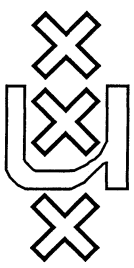
Institute for Language, Logic and Information

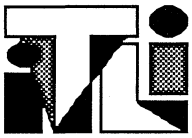
GENERAL LOWER BOUNDS FOR THE PARTITIONING OF RANGE TREES

Michiel H.M. Smid

ITLI Prepublication Series
for Compuation and Complexity Theory CT-88-02



University of Amsterdam



Institute for Language, Logic and Information
Instituut voor Taal, Logica en Informatie

GENERAL LOWER BOUNDS FOR THE PARTITIONING OF RANGE TREES

Michiel H.M. Smid
Centre for Mathematics and Computer Science
Amsterdam

Received April 1988

Correspondence to:

Faculteit der Wiskunde en Informatica
(Department of Mathematics and Computer Science)
Roetersstraat 15
1018WB Amsterdam

or

Faculteit der Wijsbegeerte
(Department of Philosophy)
Grimburgwal 10
1012GA Amsterdam

General Lower Bounds for the Partitioning of Range Trees

Michiel H. M. Smid*

April 1988

Abstract

When storing and maintaining a data structure in secondary memory it is important to partition it into parts such that queries and updates pass through a small number of parts. In this way the number of disk accesses and the amount of data transport can be kept low. This paper presents general lower bounds for partitioning range trees, which improve previous results. It is shown e.g. that if a d-dimensional range tree for a set of n points is partitioned into parts such that each update passes through at most $g(n)$ parts, there must be a part of size at least $\frac{2}{d!}(\frac{1}{g(n)})^d n^{1/g(n)}(\log n)^{d-1}$.

1 Introduction

This paper continues the study of the maintenance of range trees in secondary memory (see Overmars et al. [3], Smid et al. [4]). The range tree is an efficient data structure solving the orthogonal range searching problem, a problem having many applications in e.g. computer graphics and database design.

Definition 1 *Let S be a set of points in d-dimensional space, and let $([x_1 : y_1], [x_2 : y_2], \ldots, [x_d : y_d])$ be some hyperrectangle. The* orthogonal range searching problem *asks for all points $p = (p_1, p_2, \ldots, p_d)$ in S, such that $x_1 \leq p_1 \leq y_1, x_2 \leq p_2 \leq y_2, \ldots, x_d \leq p_d \leq y_d$.*

We assume in this paper that the data structure solving the range searching problem is too large to be stored entirely in main memory, a situation that very often occurs in databases. Therefore the structure has to be stored in secondary memory, and, in order to answer queries and to perform updates, parts of the data

*Bureau SION, Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. This work was supported by the Netherlands Organization for Scientific Research (NWO).

structure have to be transported from secondary memory to core, and vice versa. Hence, it is necessary to partition the data structure into parts, such that queries and updates pass through only a small number of parts (hence only a small amount of data has to be transported). In this paper we shall only consider updates.

Definition 2 *A partition of a dynamic data structure, representing a set of n points, is called an* $(f(n), g(n))$-partition, *if*

1. *Each part has size at most $f(n)$.*
2. *Each update passes through at most $g(n)$ parts.*

It follows from this definition that the structure can be stored in secondary memory, such that each update requires at most $g(n)$ disk accesses, and at most $g(n)$ parts, each of size $\leq f(n)$, have to be transported. In [3], several partition schemes are given for range trees, obtaining different trade-offs between $f(n)$ and $g(n)$. In [4], lower bounds for partitions are given. As an example, it is shown there that for an $(f(n), k)$-partition of a d-dimensional range tree, where k is a positive integer such that $n \geq 2^k$, we have $f(n) \geq \frac{2}{d!}\frac{1}{2^{k-1}}(\frac{1}{k})^{d-1}n^{1/k}(\log n)^{d-1}$. Clearly, this result remains valid if k is a function of n. So let $g(n)$ be an integer function such that $1 \leq g(n) \leq \log n$. Then for an $(f(n), g(n))$-partition of a d-dimensional range tree, we have $f(n) \geq \frac{2}{d!}\frac{1}{2^{g(n)-1}}(\frac{1}{g(n)})^{d-1}n^{1/g(n)}(\log n)^{d-1}$. In the present paper we improve the lower bounds in [4]. E.g. we show that for an $(f(n), g(n))$-partition of a d-dimensional range tree, $f(n) \geq \frac{2}{d!}(\frac{1}{g(n)})^{d}n^{1/g(n)}(\log n)^{d-1}$. The ideas to prove these general lower bounds are similar to those in [4] (indeed we use several lemmas from that paper).

Range trees are defined in Section 2 and we give an algorithm to insert and delete points. The notion of range trees we use in this paper is the same as in [4]. We do not require trees to be balanced. As the lower bounds we prove apply to any tree (rather than to some trees) in the class of range trees, the bounds also hold under the usual definition, i.e. for range trees having BB[α]-trees as underlying structure. (Moreover, the bounds are more general and would also apply if we for example would use AVL-trees as underlying structure.) In Section 3, we prove the lower bound for binary search trees or, equivalently, for one-dimensional range trees. In Section 4, we give a lower bound for so-called restricted partitions of multi-dimensional range trees, which turns out to be tight (except for constant factors). Then, in Section 5 the general lower bound for multi-dimensional range trees will be presented. We conclude the paper with some final remarks in Section 6.

