On the Dynamic Finger Conjecture for Splay Trees. Part I: Splay Sorting $\log n$-Block Sequences

Richard Cole
Bud Mishra
Jeanette Schmidt
Alan Siegel

Technical Report 470

October 1989
NEW YORK UNIVERSITY
COURANT INSTITUTE OF MATHEMATICAL SCIENCES
251 MERCER STREET, NEW YORK, N.Y. 10012

NEW YORK UNIVERSITY

Department of Computer Science
Courant Institute of Mathematical Sciences
251 MERCER STREET, NEW YORK, N.Y. 10012
On the Dynamic Finger Conjecture for Splay Trees.
Part I: Splay Sorting $\log n$-Block Sequences

Richard Cole
Bud Mishra
Jeanette Schmidt
Alan Siegel

Technical Report 470
October 1989

*This work was begun while all four authors were at New York University; Richard Cole continued the work while on leave at Ecole Normale Supérieure. Presently Jeanette Schmidt is at Polytechnic University. The work was supported in part by NSF grants CCR-8702271, CCR-8902221, CCR-8906949, DMS-8703458, ONR grants N00014-85-K-0046 and N00014-89-J3042, and by a John Simon Guggenheim Memorial Foundation Fellowship. The work of Richard Cole was made possible, in part, by the hospitality of the Laboratoire d'Informatique, Ecole Normale Supérieure; it is associated with and supported by CNRS as URA 1327.
On the Dynamic Finger Conjecture for Splay Trees. Part I: Splay Sorting log \( n\)-Block Sequences*

Richard Cole; Bud Mishra, Jeanette Schmidt, Alan Siegel

Courant Institute, New York University; Polytechnic University;
Ecole Normale Supérieure

Abstract

A special case of the Dynamic Finger Conjecture is proved; this special case introduces a number of useful techniques.

1 Introduction

The splay tree is a self-adjusting binary search tree devised by Sleator and Tarjan [ST-85]. It supports the operations search, insert and delete, collectively called accesses. The splay tree is simply a binary search tree; each access will cause some rotations to be performed on the tree. Sleator and Tarjan showed that a sequence of \( m \) accesses performed on a splay tree takes time \( O(m \log n) \), where \( n \) is the maximum size attained by the tree \( (n \leq m) \). They also showed that in an amortized sense, up to a constant factor, on sufficiently long sequences of searches, the splay tree has as good a running time as the optimal weighted binary search tree. In addition, they conjectured that its performance is, in fact, essentially as good as that of any search tree. Before discussing these conjectures it will be helpful to review the operation of the splay tree and the analysis of its performance. The basic operation performed by the splay tree is the operation \( \text{splay}(x) \) applied to an item \( x \) in the splay tree. \( \text{splay}(x) \) repeats the following step until \( x \) becomes the root of the tree.

Splay step.

Let \( p \) and \( g \) be respectively the parent and grandparent (if any) of \( x \).

Case 1. \( p \) is the root: Make \( x \) the new root by rotating edge \((x,p)\).

Case 2 — the zig-zag case. \( p \) is the left child of \( g \) and \( x \) is the right child of \( p \), or vice-versa: Rotate edge \((x,p)\), making \( g \) the new parent of \( x \); rotate edge \((x,g)\).

Case 3 — the zig-zig case. Both \( x \) and \( p \) are left children, or both are right children: Rotate edge \((p,g)\); rotate edge \((x,p)\).

*This work was begun while all four authors were at New York University; Richard Cole continued the work while on leave at Ecole Normale Supérieure. Presently Jeanette Schmidt is at Polytechnic University. The work was supported in part by NSF grants CCR-8702271, CCR-8902221, CCR-8906949, DMS-8703458, ONR grants N00014-85-K-0046 and N00014-89-J3042, and by a John Simon Guggenheim Memorial Foundation Fellowship. The work of Richard Cole was made possible, in part, by the hospitality of the Laboratoire d'Informatique, Ecole Normale Supérieure; it is associated with and supported by CNRS as URA 1327.
Henceforth, we refer to the rotation, single or double, performed by the splay step as a rotation of the access or splay operation. A rotation is the basic step for our analysis; the cost of one rotation is termed a unit; clearly, this is a constant.

