j$. Hence, $a_i \leq a_j$ and $b_i \geq b_j$, and so we obtain $\delta_i^c = b_i - a_i \geq b_j - a_j = |desc(i+1)| - 2$. ◀

Later, in Claim 4, we will bound the expectation of δ_i^c, given X_i and Y_i. To do so, we will use the following simple observation.

▶ **Claim 3.** *For any $i \in \{c-1, \ldots, n-1\}$, $\delta_i^c \leq \max_{j \in \{0,\ldots,i-c+1\}} (x_i^{j+c} - x_i^j)$.*

Proof. Let $y_i^a = \max\{y \in Y'_i \mid y < k_{i+1}\}$. Then $y_i^{a+1} = \min\{y \in Y'_i \mid y > k_{i+1}\}$, and $\delta_i^c = y_i^{a+1} - y_i^a$. Since $Y'_i \subseteq X_i \cup \{0, n+1\}$, elements y_i^a and y_i^{a+1} are both in $X_i \cup \{x_i^0, x_i^{i+1}\}$. Let $\ell \in \{0, \ldots, i\}$ such that $x_i^\ell = y_i^a$, and $r \in \{\ell+1, \ldots, i+1\}$ such that $x_i^r = y_i^{a+1}$.

First assume that $r - \ell > c$. Since y_i^a and y_i^{a+1} appear consecutively in the ordered list of elements in Y_i, none of $x_i^{\ell+1}, \ldots, x_i^{r-1}$ is in Y_i. Recall that $X_i \setminus Y_i$ is the set of elements in the buffer at the end of round i. Hence, all $r - \ell - 1 \geq c$ elements $x_i^{\ell+1}, \ldots, x_i^{r-1}$ are in the buffer at the end of round i. But at the end of a round, the buffer can contain at most $c-1$ elements, and we have a contradiction.

Therefore, $r - \ell \leq c$. Then there exists an index $j \in \{0, \ldots, \ell\}$ with $j + c \in \{r, \ldots, i+1\}$. In this case $x_i^j \leq x_i^\ell = y_i^a$ and $x_i^{j+c} \geq x_i^r = y_i^{a+1}$, and thus $x_i^{j+c} - x_i^j \geq y_i^{a+1} - y_i^a = \delta_i^c$. This proves the claim. ◀

For a set $S = \{s_1, \ldots, s_m\} \subseteq \{1, \ldots, n\}$, where $s_1 < \cdots < s_m$, let

$$\gamma(S) = \sum_{i=0}^{m} (s_{i+1} - s_i)^2, \text{ where } s_0 = 0 \text{ and } s_{m+1} = n+1.$$

For any $i \in \{0, \ldots, n-1\}$ we will now bound the expected value of δ_i^c as a function of $\gamma(Y_i)$, given the sets X_i and Y_i. That bound is tight for the case $c = 1$.

▶ **Claim 4.** *For any $i \in \{c-1, \ldots, n-1\}$,*

(a) $\mathbf{E}[\delta_i^c \mid X_i, Y_i] \leq \dfrac{\gamma(Y_i) - n - 1}{n - i}$;

(b) $\mathbf{E}[\delta_i^1 \mid X_i] = \dfrac{\gamma(X_i) - n - 1}{n - i}$.

Proof. Let $z = i - c + 1 = |Y_i|$, let the set of ordered elements in Y_i be $y_i^1, \ldots, y_i^z$, and let $y_i^0 = 0$ and $y_i^{z+1} = n+1$. Therefore,

$$\sum_{j=0}^{z} (y_i^{j+1} - y_i^j) = y_i^{z+1} - y_i^0 = n+1. \tag{2}$$

Let $\sigma(i)$ be the unique index in $\{0, \ldots, z\}$ such that $y_i^{\sigma(i)} < k_{i+1} < y_i^{\sigma(i)+1}$. Then $y_i^{\sigma(i)} = \max\{y \in \{y_i^0, \ldots, y_i^{z+1}\} \mid y < k_{i+1}\}$ and $y_i^{\sigma(i)+1} = \min\{y \in \{y_i^0, \ldots, y_i^{z+1}\} \mid y > k_{i+1}\}$. Hence,