The following notations will be used in the rest of the paper. First, logarithms, and powers of logarithms, are given in the usual way, i.e., we write $\log n$, $(\log n)^2$, etc. (in this paper, $\log n$ denotes the logarithm to the base 2). Furthermore, the k-th iterated logarithm is written as follows. If $k = 1$, then $(\log)^1 n = \log n$. If

$k > 1$, then $(\log)^k n = \log\left((\log)^{k-1} n\right)$. The function $\log^* n$ is defined by $\log^* n = \min\{k \geq 1 | (\log)^k n \leq 1\}$.

If we have a partition of a range tree into parts, then the *size* of a part is defined as the number of nodes it contains.

2 Range trees

In this section we define range trees, and we give an algorithm to insert and delete points.

A *binary search tree*, or just binary tree, is a rooted tree, in which each node has zero or two sons. In this paper, binary trees are used as leaf search trees. That is, if we use a binary tree to represent a set S of real numbers, we store the elements of S in sorted order in the leaves of the tree. Internal nodes of the tree contain information to guide searches. The binary tree is the underlying structure for range trees (see Bentley [1], Lueker [2], Willard and Lueker [5]).

Definition 3 *Let S be a set of points in d-dimensional space. A* d-dimensional range tree *T, representing the set S, is defined as follows.*

1. *If $d = 1$, then T is a binary tree, containing the points of S in sorted order in its leaves.*
2. *If $d > 1$, then T consists of a binary tree, called the* main tree, *which contains the points of S in its leaves, ordered according to their first coordinates. Also, each internal node v of this main tree contains an* associated structure, *which is a $(d-1)$-dimensional range tree, representing those points of S which are in the subtree rooted at v, taking only the second to d-th coordinate into account.*

Note that in our definition, we do not impose any balance condition on the binary trees. All results in this paper apply as well for balanced as for unbalanced range trees.

Let T be a d-dimensional range tree, representing the set S, and let v be a node of T (v is a node of the main tree, or of an associated structure, or of an associated structure of an associated structure, etc.). Let S_v be the set of those points of S, which are in the subtree of v. Then node v is said to *represent* the set S_v.

For an algorithm, solving the orthogonal range searching problem using range trees, we refer the reader to [1,2,5]. The update algorithm for these trees is as follows. Suppose we want to insert or delete point p in the range tree. Then we search with p in the main tree to locate its position among the leaves, and we insert or delete p in all associated structures we encounter on our search path. If these associated structures are one-dimensional range trees, we apply the usual

insertion/deletion algorithm for binary trees (possibly with rebalancing); otherwise we use the same procedure recursively. Finally, we insert or delete p among the leaves of the main tree.

In this paper, we consider two types of partitions of range trees. A partition of a d-dimensional range tree, where $d > 1$, is called *restricted* if only the main tree is partitioned, whereas associated structures are never subdivided. In a restricted partition, a node of the main tree and its associated structure are contained in the same part. The second type of partitions we consider, are those in which also associated structures are divided into parts. In the rest of this paper, the term partition (without the adjective restricted) indicates a partition of this second type.

3 A lower bound for binary search trees

Let $g(n)$ be an integer function such that $1 \leq g(n) \leq \log n$. In [4] it is shown that if a binary leaf search tree is partitioned into parts, such that each path from the root to a leaf passes through at most $g(n)$ parts, there must be a part of size at least $\frac{1}{2^{g(n)-1}} n^{1/g(n)}$. In terms of Definition 2, for an $(f(n), g(n))$-partition of a binary tree, we have $f(n) \geq \frac{1}{2^{g(n)-1}} n^{1/g(n)}$. (This result is stated in [4] only for constant functions $g(n)$. It is clear, however, that the bound remains valid for non-constant functions $g(n)$.) In this section we shall improve this lower bound. We first give the following lemma.

Lemma 1 *For any non-negative integer k, we have*

$$\left(\frac{k}{e}\right)^k \leq k! \leq k^k,$$

where e is the basis of the natural logarithm.

Proof. The proof follows from a straightforward calculation: $\ln k! = \sum_{i=2}^{k} \ln i \geq \sum_{i=2}^{k} \int_{i-1}^{i} \ln x \, dx = \int_{1}^{k} \ln x \, dx = k \ln k - k + 1 \geq k \ln k - k = k \ln \frac{k}{e}$. Hence $k! \geq (\frac{k}{e})^k$. The other inequality is trivial. □

We need the following lemma from [4]. (We remind the reader to our notion of the set of points *represented* by a node, see Section 2.)

Lemma 2 *Let T be a binary tree with n leaves. Let $m \geq 1$ be a real number. Then the number of nodes in T, representing at least m leaves, is at least $\frac{n}{m} - 1$.*

If we take for T a balanced binary tree, we see that this bound is tight (except for constant factors). Lemma 2 enables us to prove our first lower bound. Clearly, for a binary tree from a class of $O(\log n)$-maintainable trees— e.g. an AVL-tree—there exists a $(1, O(\log n))$-partition: put each node in a separate part. Therefore, it is sufficient to consider only $(f(n), g(n))$-partitions of binary trees with $g(n) \leq \log n$.

Theorem 1 *Let $g(n)$ be an integer function, such that $1 \leq g(n) \leq \log n$. Let T be a binary leaf search tree representing n points. Suppose the tree T is partitioned into parts, such that each update passes through at most $g(n)$ parts. Then there is a part of size at least*

$$\left(\frac{n}{g(n)!}\right)^{1/g(n)} - 1 \quad \sim \quad \frac{e}{g(n)} n^{1/g(n)} - 1.$$

Proof. Let $m_i = \frac{n}{i!}(\frac{g(n)!}{n})^{i/g(n)}$ for $0 \leq i \leq g(n) - 1$. We first show that the m_i's are at least 1. Clearly, $m_0 = n \geq 1$.