Sleator and Tarjan use the following centroid potential to analyze the amortized performance of a splay operation. Node \( z \) is given weight, \( wt(z) \), equal to the number of nodes in its subtree; they define the centroid rank of \( z \), or simply the rank of \( z \) to be \( rank(z) = \lfloor \log wt(z) \rfloor \) (our terminology). Each node is given a centroid potential equal, in units, to its centroid rank. Let \( \delta \) denote the increase in centroid rank from \( x \) to \( g \), if \( g \) is present; otherwise it denotes the increase in centroid rank from \( x \) to \( p \). Sleator and Tarjan showed that the amortized cost of the splay step if \( g \) is present is at most \( 3\delta \) units, while if \( g \) is not present the cost is at most \( \delta + 1 \) units. Since the total increase in rank for the complete access is bounded by \( \log n \), the amortized cost of an access is at most \( 3\log n + 1 \) units. More generally, this analysis can be applied to weighted trees, in exactly the same way. We call this the centroid potential analysis.

The operation \( inser(t) \) is performed as follows: first, item \( x \) is inserted as in a binary search tree and then the operation \( splay(x) \) is carried out. Clearly, the cost of an insertion is dominated by the cost of the corresponding access. So, subsequently, when analyzing the cost of insertions we only count the cost of the splays themselves.

We now list the conjectures formulated by Sleator and Tarjan.

- **Dynamic Optimality Conjecture.** Consider any sequence of successful accesses on an \( n \)-node binary search tree. Let \( A \) be any algorithm that carries out each access by traversing the path from the root to the node containing the accessed item, at a cost of one plus the depth of the node containing the item, and that between accesses performs an arbitrary number of rotations anywhere in the tree, at a cost of one per rotation. Then the total time to perform all the accesses by splaying is no more than \( O(n) \) plus a constant times the time required by algorithm \( A \).

- **Dynamic Finger Conjecture.** The total time to perform \( m \) accesses on an arbitrary splay tree, initially of \( n \) nodes, is \( O(m + n + \sum_{j=1}^{m-1} \log(|i_{j+1} - i_j| + 1)) \), where for \( 1 \leq i \leq m \) the \( j \)th splay is performed on item \( i_j \) (items are denoted by their current symmetric-order positions).

- **Traversal Conjecture.** Let \( T_1 \) and \( T_2 \) be any two \( n \)-node binary search trees containing exactly the same items. Suppose we access the items in \( T_1 \) one after another using splaying, accessing them in the order they appear in \( T_2 \) in preorder (the item in the root of \( T_2 \) first, followed by the items in the left subtree of \( T_2 \) in preorder, followed by items in the right subtree of \( T_2 \) in preorder). Then the total access time is \( O(n) \).

Sleator and Tarjan state that the Dynamic Optimality Conjecture implies the other two conjectures (the proof is non-trivial).

There have been several works on, or related to, the optimality of splay trees [STT-86], [W-86], [T-85]. [STT-86] shows that the rotation distance between any two binary search trees is at most \( 2n - 6 \) and that this bound is tight; they also relate this to distinct triangulations of polygons; although connected to the splay tree conjectures, this result has no immediate application to them. [W-86] provides two methods for obtaining lower bounds on the time for sequences of accesses to a binary search tree; while some specific tight bounds are obtained (such as accessing the bit reversal permutation takes time \( \Theta(n \log n) \)) no general results related
to the above conjectures follow. [T-85] proved the Scanning Theorem, a special case of the Traversal Conjecture (also a special case of the Dynamic Finger Conjecture): accessing the items of an arbitrary splay tree, one by one, in symmetric order, takes time \( O(n) \).

In this paper we investigate a special instance of the Splay Sorting problem, which is related to the Dynamic Finger Conjecture (it is the restriction of this conjecture to just the insert operation). Splay Sorting is defined as follows. Consider sorting a sequence of \( n \) items by inserting them, one by one, into an initially empty splay tree: following the insertions, an inorder traversal of the splay tree yields the sorted order. We call this Splay Sort. A corollary of the Dynamic Finger Conjecture is the:

**Splay Sort Conjecture.** Let \( S \) be sequence of \( n \) items. Suppose the \( i \)th item in \( S \) is distance \( I_i \) in sorted order from the \( i - 1 \)th item in \( S \), for \( i > 1 \). Then Splay Sort takes time \( O(n + \sum_{i=2}^{n} \log(I_i + 1)) \).