$$\delta_i^c = y_i^{\sigma(i)+1} - y_i^{\sigma(i)}.$$

Recall that X_i is the set of keys arrived by the end of round i. Hence, given X_i, the element k_{i+1}, which arrives in round $i+1$, is uniformly distributed over $\{1,\dots,n\} \setminus X_i$ (the tag $(*)$ below indicates where this fact is being used). Therefore,

$$\begin{aligned}
\mathbf{E}[\delta_i^c \mid X_i, Y_i] &= \sum_{j=0}^{z} \Pr(\sigma(i)=j) \mid X_i, Y_i) \cdot (y_i^{j+1} - y_i^j) \\
&= \sum_{j=0}^{z} \Pr\left(k_{i+1} \in \{y_i^j+1,\dots,y_i^{j+1}-1\} \,\middle|\, X_i, Y_i\right) \cdot (y_i^{j+1} - y_i^j) \\
&\overset{(*)}{=} \sum_{j=0}^{z} \frac{|\{y_i^j+1,\dots,y_i^{j+1}-1\} \setminus X_i|}{|\{1,\dots,n\} \setminus X_i|} \cdot (y_i^{j+1} - y_i^j) \\
&\le \sum_{j=0}^{z} \frac{y_i^{j+1} - y_i^j - 1}{|\{1,\dots,n\} \setminus X_i|} \cdot (y_i^{j+1} - y_i^j) \qquad (3) \\
&= \sum_{j=0}^{z} \frac{(y_i^{j+1} - y_i^j)^2 - y_i^{j+1} + y_i^j}{n-i} \\
&\overset{(2)}{=} \sum_{j=0}^{z} \frac{(y_i^{j+1} - y_i^j)^2}{n-1} - \frac{n+1}{n-i} \\
&= \frac{\gamma(Y_i) - n - 1}{n-i}. \qquad (4)
\end{aligned}$$

This proves Part (a) of the claim.

For Part (b) note that if the buffer has size $c = 1$, then $X_i = Y_i$. Hence, $\{y_i^j+1,\dots,y_i^{j+1}-1\}$ contains no element in X_i, and so inequality (3) above holds as an equality. Now Part (b) follows from substituting x_i^j with y_i^j and X_i with Y_i in the above bound. ◀

According to Claim 2, we have

$$\Delta(n,c) - \Delta(n,1) \le \frac{1}{n} \sum_{i=0}^{n-1} (\mathbf{E}[\delta_i^c - \delta_i^1]).$$

In the following we will bound the expectation of the maximum of $\gamma(Y_i) - \gamma(X_i)$ for all sets $Y_i \subseteq X_i$ of size $i-c+1$, for $i \in \{c-1,\dots,n-1\}$. This implies an upper bound on $\mathbf{E}[\gamma(Y_i) - \gamma(X_i)]$, and thus by Claim 4 on $\mathbf{E}[\delta_i^c - \delta_i^1]$.

▶ **Lemma 5.** *For any $i \in \{c-1,\dots,n-1\}$,*

$$\mathbf{E}[\delta_i^c - \delta_i^1] = O\left(\frac{n^2c^2 + n^2 c \log^2 i}{i^2(n-i)}\right).$$

First we show a high probability upper bound on the differences $x_i^{j+d} - x_i^j$, for all values of $j \in \{0,\dots,i\}$ and $d \in \{1,\dots,i-j+1\}$ in Claim 6 below. This allows us to bound for a random set $S \subseteq \{1,\dots,n\}$ of m elements, the expectation of $\big(\gamma(S \setminus C) - \gamma(S)\big)$ for the "worst" subset C of S of a given size. This bound is stated in Lemma 7 below. Using this bound, we will then prove Lemma 5.