Let $0 < i \leq \frac{1}{2}g(n)$. Then, by using Lemma 1,

$$m_i = n^{1-i/g(n)} \frac{(g(n)!)^{i/g(n)}}{i!} \geq n^{1-i/g(n)} \left(\frac{g(n)}{e\,i}\right)^i \geq \sqrt{n} \left(\frac{2}{e}\right)^i.$$

Since $i \leq \frac{1}{2}g(n) \leq \frac{1}{2}\log n$, it follows that $\sqrt{n} \geq 2^i$, and hence $m_i \geq (\frac{4}{e})^i \geq 1$.

Let $\frac{1}{2}g(n) < i \leq g(n) - 2$. Again using Lemma 1, we get

$$\frac{m_i}{m_{i+1}} = (i+1)\left(\frac{n}{g(n)!}\right)^{1/g(n)} \geq \frac{i+1}{g(n)} n^{1/g(n)} \geq \frac{1}{2} n^{1/g(n)} \geq 1.$$

Also

$$m_{g(n)-1} = g(n)\left(\frac{n}{g(n)!}\right)^{1/g(n)} \geq g(n)\frac{n^{1/g(n)}}{g(n)} = n^{1/g(n)} \geq 1.$$

Hence for $\frac{1}{2}g(n) < i \leq g(n) - 2$, we have

$$m_i \geq m_{i+1} \geq m_{i+2} \geq \ldots \geq m_{g(n)-1} \geq 1.$$

Let P_i be the following property:

> v_i is a node in T, representing at least m_i points. $\Pi_0, \Pi_1, \ldots, \Pi_i$ is a sequence of $i+1$ different parts of the partition. Each update in T, passing through v_i, passes through $\Pi_0, \Pi_1, \ldots, \Pi_i$.

Now execute the following algorithm:

> $v_0 :=$ root of T;
> $\Pi_0 :=$ part of the partition containing v_0;
> $i := 0$;
> QED := **false**;
> { property P_i holds }
> **while** $i \neq g(n) - 1 \wedge$ **not** QED
> **do** { property P_i holds }
> Let S be the set of all nodes below v_i representing at least m_{i+1} points;

if $S \subset \bigcup_{j=0}^{i} \Pi_j$
then QED := **true**
else v_{i+1} := a node in $S \setminus \bigcup_{j=0}^{i} \Pi_j$;
Π_{i+1} := part of the partition containing v_{i+1};
$i := i+1$
{ property P_i holds }
fi
od.

First note that this algorithm terminates. Suppose that after the algorithm is completed, QED has the value **true**. Then there is an i, $0 \leq i \leq g(n)-2$, such that P_i holds, and all nodes below v_i representing at least m_{i+1} points, are contained in $\bigcup_{j=0}^{i} \Pi_j$. Since by Lemma 2 — which may be applied since $m_{i+1} \geq 1$ — there are at least $\frac{m_i}{m_{i+1}} - 1$ such nodes, it follows that $|\bigcup_{j=0}^{i} \Pi_j| = \sum_{j=0}^{i} |\Pi_j| \geq \frac{m_i}{m_{i+1}} - 1$. Hence there is a part in the partition of size at least

$$\frac{1}{i+1} \sum_{j=0}^{i} |\Pi_j| \geq \frac{1}{i+1} \left(\frac{m_i}{m_{i+1}} - 1 \right) \geq \left(\frac{n}{g(n)!} \right)^{1/g(n)} - 1.$$

Otherwise, QED has the value **false**, and since the algorithm terminates, we must have $i = g(n)-1$. Also property $P_{g(n)-1}$ holds. So we have a node $v = v_{g(n)-1}$ representing at least $m_{g(n)-1}$ points, and we have a sequence $\Pi_0, \Pi_1, \ldots, \Pi_{g(n)-1}$ of $g(n)$ different parts of the partition, such that each update through node v passes through these $g(n)$ parts. Since each update in the tree passes through at most $g(n)$ parts, it follows that all nodes in the subtree of v are contained in $\bigcup_{j=0}^{g(n)-1} \Pi_j$. Hence $|\bigcup_{j=0}^{g(n)-1} \Pi_j| = \sum_{j=0}^{g(n)-1} |\Pi_j| \geq m_{g(n)-1}$. It follows that there is a part of size at least

$$\frac{1}{g(n)} \sum_{j=0}^{g(n)-1} |\Pi_j| \geq \frac{1}{g(n)} m_{g(n)-1} \geq \left(\frac{n}{g(n)!} \right)^{1/g(n)} - 1$$

(and we can set QED := **true**).