Incidentally, an interesting corollary of the Splay Sort Conjecture is the:

**Splay Sort Inversion Conjecture.** Let \( S \) be sequence of \( n \) items. Suppose the \( i \)th item in \( S \) has \( I_i \) inversions in \( S \). Then Splay Sort takes time \( O(n + \sum_{i=1}^{n} \log(I_i + 1)) \).

In the remainder of this paper we prove the Splay Sort Conjecture for the following type of sequence. Suppose the sorted set of \( n \) items is partitioned into subsets of \( \log n \) contiguous items, called *blocks*. Consider an arbitrary sequence in which the items in each block are contiguous and in sorted order. We call such a sequence a \( \log n \)-block sequence. We show an \( O(n) \) bound for Splay Sorting a \( \log n \)-block sequence. It is convenient to assume that the set being sorted comprises the integers \( 1 \ldots n \).

In brief, our analysis has the following form. The first insertion in each block is provided \( \Theta(\log n) \) potential; it is called a *global* insertion. Every other insertion in the block is provided \( O(1) \) potential; these insertions are called *local* insertions. In Section 2, we analyze the global insertions; then, in Section 3, we modify the analysis to take account of local insertions.

Our work has a number of interesting features and introduces several new techniques for proving amortized results.

- We introduce the notion of *lazy potential*; this notion can be viewed as a tool for designing potential functions. The lazy potential is a refinement of an initial potential function that avoids waste when potentials decrease. (For an example of waste, consider the following splay tree analyzed using the centroid potential. The tree is a path of \( n \) nodes, each of unit weight. The last node on the path is accessed. The resulting splay will have a real cost of \( \Theta(n) \), but will reduce the potential by \( \Theta(n \log n) \). The desired amortized cost of this operation is \( O(\log n) \), so essentially all the reduction in potential is wasted.)

  The idea of the lazy potential is to keep an old potential, \( \phi_{old} \), in situations in which the creation of a new (larger) potential, \( \phi_{new} \), may be followed by a return to the \( \phi_{old} \) potential, with essentially \( \phi_{new} - \phi_{old} \) potential being wasted. Not surprisingly, some care is needed in choosing which operations to treat as 'lazy'.

- The potential depends on the access sequence; that is, identical trees, if created by distinct access sequences, may have different potentials. (Actually, this is a difference 'in principle'; we have not constructed any examples to demonstrate it.) Nonetheless, the bounds we obtain do not depend on the input sequence.

3
2 Global Insertions

We start with some definitions.

The left path of a tree comprises the nodes traversed in following left child pointers from
the root, but excluding the root itself; the right path is defined analogously. Collectively, they
are called extreme paths. Extreme paths of a subtree rooted at v are defined analogously; they
are also called v's extreme paths. Node u is a right ancestor of node v if u is an ancestor of
v that is to the right of v in symmetric order; note that v must be in u's left subtree. A left
ancestor is defined analogously. A right edge is an edge to a right child; a left edge is defined
analogously.

A block B is an interval \([i, j] \subseteq [1, n]\) of items; here a block always comprises \(\log n\) items
with \(i = c \log n + 1\) and \(j = (c + 1) \log n\), for some integer c. Any block \([i, j]\) induces a binary
tree \(T_B\), called the block tree of \(B\), which comprises exactly those nodes of the splay tree, \(S\),
containing items i to j; they are called the nodes of \(B\). Loosely speaking, the block tree is
constructed by shrinking paths in \(S\) between nodes of \(B\) to single edges. More formally, the
root, \(r\), of \(T_B\) is the lowest common ancestor of nodes \(i\) and \(j\) in \(S\). The left (resp. right)
subtree of \(r\) is the tree induced by the set of items in \(S\) to the left (resp. right) of \(r\), if
non-empty; otherwise the subtree is empty.

The root of block \(B\) is the root of the corresponding block tree. The global nodes are exactly
the block roots. The remaining nodes are called local nodes.

Every node on an extreme path carries a potential of \(c\) units, \(c\) a constant to be specified
later. Also, each node on an extreme path may carry a debit, either small or large, comprising
sd and ld units, respectively; sd and ld are constants that are specified later. A node may not
have both a small and a large debit. We note that for any block \(B\) the only nodes that may
be visited henceforth are those presently on the extreme paths of its block tree, plus its root.