▶ **Claim 6.** *Let S be a set of $m \in \{1,\dots,n\}$ elements chosen at random from $\{1,\dots,n\}$ without replacement. Let $s_1,\dots,s_m$ be the elements of S in sorted order, and let $s_0 = 0$ and $s_{m+1} = n+1$. Then for any $d \in \{1,\dots,m+1\}$ and $j \in \{0,\dots,m-d+1\}$,*

$$\Pr\left(s_{j+d} - s_j > \frac{8n(d + \ln m)}{m}\right) < e^{-3d} \cdot m^{-3}.$$

Proof. Let

$$b = 8n(d + \ln m)/m.$$

The distribution of the difference $s_{j+d} - s_j$ does not depend on j (see Claim 9 in the appendix), so we can assume $j = 0$. Thus, it suffices to show

$$\Pr(s_d > b) < e^{-3d} \cdot m^{-3}. \tag{5}$$

If $b > n$, then the claim is trivially true, so assume $b \le n$. Let R_t, $t \in \{1, \dots, m\}$, be an indicator random variable, where $R_t = 1$ if and only if the t-th element chosen for S has a value of at most b. Further, let $R = R_1 + \cdots + R_m$. Then $\Pr(R_t = 1) = b/n$, and so

$$\mathbf{E}[R] = \sum_{t=1}^{m} \mathbf{E}[R_j] = m \cdot \frac{b}{n} = 8(d + \ln m).$$

Recall that $s_0, s_1, \dots, s_{m+1}$ is an increasing sequence. Therefore, R is the smallest index in $\{1, \dots, m\}$ such that $s_{R+1} > b$. Hence, $s_d > b$ is equivalent to $R < d$, so according to (5) it suffices to show that

$$\Pr(R < d) < e^{-3d} \cdot m^{-3}. \tag{6}$$

The random variables R_t are negatively associated [5, Section 3.1(c)], so we can use Chernoff Bounds [4, Theorem 3.1] to obtain

$$\begin{aligned}\Pr(R < d) = \Pr\left(R < \frac{E[R]}{8} - \ln m\right) &\le \Pr\big(R < E[R](1 - 7/8)\big) < e^{-(7/8)^2 \cdot E[R]/2} \\ &\le e^{-(7/8)^2 \cdot 8(d + \ln m)/2} < e^{-3(d + \ln m)} = e^{-3d} \cdot m^{-3}.\end{aligned}$$

This proves (6), and thus the claim. ◀

▶ **Lemma 7.** *Let S be a set of $m \in \{1, \dots, n\}$ elements chosen at random from $\{1, \dots, n\}$ without replacement. Then for any $\tau \in \{1, \dots, m\}$,*

$$\mathbf{E}\left[\max_{\substack{C \subseteq S \\ |C| = \tau}} \gamma(S \setminus C) - \gamma(S)\right] = O\left(\frac{n^2\tau^2 + n^2\tau \log^2 m}{m^2}\right).$$

Proof. Let $s_1, \dots, s_m$ be the elements of S in sorted order, and let $s_0 = 0$ and $s_{m+1} = n + 1$. Define the following event,

$$\text{Event } A\text{:} \quad \forall d \in \{1, \dots, m+1\}, i \in \{0, \dots, m - d + 1\} : s_{i+d} - s_i \le \frac{8n(d + \ln m)}{m}.$$

Using the union bound and applying Claim 6, we obtain

$$\begin{aligned}\Pr(\neg A) &= \Pr\left(\exists d \in \{1, \dots, m+1\}, i \in \{0, \dots, m - d + 1\} : s_{i+d} - s_i > \frac{8n(d + \ln m)}{m}\right) \\ &\le \sum_{d=1}^{m+1} \sum_{i=0}^{m-d+1} \Pr\left(s_{i+d} - s_i > \frac{8n(d + \ln m)}{m}\right) \le \sum_{d=1}^{m+1} \sum_{i=0}^{m-d+1} e^{-3d} \cdot m^{-3} = O(m^{-2}).\end{aligned} \tag{7}$$