The approximation $(\frac{n}{g(n)!})^{1/g(n)} \sim \frac{e}{g(n)} n^{1/g(n)}$ follows from Stirling's formula. This proves the theorem. □

Consider again the proof of Theorem 1. We started with a sequence $m_0, \ldots, m_{k-1}$ of real numbers, such that $m_0 = n$, and $m_i \geq 1$ for all i (we write $k = g(n)$). Then we showed that the partition contains a part of size at least $\min\{\frac{1}{i+1}(\frac{m_i}{m_{i+1}} - 1), \frac{1}{k} m_{k-1}\}$ for some $0 \leq i \leq k-2$. Since i can take any value between 0 and $k-2$, it follows that the partition contains a part of size at least

$$\min\{\frac{m_0}{m_1} - 1, \frac{1}{2}(\frac{m_1}{m_2} - 1), \frac{1}{3}(\frac{m_2}{m_3} - 1), \ldots, \frac{1}{k-1}(\frac{m_{k-2}}{m_{k-1}} - 1), \frac{1}{k} m_{k-1}\}. \quad (1)$$

Clearly the proof remains valid for any such sequence $m_0, m_1, \ldots, m_{k-1}$. (Note that the condition $m_i \geq 1$ is important, since otherwise Lemma 2 may not be

applied. Also, the condition $m_0 = n$ is necessary to make property P_0 hold after the initialization of the algorithm.) Therefore it might be possible that for some other choice of the numbers m_i a better lower bound follows. We shall show that this is not the case. That is, the value of Equation (1) is at most $(\frac{n}{k!})^{1/k}$, for any sequence of real numbers $m_0 = n, m_1 \geq 1, m_2 \geq 1, \ldots, m_{k-1} \geq 1$. Take such a sequence $m_0, \ldots, m_{k-1}$. There are two possibilities.

There is an i, $0 \leq i \leq k-2$, such that $m_i \leq \frac{n}{i!}(\frac{k!}{n})^{i/k}$ and $m_{i+1} > \frac{n}{(i+1)!}(\frac{k!}{n})^{(i+1)/k}$. Then

$$\frac{1}{i+1}\frac{m_i}{m_{i+1}} \leq \left(\frac{n}{k!}\right)^{1/k},$$

and hence the value of Equation (1) is at most $(\frac{n}{k!})^{1/k}$.

Otherwise, for each i, $0 \leq i \leq k-2$, we have that if $m_i \leq \frac{n}{i!}(\frac{k!}{n})^{i/k}$, then $m_{i+1} \leq \frac{n}{(i+1)!}(\frac{k!}{n})^{(i+1)/k}$. Since $m_0 = n \leq \frac{n}{0!}(\frac{k!}{n})^{0/k}$, it follows that $m_1 \leq \frac{n}{1!}(\frac{k!}{n})^{1/k}$, and hence $m_2 \leq \frac{n}{2!}(\frac{k!}{n})^{2/k}, \ldots, m_{k-1} \leq \frac{n}{(k-1)!}(\frac{k!}{n})^{(k-1)/k}$. We conclude that

$$\frac{1}{k}m_{k-1} \leq \frac{1}{k}\frac{n}{(k-1)!}\left(\frac{k!}{n}\right)^{(k-1)/k} = \left(\frac{n}{k!}\right)^{1/k},$$

hence the value of Equation (1) is bounded above by $(\frac{n}{k!})^{1/k}$.

4 A lower bound for restricted partitions

In this section we prove a tight lower bound for restricted partitions of range trees. We remind the reader to the notion of a restricted partition (see Section 2), and to our notation for iterated logarithms (see Section 1). The lower bound of this section improves the following one (see [4]): Let $g(n)$ be an integer function such that $1 \leq g(n) \leq \log^* n$. For a restricted $(f(n), g(n))$-partition of a d-dimensional range tree ($d \geq 2$), where $(\log)^{g(n)} n \geq 16$, we have $f(n) \geq (\frac{1}{2})^{(d-2)(g(n)-1)}\frac{1}{d!}\, n\, (\log n)^{d-2} (\log)^{g(n)} n$. (Again this lower bound is valid for non-constant functions $g(n)$, although it is stated in [4] only for constant functions.) First we recall a lemma from [4].

Lemma 3 *Consider a d-dimensional range tree ($d \geq 2$), representing n points. Let $m \geq 1$ be a real number. For each node v of the main tree, the weight $wt(v)$ of v is defined as the total number of leaves in the associated structure of v (here we count the leaves in the main tree of the associated structure, in associated structures of the associated structure, etc.). Then*

$$\sum_{v:v \text{ represents } \geq m \text{ points}} wt(v) \;\geq\; \frac{2}{d!}\, n\, (\log n)^{d-2} \log \frac{n}{m}, \quad \text{if} \quad n \geq \lfloor m \rfloor + 1,$$

where the summation runs over all nodes in the main tree, representing at least m points.

Note that the bound in Lemma 3 is tight (except for constant factors): equality is obtained if all binary trees involved are balanced.

Corollary 1 *A d-dimensional range tree ($d \geq 2$), representing n points, has size at least $\frac{2}{d!} n (\log n)^{d-1}$.*

Proof. This follows from Lemma 3 by taking $m = 1$. □

It follows from Corollary 1 that in a restricted partition of a d-dimensional range tree, there is a part—the part containing the associated structure of the root of the main tree—of size $\Omega(n(\log n)^{d-2})$. Hence in a restricted $(f(n), g(n))$-partition, we always have $f(n) = \Omega(n(\log n)^{d-2})$. In [3] it is shown that there exist d-dimensional range trees that can be maintained efficiently (that is, they have the same performances as balanced range trees, see Willard and Lueker [5]), that can be partitioned into a restricted $(O(n(\log n)^{d-2}), \log^* n + O(1))$-partition. Therefore it is sufficient to consider only restricted $(f(n), g(n))$-partitions with $g(n) \leq \log^* n + O(1)$.