Each global node is given a centroid potential, called its global potential; it is defined on
the splay tree using the following weights: each global node has weight 1 and each local node
has weight 0. It is convenient to define a global rank for all nodes: this is the centroid rank
in the splay tree under this weighting: \(g\text{-}rank(v)\) denotes the global rank of \(v\). Global nodes
have a global potential equal to \(gp\) times their global rank, \(gp\) a constant to be specified later;
the local nodes do not have a global potential.

In the analysis of global insertions, whenever a rotation is performed (and paid for) another
s spare rotations are also paid for, s a constant to be specified later. The spare rotations are
needed subsequently, to handle the effects of local insertions. We provide \((3 \log n + 1)gp\) units
to pay for a global insertion. If a rotation increases the global rank of the item being inserted
by \(\Delta\), then the rotation is paid for with \(3\Delta gp\) of these units. If the last rotation (the one
involving the root of the splay tree) involves just two nodes, we call it an incomplete rotation;
the remaining \(gp\) units are used as additional payment for the incomplete rotation, if any.

To avoid special cases it is convenient to redefine the access path for an insertion to exclude
the splay tree root \(r\) in the event that \(r\) is involved in an incomplete rotation. Now consider
a rotation performed during the splay along the access path. Of the three nodes involved in
the rotation, the top two are called the coupled nodes of the rotation, or a couple for short.
The analysis focusses on the coupled nodes in a rotation. In order to provide the reader some
intuition we describe two simple cases.

Case 1 The coupled nodes are both local nodes in the same block (see Figure 1). Nodes \(u\) and
\(v\) are on an extreme path, without loss of generality the left path; \(v\) is the left child of \(u\), and
Let \( w \), the node being inserted, be the left child of \( v \). Since node \( u \) leaves the extreme path of its block it loses potential \( c \). This pays for the removal of debits (small or large) from nodes \( u \) and \( v \) and in addition pays for \( s + 1 \) rotations. Later, we see that we need \( c \geq s + 1 + \max\{2sd, ld\} \).

**Case 2** The coupled nodes are both global nodes (see Figure 2). Again, let \( w \) be the inserted node, let \( v \) be \( w \)'s parent and \( u \) be \( v \)'s parent. This case is analysed using the centroid potential analysis of Sleator and Tarjan. The cost is at most 3 times the jump in global potential from \( w \) to \( u \); to pay for \( s + 1 \) rotations we therefore need \( gp \geq s + 1 \); later, we see that in fact we need \( gp \geq 6(s + 1) + 11sd \).

The full analysis is more involved. We begin by stating several invariants about the debits.

**Invariant 1** *Only local nodes on an extreme path can have debits.*

**Invariant 2** Let \( v \) be a local node on the extreme path of block \( B \). Suppose that in the splay tree \( v \) is the left (resp. right) child of its parent \( u \). \( v \) can have a large debit only if

(i) \( u \) is a local node of block \( B \),

(ii) \( v \) has a left (resp. right) child \( w \) which is a local node of block \( B \), and

(iii) neither \( u \) nor \( w \) carry any debit.

**Invariant 3** Let \( u \) be the root of block \( B \). Let \( v \) be a child of \( u \). If \( v \) is in \( B \), then \( v \) can have a small debit only if \( g_{-\text{rank}}(v) < g_{-\text{rank}}(u) \).

**Invariant 4** Let \( u \) be the root of block \( B \). Let \( w \) be the last node in block \( B \) on an extreme path descending from \( u \). \( w \) can have a small debit only if \( g_{-\text{rank}}(w) < g_{-\text{rank}}(u) \).

For the purposes of the analysis the access path is partitioned into segments. Each segment comprises an even number of nodes, every two nodes on the path forming the coupled nodes of a rotation of the present splay operation. The segments are created by a traversal of the access path from bottom to top; each segment is chosen to have the maximum length such that following the removal of its front (top) two nodes, the (truncated) segment satisfies the following conditions:

(i) The node being inserted has the same global rank throughout the rotations involving the truncated segment.

(ii) Each global node on the truncated segment has the same rank following its rotation.

(iii) (This is implied by (i) and (ii).) Each couple in the truncated segment includes at least one local node (i.e., it does not comprise two global nodes).

(iv) Define a node to be *visible* if it is involved in a zig-zag rotation or it is the lower node in a couple. (Intuitively, the visible nodes are those that remain on one of the traversed paths following the splay. Note that the splay, in general, creates two traversed paths.) For each block there are at most two visible nodes in the truncated segment that are local following the rotation.
The topmost segment is said to be incomplete if it satisfies conditions (i)-(iv) prior to truncation. We consider an incomplete segment to comprise a (trivially) truncated segment.