Now consider an arbitrary set $C = \{s_{i_1}, \ldots, s_{i_\tau}\} \subseteq S$. Let $I = \{i_1, \ldots, i_\tau\}$, and partition I into maximal subsets $I_1, \ldots, I_\ell$ of consecutive indices. Further, for $j \in \{1, \ldots, \ell\}$ let $\alpha_j = \min I_j - 1$ and $d_j = |I_j| + 1$. Thus,

$$C = \bigcup_{j=1}^{\ell} \{s_{\alpha_j+1}, s_{\alpha_j+2}, \ldots, s_{\alpha_j+d_j-1}\},$$

and so for $S' = S \cup \{s_0, s_{m+1}\}$,

$$S' \setminus C = \{s_0, s_1, \ldots, s_{\alpha_1}, s_{\alpha_1+d_1}, s_{\alpha_1+d_1+1}, \ldots, s_{\alpha_2}, s_{\alpha_2+d_2}, \ldots, s_{\alpha_\ell}, s_{\alpha_\ell+d_\ell}, \ldots, s_{m+1}\}.$$

Each pair (s_i, s_{i+1}) over $S' \setminus C$ contributes $(s_{i+1} - s_i)^2$ to the sum $\gamma(S \setminus C)$, and it contributes the same amount to the sum $\gamma(S)$. On the other hand, each pair $(s_{\alpha_j}, s_{\alpha_j+d_j})$ over $S' \setminus C$ contributes $(s_{\alpha_j+d_j} - s_{\alpha_j})^2$ to $\gamma(S \setminus C)$, while the corresponding contribution to $\gamma(S)$ is potentially much smaller (precisely it is $\sum_{t=1}^{d_j}(s_{\alpha_j+t} - s_{\alpha_j+t-1})^2$). Therefore, ignoring these latter contributions to $\gamma(S)$, we can upper bound the difference $\gamma(S \setminus C) - \gamma(S)$ as follows:

$$\gamma(S \setminus C) - \gamma(S) \leq \sum_{j=1}^{\ell} (s_{\alpha_j+d_j} - s_{\alpha_j})^2.$$

Now, if event A occurs, then $s_{\alpha_j+d_j} - s_{\alpha_j} \leq 8n(d_j + \ln m)/m$, and so in this case

$$\gamma(S \setminus C) - \gamma(S) \leq \sum_{j=1}^{\ell} \left(\frac{8n(d_j + \ln m)}{m}\right)^2 \leq \sum_{j=1}^{\ell} \frac{128n^2\left({d_j}^2 + \ln^2 m\right)}{m^2}, \quad (8)$$

where the last inequality was obtained by using the fact that $(a+b)^2 \leq 2(a^2+b^2)$. Now recall that $d_j = |I_j| + 1$, so $d_1 + \cdots + d_\ell - \ell = |I_1 \cup \cdots \cup I_\ell| = |C| = \tau$, and $\ell \leq |C| = \tau$ (because C contains ℓ sets of consecutive indices. Therefore,

$$d_1 + \cdots + d_\ell = \tau + \ell \leq 2\tau, \text{ and } {d_1}^2 + \cdots + {d_\ell}^2 \leq (d_1 + \cdots + d_\ell)^2 = (\tau + \ell)^2 \leq 4\tau^2. \quad (9)$$

Observe that if event A occurs, bound (8) is true for every set $C \subseteq S$ of size τ. Thus, if A occurs, then combining (8) and (9), we obtain

$$\max_{\substack{C \subseteq S \\ |C|=\tau}} \gamma(S \setminus C) - \gamma(S) \leq \frac{128n^2\left(4\tau^2 + \ell \ln^2 m\right)}{m^2} \leq \frac{128n^2\left(4\tau^2 + \tau \ln^2 m\right)}{m^2}. \quad (10)$$

If event A does not occur, then we can use the trivial bound

$$\max_{\substack{C \subseteq S \\ |C|=\tau}} \gamma(S \setminus C) - \gamma(S) \leq \max_{\substack{C \subseteq S \\ |C|=\tau}} \gamma(S \setminus C) \leq \gamma(\emptyset) = (n+1)^2. \quad (11)$$

According to (7), event $\neg A$ occurs with probability $O(m^{-2})$. Hence, (10) and (11) imply

$$\begin{aligned}\mathbf{E}[\max_{\substack{C \subseteq S \\ |C|=\tau}} \gamma(S \setminus C) - \gamma(S)] &\leq \frac{128n^2\left(4\tau^2 + \tau \ln^2 m\right)}{m^2} + (n+1)^2 \cdot O(m^{-2}) \\ &= O\left(\frac{n^2\left(\tau^2 + \tau \log^2 m\right)}{m^2}\right).\end{aligned}$$

◀

We can now prove Lemma 5.