Theorem 2 *Let $g(n)$ be an integer function, such that $1 \leq g(n) \leq \log^* n$. Let T be a d-dimensional range tree ($d \geq 2$), representing n points, where $(\log)^{g(n)} n \geq 8$ and $\frac{1}{2} \log n \geq (d-1) \log \log n$. Suppose T is partitioned, in the restricted sense, into parts such that each update passes through at most $g(n)$ parts. Then there is a part of size at least*

$$\frac{1}{2} \frac{1}{d!} n (\log n)^{d-2} (\log)^{g(n)} n.$$

Proof. Let $m_i = (i+1)\, n \frac{(\log)^{g(n)} n}{(\log)^{g(n)-i} n}$ for $0 \leq i \leq g(n) - 1$. Note that the iterated logarithms exist since $1 \leq g(n) \leq \log^* n$. Then

$$m_{g(n)-1} = g(n)\, n \frac{(\log)^{g(n)} n}{\log n} \geq \frac{n}{\log n} \geq 2,$$

and

$$\frac{m_i}{m_{i+1}} = \frac{i+1}{i+2} \frac{(\log)^{g(n)-i-1} n}{(\log)^{g(n)-i} n} \geq \frac{1}{2} \frac{(\log)^{g(n)-i-1} n}{(\log)^{g(n)-i} n} \geq 2,$$

since for $N = (\log)^{g(n)-i} n$, we have $\frac{1}{2} \frac{2^N}{N} \geq 2$. (Note that $N \geq (\log)^{g(n)} n \geq 8$.) It follows that all m_i are at least 1, and that $m_i \geq 2m_{i+1} \geq m_{i+1} + 1 \geq \lfloor m_{i+1} \rfloor + 1$.

Let P_i be the following property:

> v_i is a node in the main tree of T, representing at least m_i points.
> $\Pi_0, \Pi_1, \ldots, \Pi_i$ is a sequence of $i+1$ different parts of the partition.
> Each update in T, passing through v_i, passes through $\Pi_0, \Pi_1, \ldots, \Pi_i$.

Now execute the following algorithm:

v_0 := root of the main tree;
Π_0 := part of the partition containing v_0;
$i := 0$;
QED := **false**;
{ property P_i holds }
while $i \neq g(n) - 1 \wedge$ **not** QED
do { property P_i holds }
  Let S be the set of all nodes in the main tree below v_i, representing at least m_{i+1} points;
  if $S \subset \bigcup_{j=0}^{i} \Pi_j$
  then QED := **true**
  else v_{i+1} := a node in $S \setminus \bigcup_{j=0}^{i} \Pi_j$;
    Π_{i+1} := part of the partition containing v_{i+1};
    $i := i + 1$
    { property P_i holds }
  fi
od.

Suppose that after the algorithm is completed QED has the value **true**. Then there is an i, $0 \leq i \leq g(n) - 2$, such that P_i holds, and all nodes in the main tree below v_i representing at least m_{i+1} points, are contained in $\bigcup_{j=0}^{i} \Pi_j$. By Lemma 3, all these nodes together with their associated structures — and hence $\bigcup_{j=0}^{i} \Pi_j$ — have size at least

$$\frac{2}{d!}\, m_i\, (\log m_i)^{d-2}\, \log \frac{m_i}{m_{i+1}}.$$

(Since $m_{i+1} \geq 1$ and $m_i \geq \lfloor m_{i+1} \rfloor + 1$, Lemma 3 may be applied. Note that we apply Lemma 3 to the tree having v_i as its root, which is a d-dimensional range tree, representing at least m_i points.) We have

$$\begin{aligned}
\log m_i &= \log \left((i+1)\, n\, \frac{(\log)^{g(n)} n}{(\log)^{g(n)-i} n} \right) \\
&\geq \log \left(\frac{n}{(\log)^{g(n)-i} n} \right) \\
&\geq \log \left(\frac{n}{\log n} \right) \\
&= \log n - \log \log n.
\end{aligned}$$

Then by using the inequality $(a-b)^{d-2} \geq a^{d-2} - (d-2)a^{d-3}b$ for real numbers $a \geq b \geq 0$—which can easily be proved by induction on d—we get

$$\begin{aligned}
(\log m_i)^{d-2} &\geq (\log n - \log \log n)^{d-2} \\
&\geq (\log n)^{d-2} - (d-2)(\log n)^{d-3} \log \log n \\
&\geq \frac{1}{2} (\log n)^{d-2},
\end{aligned}$$

since we assumed that $\frac{1}{2}\log n \geq (d-1)\log\log n \geq (d-2)\log\log n$. Furthermore,

$$\begin{aligned}
\log\frac{m_i}{m_{i+1}} &= \log\left(\frac{i+1}{i+2}\,\frac{(\log)^{g(n)-i-1}n}{(\log)^{g(n)-i}n}\right)\\
&\geq \log\left(\frac{1}{2}\,\frac{(\log)^{g(n)-i-1}n}{(\log)^{g(n)-i}n}\right)\\
&= (\log)^{g(n)-i}n - (\log)^{g(n)-i+1}n - 1\\
&\geq \frac{1}{2}(\log)^{g(n)-i}n,
\end{aligned}$$

since for $N = (\log)^{g(n)-i}n \geq (\log)^{g(n)}n \geq 8$, we have $N - \log N - 1 \geq N/2$.

It follows that there is a part in our partition of size at least

$$\begin{aligned}
\frac{1}{i+1}\,|\bigcup_{j=0}^{i}\Pi_j| &\geq \frac{1}{i+1}\,\frac{2}{d!}\,m_i\,(\log m_i)^{d-2}\,\log\frac{m_i}{m_{i+1}}\\
&\geq \frac{1}{i+1}\,\frac{2}{d!}\,m_i\,\frac{1}{2}\,(\log n)^{d-2}\,\frac{1}{2}(\log)^{g(n)-i}n\\
&= \frac{1}{2}\,\frac{1}{d!}\,n\,(\log n)^{d-2}\,(\log)^{g(n)}n.
\end{aligned}$$

Otherwise, QED has the value **false**, and since the algorithm terminates, we must have $i = g(n)-1$. Also property $P_{g(n)-1}$ holds. So we have a node $v = v_{g(n)-1}$ in the main tree, representing at least $m_{g(n)-1}$ points, and we have a sequence $\Pi_0, \Pi_1, \ldots, \Pi_{g(n)-1}$ of $g(n)$ different parts of the partition, such that each update through node v passes through these $g(n)$ parts. Since each update in the tree passes through at most $g(n)$ parts, it follows that all nodes in the subtree of v, together with their associated structures, are contained in $\bigcup_{j=0}^{g(n)-1}\Pi_j$. Since this subtree is a d-dimensional range tree, representing at least $m_{g(n)-1}$ points, it follows that (apply Lemma 3 with $m = 1$, which is allowed since $m_{g(n)-1} \geq 2$)

$$|\bigcup_{j=0}^{g(n)-1}\Pi_j| \geq \frac{2}{d!}\,m_{g(n)-1}\,(\log m_{g(n)-1})^{d-1}.$$

In the same way as above, we have

$$\log m_{g(n)-1} \geq \log n - \log\log n,$$

and hence

$$\begin{aligned}
(\log m_{g(n)-1})^{d-1} &\geq (\log n - \log\log n)^{d-1}\\
&\geq (\log n)^{d-1} - (d-1)(\log n)^{d-2}\log\log n\\
&\geq \frac{1}{2}(\log n)^{d-1},
\end{aligned}$$

since $\frac{1}{2}\log n \geq (d-1)\log\log n$. Hence there is a part of size at least

$$\frac{1}{g(n)}\,|\bigcup_{j=0}^{g(n)-1}\Pi_j| \geq \frac{1}{g(n)}\,\frac{2}{d!}\,m_{g(n)-1}\,\frac{1}{2}(\log n)^{d-1} \geq \frac{1}{2}\,\frac{1}{d!}\,n\,(\log n)^{d-2}\,(\log)^{g(n)}n$$

(and we can set QED := **true**). This proves the theorem. □

Remark. The lower bound in Theorem 2 is tight. That is, there exist d-dimensional range trees, having asymptotically the same complexity as balanced range trees (see Willard and Lueker [5]), that can be partitioned into a restricted $(O(n(\log n)^{d-2}(\log)^{g(n)}n), g(n))$-partition. For details, see [3]. Note that this result is given in [3] only for constant functions $g(n) = k$. Since the constants in the complexity bounds are independent of k, the result remains valid for arbitrary functions $g(n)$ with $1 \leq g(n) \leq \log^* n$.

5 A lower bound for general partitions

In this section the lower bound for general partitions is proved. Just as in the previous sections, we first recall the lower bound from [4]. Let $g(n)$ be an integer function such that $1 \leq g(n) \leq \log n$. For an $(f(n), g(n))$-partition of a d-dimensional range tree, we have $f(n) \geq \frac{2}{d!}\,\frac{1}{2^{g(n)-1}}(\frac{1}{g(n)})^{d-1}\,n^{1/g(n)}(\log n)^{d-1}$. The proof of the following lemma also comes from [4].

Lemma 4 *Consider a d-dimensional range tree ($d \geq 2$), representing at least n points. Let S be a subset of these points, of cardinality n. Let $m \geq 1$ be a real number. Then the total number of nodes in the range tree (in the main tree, or in an associated structure, or in an associated structure of an associated structure, etc.), representing at least m points of S, is at least $\frac{2}{d!}\,\frac{n}{m}(\log\frac{n}{m})^{d-1}$, if $n \geq \lfloor m \rfloor + 1$.*

As in the previous sections—by taking all binary trees balanced— we see that this bound is tight (except for constant factors).

Theorem 3 *Let $g(n)$ be an integer function, such that $1 \leq g(n) \leq \log n$. Let T be a d-dimensional range tree ($d \geq 2$), representing n points. Suppose this range tree is partitioned into parts, such that each update passes through at most $g(n)$ parts. Then there is a part of size at least*

$$\frac{2}{d!}\left(\frac{1}{g(n)}\right)^d n^{1/g(n)}\,(\log n)^{d-1}.$$

Proof. Let $m_i = n^{1-i/g(n)}$ for $i \geq 0$. Let P_i be the following property:

> S_i is a subset of the set of points represented by T, of cardinality at least m_i. $\Pi_0, \Pi_1, \ldots, \Pi_i$ is a sequence of $i+1$ different parts of the partition. Each update in T, passing through a point of S_i, passes through $\Pi_0, \Pi_1, \ldots, \Pi_i$ (note that there exist such updates).