Proof of Lemma 5. Let $\tau = c - 1$, and recall $i \in \{c-1, \ldots, n-1\}$. According to Claim 4,

$$\mathbf{E}[\delta_i^c - \delta_i^1 \mid X_i, Y_i] \le \frac{\gamma(Y_i) - \gamma(X_i)}{n-i}.$$

Recall that X_i is a set of i elements in $\{1, \ldots, n\}$ chosen uniformly at random without replacement. Moreover, $Y_i \subseteq X_i$ is chosen by the adversary, where $|X_i \setminus Y_i| = \tau$ by (1). Hence, from Lemma 7 (substituting X_i for S and i for m) we obtain

$$\mathbf{E}[\delta_i^c - \delta_i^1] \le \frac{1}{n-i} \cdot \mathbf{E}\left[\max_{\substack{C \subseteq X_i \\ |C| = \tau}} \gamma(X_i \setminus C) - \gamma(X_i)\right] = O\left(\frac{1}{n-1} \cdot \frac{n^2\tau^2 + n^2\tau \log^2 i}{i^2}\right).$$

Since $\tau < c$, the claim follows. ◀

For large i, close to n, the bound in Lemma 5 is too weak. Instead we use the following result which holds for $i \ge n/2$:

▶ **Lemma 8.** *Let $c < n/2$ and $i \in \{\lceil n/2 \rceil, \ldots, n-1\}$. Then $\mathbf{E}[\delta_i^c] = O(c + \log n)$.*

Proof. Define

$$\text{Event } B: \quad \forall j \in \{0, \ldots, i-c+1\} : x_i^{j+c} - x_i^j \le 16(c + \ln n).$$

If B occurs, then by Claim 3, $\delta_i^c \le 16(c + \ln n)$. On the other hand, if B does not occur, then we will use the trivial bound $\delta_i^c \le n - i < n$. We will show below that

$$\Pr(\neg B) = O(n^{-2}). \tag{12}$$

Hence, we obtain

$$\mathbf{E}[\delta_i^c] \le 16(c + \ln n) + n \cdot O(n^{-2}) = O(c + \ln n).$$

Next we prove (12).

Recall that $k_1, \ldots, k_i$ are the first i elements that arrive, i.e., they are chosen uniformly at random without replacement from $\{1, \ldots, n\}$. Moreover, $x_i^1, \ldots, x_i^i$ is the sorted order of those elements. Thus, by Claim 6, for any $j \in \{0, \ldots, i-c+1\}$,

$$\Pr\left(x_i^{j+c} - x_i^j > \frac{8n(c + \ln i)}{i}\right) < e^{-3c} \cdot i^{-3}.$$

Using the assumption of this lemma, $n/2 \le i < n$, and a union bound on j, we obtain

$$\Pr\left(\exists j \in \{0, \ldots, i-c+1\} : x_i^{j+c} - x_i^j > 16(c + \ln n)\right) < e^{-3c} \cdot (n/2)^{-3}(i - c + 2) = O(n^{-2}).$$

This proves (12), and thus the lemma. ◀

We now combine Lemmas 5 and 8 to prove our main result, i.e., $\Delta(n, c) = \Delta(n, 1) + O(c)$.

Proof of Theorem 1. Clearly, $\Delta(n, c) < n$ for all c. Therefore, if $c \ge n/2$, the theorem is trivially true. Hence, assume $c < n/2$, and let

$$b = n - \frac{cn}{2(c + \log n)} > n - \frac{cn}{2c} = n/2 > c.$$

According to Claim 2, we have

$$\Delta(n,c) - \Delta(n,1) \leq \frac{1}{n} \cdot \sum_{i=0}^{n-1} \mathbf{E}[\delta_i^c - \delta_i^1]. \tag{13}$$

It may be useful to recall the reason for inequality (13). As explained in the proof of Claim 2, δ_i^1 is equal to the number of descendants of node k_{i+1} in the final tree in case the buffer size is 1, and δ_i^c is an upper bound for the same if the buffer has size c. Hence, the average of all values $\delta_i^c - \delta_i^1$ upper bounds the difference in the average number of descendants of all nodes between the two final trees, which in turn is equal to the difference of the average node depth between the two final trees.

We bound the sum above separately for $0 \leq i \leq c-1$, $c \leq i \leq b$, and $b < i \leq n-1$.

From the trivial bounds $\delta_i^c \leq n+1$ and $\delta_i^1 \geq 0$, we get

$$\sum_{0 \leq i \leq c-1} \mathbf{E}[\delta_i^c - \delta_i^1] \leq \sum_{0 \leq i \leq c-1} \mathbf{E}[\delta_i^c] \leq c(n+1). \tag{14}$$

By Lemma 8, we have $\mathbf{E}[\delta_i^c] = O(c + \log n)$ for $i \in \{\lceil n/2 \rceil, \ldots, n-1\}$, thus

$$\sum_{b < i \leq n-1} \mathbf{E}[\delta_i^c - \delta_i^1] = O\big((n-b) \cdot (c + \log n)\big) = O\left(cn\right). \tag{15}$$

To bound the remaining sum of $\mathbf{E}[\delta_i^c - \delta_i^1]$, for $c-1 \leq i \leq b$, first recall the following basic facts for any real $z \geq 0$ and integers $1 \leq a \leq \ell$:

$$\sum_{i=a}^{\infty} \frac{1}{i^2} \leq \frac{1}{a} + \frac{1}{a^2}; \quad \sum_{i=a}^{\ell} \frac{1}{i} \leq \frac{1}{a} + \ln(\ell/a); \quad \sum_{i=1}^{\infty} \frac{(\log i)^2}{i^2} \text{ converges; and } \quad \ln(1+z) \leq z. \tag{16}$$

By Lemma 5,

$$\begin{aligned}
\sum_{c \leq i \leq b} \mathbf{E}\left[\delta_i^c - \delta_i^1\right] &= O\left(\sum_{c \leq i \leq b} \frac{n^2c^2 + n^2 c \log^2 i}{i^2(n-i)}\right) \\
&= O\left(\sum_{c \leq i \leq n/2} \frac{n^2c^2 + n^2 c \log^2 i}{i^2(n-i)} + \sum_{n/2 < i \leq b} \frac{n^2c^2 + n^2 c \log^2 i}{i^2(n-i)}\right) \\
&= O\left(\sum_{c \leq i \leq n/2} \frac{nc^2 + nc \log^2 i}{i^2}\right) + O\left(\sum_{n/2 < i \leq b} \frac{c^2 + c\log^2 n}{n-i}\right) \\
&= O\left(nc \cdot \sum_{i=c}^{\infty} \left(\frac{c}{i^2} + \frac{\log^2 i}{i^2}\right)\right) + O\left((c^2 + c\log^2 n) \sum_{n-b \leq j < n/2} \frac{1}{j}\right) \\
&\overset{(16)}{=} O\left(nc\right) + O\left((c^2 + c\log^2 n) \cdot \left(\frac{1}{n-b} + \ln \frac{n/2}{n-b}\right)\right) \\
&= O(nc) + O\left((c^2 + c\log^2 n) \cdot \left(1 + \ln \frac{c + \log n}{c}\right)\right) \\
&\overset{(16)}{=} O(nc) + O\left((c^2 + c\log^2 n) \cdot \left(1 + \frac{\log n}{c}\right)\right) \\
&= O(nc) + O\left(c^2 + c\log^2 n + c \log n + \log^3 n\right) \\
&= O(nc).
\end{aligned} \tag{17}$$

Combining (13), (14), (15), and (17), we obtain $\Delta(n,c) - \Delta(n,1) = O(c)$. This completes the proof of Theorem 1. ◀

Acknowledgment

The authors are grateful to Eric Ruppert for his comments on a draft of the proof presented here. Eric also pointed out that our earlier, much simpler proof attempt was indeed too simple to be true.