Now execute the following algorithm:

$i := 0$;
$S_0 :=$ set of all points represented by the range tree;
$\Pi_0 :=$ part of the partition containing the root of the main tree;
QED := **false**;
{ property P_i holds }
while $i \neq g(n) - 1 \wedge$ **not** QED
do { property P_i holds }
  Let V be the set of all nodes representing at least m_{i+1} points of S_i;
  if $V \subset \bigcup_{j=0}^{i} \Pi_j$
  then QED := **true**
  else $w :=$ a node in $V \setminus \bigcup_{j=0}^{i} \Pi_j$;
    $S_{i+1} :=$ the set of all points in S_i, represented by w;
    $\Pi_{i+1} :=$ part of the partition containing w;
    $i := i + 1$
    { property P_i holds }
  fi
od.

Suppose that after the algorithm is completed QED has the value **true**. Then there is an i, $0 \leq i \leq g(n) - 2$, such that P_i holds, and all nodes representing at least m_{i+1} points of S_i, are contained in $\bigcup_{j=0}^{i} \Pi_j$. Since $|S_i| \geq m_i$, it follows from Lemma 4 that there are at least $\frac{2}{d!} \frac{m_i}{m_{i+1}} (\log \frac{m_i}{m_{i+1}})^{d-1}$ such nodes. (Note that $m_{i+1} \geq 1$, and that $\lfloor m_{i+1} \rfloor + 1 \leq 2m_{i+1} \leq m_i$, since $g(n) \leq \log n$. Hence Lemma 4 may be applied.) Hence

$$|\bigcup_{j=0}^{i} \Pi_j| \geq \frac{2}{d!} \frac{m_i}{m_{i+1}} \left(\log \frac{m_i}{m_{i+1}}\right)^{d-1}.$$

This proves that there is a part of size at least

$$\begin{aligned} \frac{1}{i+1} |\bigcup_{j=0}^{i} \Pi_j| &\geq \frac{1}{i+1} \frac{2}{d!} \frac{m_i}{m_{i+1}} \left(\log \frac{m_i}{m_{i+1}}\right)^{d-1} \\ &= \frac{1}{i+1} \frac{2}{d!} n^{1/g(n)} \left(\frac{1}{g(n)}\right)^{d-1} (\log n)^{d-1} \\ &\geq \frac{2}{d!} \left(\frac{1}{g(n)}\right)^{d} n^{1/g(n)} (\log n)^{d-1}. \end{aligned}$$

Otherwise, if QED has the value **false**, we must have $i = g(n) - 1$. Also property $P_{g(n)-1}$ holds. So we have a set $S = S_{g(n)-1}$ of points of cardinality at least $m_{g(n)-1}$, and a sequence $\Pi_0, \Pi_1, \ldots, \Pi_{g(n)-1}$ of $g(n)$ different parts of the

partition, such that each update passing through a point of S, passes through these $g(n)$ parts. Since each update in the tree passes through at most $g(n)$ parts, it follows that all nodes representing at least one point of S, are contained in $\bigcup_{j=0}^{g(n)-1} \Pi_j$. By Lemma 4 there are at least $\frac{2}{d!}|S|(\log|S|)^{d-1}$ nodes representing at least one point of S. (Here we apply Lemma 4 with $m = 1$, which is allowed since $|S| \geq m_{g(n)-1} \geq 2$.) Hence

$$|\bigcup_{j=0}^{g(n)-1} \Pi_j| \geq \frac{2}{d!}|S|\,(\log|S|)^{d-1} \geq \frac{2}{d!} m_{g(n)-1}\,\left(\log m_{g(n)-1}\right)^{d-1}.$$

It follows that there is a part of size at least

$$\frac{1}{g(n)}|\bigcup_{j=0}^{g(n)-1} \Pi_j| \geq \frac{1}{g(n)}\frac{2}{d!} m_{g(n)-1}\,\left(\log m_{g(n)-1}\right)^{d-1} = \frac{2}{d!}\left(\frac{1}{g(n)}\right)^{d} n^{1/g(n)}\,(\log n)^{d-1}$$

(and we can set QED := **true**). This finishes the proof. □

Consider again the proof of Theorem 3. We write $k = g(n)$. We started with a sequence $m_0, \ldots, m_{k-1}$ of real numbers such that $m_0 = n$, $m_i \geq 1$, $m_i \geq \lfloor m_{i+1} \rfloor + 1$ for all i, and $m_{k-1} \geq 2$. Then it was shown that the partition contains a part of size at least

$$\frac{2}{d!} \times \min\{\frac{1}{i+1}\frac{m_i}{m_{i+1}}\left(\log\frac{m_i}{m_{i+1}}\right)^{d-1}, \frac{1}{k} m_{k-1}\,(\log m_{k-1})^{d-1}\},$$

for some i, $0 \leq i \leq k-2$. Since i can take any value between 0 and $k-2$, there is a part of size at least (we omit the factor $\frac{2}{d!}$)

$$\min\{\quad \frac{m_0}{m_1}\left(\log\frac{m_0}{m_1}\right)^{d-1}, \frac{1}{2}\frac{m_1}{m_2}\left(\log\frac{m_1}{m_2}\right)^{d-1}, \ldots, \frac{1}{k-1}\frac{m_{k-2}}{m_{k-1}}\left(\log\frac{m_{k-2}}{m_{k-1}}\right)^{d-1},$$
$$\frac{1}{k} m_{k-1}\,(\log m_{k-1})^{d-1} \quad \}. \tag{2}$$

Just as in Section 3, it might be possible to improve the lower bound in Theorem 3 by taking another sequence $m_0, \ldots, m_{k-1}$. We shall show that in this way the lower bound can only be improved by a constant factor. More precisely, we shall prove that for any sequence $m_0, \ldots, m_{k-1}$ of positive real numbers, where $m_0 = n$, $m_i \geq 1$ and $m_i \geq m_{i+1}$ for all i, the value of Equation (2) is at most

$$e(1+\log e)^{d-1}\left(\frac{1}{k}\right)^{d} n^{1/k}(\log n)^{d-1}.$$

Take such a sequence $m_0, \ldots, m_{k-1}$. There are two possibilities.