This research was undertaken, in part, thanks to funding from the ANR Project NDFusion (ANR-16-TERC-0007), the ANR Project PAMELA (ANR-16-CE23-0016-01), the Canada Research Chairs program, and the Discovery Grants program of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1 James Aspnes and Eric Ruppert. Depth of a random binary search tree with concurrent insertions. In *Proc. 30th International Symposium on Distributed Computing (DISC)*, pages 371–384, 2016.

2 Andrew Donald Booth and Andrew J.T. Colin. On the efficiency of a new method of dictionary construction. *Information and Control*, 3(4):327–334, 1960.

3 Luc Devroye. A note on the height of binary search trees. *Journal of the ACM*, 33(3):489–498, 1986.

4 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.

5 Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, 11(1):286–295, 1983.

6 Hosam Mahmoud Mahmoud. *Evolution of Random Search Trees*. Wiley-Interscience, 1992.

7 Bodo Manthey and Rüdiger Reischuk. Smoothed analysis of binary search trees. *Theoretical Computer Science*, 378(3):292–315, 2007.

8 Bruce Reed. The height of a random binary search tree. *Journal of the ACM*, 50(3):306–332, 2003.

9 John Michael Robson. The height of binary search trees. *Australian Computer Journal*, 11(4):151–153, 1979.

10 Peter F. Windley. Trees, forests and rearranging. *The Computer Journal*, 3(2):84–88, 1960.

A Appendix

In the following we provide a basic claim to facilitate our analysis. We believe that this simple observation should be known, but we could not find a reference for it.

▶ **Claim 9.** *Let S be a set of $m \in \{1,\dots,n\}$ elements chosen at random from $\{1,\dots,n\}$ without replacement. Let $s_1,\dots,s_m$ be the elements of S in sorted order, and let $s_0 = 0$, $s_{m+1} = n+1$. Then for any $d \in \{1,\dots,m+1\}$ and $j \in \{0,\dots,m-d+1\}$, the distribution of $s_{j+d} - s_j$ is identical to the distribution of $s_d - s_0$.*

Proof. Fix $d \in \{1,\dots,m+1\}$. Choose a set S' of $m+1$ elements in $\{0,\dots,n\}$ at random without replacement, and let $s'_0,\dots,s'_m$ be the chosen elements in sorted order. Considering the elements as chosen from the $\bmod(n+1)$ ring $\{0,\dots,n\}$, it is obvious by symmetry that all random variables $Z_j = (s'_{(j+d) \bmod (m+1)} - s'_j) \bmod (n+1)$, $j \in \{0,\dots,m\}$, are identically distributed. In particular, each variable Z_j, $j \in \{0,\dots,m\}$ has the same distribution as Z_0.

Then letting $s_i = s'_i - s'_0$, we have that $s_1,\dots,s_m$ have the same distribution as if they were chosen from $\{1,\dots,n\}$ at random without replacement and then sorted. Moreover, for

$j \in \{0, \ldots, m-d\}$ we have $s_j \leq s_{j+d} \leq n$, so

$$s_{j+d} - s_j = (s_{j+d} - s_j) \bmod (n+1) = (s'_{j+d} - s'_j) \bmod (n+1)$$
$$= (s'_{(j+d) \bmod (m+1)} - s'_j) \bmod (n+1) = Z_j.$$

And since $s_0 \equiv s_{m+1} \pmod{n+1}$, we obtain for $j = m-d+1$ using the same calculation as above

$$s_{j+d} - s_j = (s_0 - s_j) \bmod (n+1) = (s'_{(j+d) \bmod (m+1)} - s'_j) \bmod (n+1) = Z_j.$$

Since all random variables $Z_j = s_{j+d} - s_j$, for $j \in \{0, \ldots, m-d+1\}$, have the same distribution as Z_0, the claim follows. ◀