There is an i, $0 \leq i \leq k-2$, such that $m_i \leq \frac{n}{i!}(\frac{k!}{n})^{i/k}$ and $m_{i+1} > \frac{n}{(i+1)!}(\frac{k!}{n})^{(i+1)/k}$. Then, by using Lemma 1,

$$\frac{m_i}{m_{i+1}} \leq (i+1)\left(\frac{n}{k!}\right)^{1/k} \leq (i+1)\,\frac{e}{k}\,n^{1/k},$$

and hence (note that $\log \frac{m_i}{m_{i+1}} \geq 0$, since $m_i \geq m_{i+1}$)

$$\begin{aligned}\frac{1}{i+1}\,\frac{m_i}{m_{i+1}}\left(\log \frac{m_i}{m_{i+1}}\right)^{d-1} &\leq \frac{e}{k}\,n^{1/k}\left(\log\left(\frac{i+1}{k}\,e\,n^{1/k}\right)\right)^{d-1} \\ &\leq \frac{e}{k}\,n^{1/k}\left(\log\left(e\,n^{1/k}\right)\right)^{d-1}.\end{aligned}$$

Hence the value of Equation (2) is at most $\frac{e}{k}\,n^{1/k}(\log(e\,n^{1/k}))^{d-1}$.

Otherwise, for each i, $0 \leq i \leq k-2$, we have that if $m_i \leq \frac{n}{i!}(\frac{k!}{n})^{i/k}$, then $m_{i+1} \leq \frac{n}{(i+1)!}(\frac{k!}{n})^{(i+1)/k}$. In the same way as in Section 3 it follows that $m_{k-1} \leq \frac{n}{(k-1)!}(\frac{k!}{n})^{(k-1)/k} = k\,(\frac{n}{k!})^{1/k} \leq e\,n^{1/k}$. Here the last inequality follows from Lemma 1. Hence (note that $\log m_{k-1} \geq 0$)

$$\frac{1}{k}\,m_{k-1}\,(\log m_{k-1})^{d-1} \leq \frac{e}{k}\,n^{1/k}\left(\log\left(e\,n^{1/k}\right)\right)^{d-1}.$$

Again we conclude that the value of Equation (2) is at most $\frac{e}{k}\,n^{1/k}(\log(e\,n^{1/k}))^{d-1}$.

Now since

$$\begin{aligned}\left(\log\left(e\,n^{1/k}\right)\right)^{d-1} &= \left(\log n^{1/k} + \log e\right)^{d-1} \\ &= \left(\log n^{1/k}\right)^{d-1}\left(1 + \frac{\log e}{\log n^{1/k}}\right)^{d-1} \\ &\leq \left(\log n^{1/k}\right)^{d-1}(1+\log e)^{d-1},\end{aligned}$$

—where the inequality follows from the fact that $\log n^{1/k} \geq 1$ or, equivalently, $k \leq \log n$—it follows that the value of Equation (2) is at most

$$\frac{e}{k}\,n^{1/k}\left(\log\left(e\,n^{1/k}\right)\right)^{d-1} \leq e(1+\log e)^{d-1}\left(\frac{1}{k}\right)^{d} n^{1/k}(\log n)^{d-1},$$

which proves our claim.

6 Conclusions

We have given several lower bounds for the problem of maintaining a range tree in secondary memory. These lower bounds are of the following type: Given a partition of a range tree into parts of size at most $f(n)$, such that each update

passes through at most $g(n)$ of these parts, give a lower bound for $f(n)$. We have studied two types of partitions. In the so-called restricted partitions, only the main tree is divided into parts, and each node of this main tree is contained in the same part as its associated structure. We have shown that for a restricted partition of a d-dimensional range tree, we have $f(n) = \Omega(n\,(\log n)^{d-2}(\log)^{g(n)}n)$. As is shown in [3], this lower bound is tight. For general partitions we have proved that $f(n) = \Omega((\frac{1}{g(n)})^d n^{1/g(n)}(\log n)^{d-1})$.

Acknowledgements

I would like to thank Mark Overmars, Leen Torenvliet and Peter van Emde Boas for reading preliminary versions of this paper, and for some helpful suggestions.

References

[1] J.L. Bentley. *Decomposable Searching Problems.* Inform. Proc. Lett. **8** (1979), pp. 244-251.

[2] G.S. Lueker. *A Data Structure for Orthogonal Range Queries.* Proc. 19-th Annual IEEE Symp. on Foundations of Computer Science, 1978, pp. 28-34.

[3] M.H. Overmars, M.H.M. Smid, M.T. de Berg and M.J. van Kreveld. *Maintaining Range Trees in Secondary Memory, Part I: Partitions.* Report FVI-87-14, University of Amsterdam, 1987.

[4] M.H.M. Smid and M.H. Overmars. *Maintaining Range Trees in Secondary Memory, Part II: Lower Bounds.* Report FVI-87-15, University of Amsterdam, 1987.

[5] D.E. Willard and G.S. Lueker. *Adding Range Restriction Capability to Dynamic Data Structures.* Journal of the ACM **32** (1985), pp. 597-617.