Next, we mark the following types of coupled nodes in each truncated segment. The rotations involving marked couples are self-paying, as is demonstrated later. For each type below, suppose the segment includes a couple, \( u, v \), with \( u \) the parent of \( v \).

**Type 1** See Figure 3. Suppose \( u \) is the root of \( v \)'s block; then both \( u \) and \( v \) are marked.

**Type 2** See Figure 4. Suppose \( u \) is a local node and let \( x \) be the root of \( u \)'s block. Further suppose \( u \) is on the left (resp. right) path descending from \( x \). Let \( v \) be the left (resp. right) child of \( u \); if \( v \) is global then both \( u \) and \( v \) are marked.

**Type 3** Suppose \( u \) and \( v \) are both local nodes of the same block; then both \( u \) and \( v \) are marked.

We can now prove a bound on the length of a truncated segment.

**Lemma 1** A truncated segment comprises at most 10 unmarked nodes, of which at most 7 are local.

**Proof.** Observation 1. Define the left (resp. right) side of the segment to comprise those nodes that are to the left (resp. right) of the item being inserted. From (ii) we deduce that one side, at least, contains no global nodes; without loss of generality suppose that this is the right side. As the right side contains no global nodes, its nodes must all come from the same block; so, by (iv), the right side comprises at most two sets of contiguous nodes, each one of at most two nodes. Because of Type 3 markings the right side contains at most two unmarked nodes.

The left side of the segment is partitioned in the obvious way into subsegments by the nodes on the right side. By Observation 1, the left side comprises at most three subsegments.

Observation 2. There are at most two couples that include both a node on the left side and a node on the right side. This follows from (iv) applied to the right side of the segment.

Observation 3. There is at most one unmarked couple within each subsegment; if present, this couple comprises the topmost global node and its parent, a local node. For the marking strategy marks all other couples within the subsegment. This is a total of at most three unmarked couples.

Thus there are a total of at most 5 unmarked couples, and at most 2 of these do not include a global node; so there are at most 7 unmarked local nodes. •

Next, we show how to pay for the rotations (and associated spares) along a segment. Each marked couple pays for its rotation (and spares), as follows.

**Type 1** See Figure 3. By Invariants 2 and 3, node \( v \) does not have a debit. The rotation is paid for by giving node \( u \) a small debit following the rotation. It is straightforward to check that Invariants 1-4 are unaffected. So it suffices that:

\[ sd \geq s + 1 \]  

(1)

**Type 2** See Figure 4. By Invariants 2 and 4, node \( u \) does not have a debit. The rotation is paid for by giving \( u \) a small debit following the rotation. Again, it is straightforward to check that Invariants 1-4 are unaffected. Here too, Equation 1 suffices.
**Type 3** This is the Case 1 analyzed earlier. We need to pay for the rotation plus the removal of two small debits or one large debit. Again, it is straightforward to check that Invariants 1-4 are unaffected. So it suffices that:

\[ c \geq s + 1 + \max\{2sd, ld\} \quad (2) \]

The remaining rotations are paid for by the first couple in the segment, which was removed in truncating the segment and which causes a violation of at least one of the conditions (i)-(iv), except in the case of an incomplete topmost segment, which is handled subsequently. Either the rotation involving the first couple causes a change in global potential (case A) or it creates a sequence of three contiguous local nodes from the same block (Case B), or possibly both. Both cases involve three costs.

- **Cost1.** The rotation and spares for each couple: \( \leq 6(s + 1) \).
- **Cost2.** Removal of small debits from nodes on the segment; Cost2 is analyzed below.
- **Cost3.** Removal of small debits for local nodes that now violate Invariant 3 or 4; Cost3 is analyzed below.

**Lemma 2** *For each segment* \( \text{Cost2} + \text{Cost3} \leq 11sd. \)

**Proof.** A definition is helpful here. Let \( B \) be a block and let \( v \) be the root of \( B \). Consider an extreme path of block \( B \) and consider the topmost portion that is contiguous in the splay tree. If this topmost portion is incident on \( v \) in the splay tree then it is called an *abuting extreme path portion* for \( v \), or an *abuting portion* for short. (*Comment. Node \( v \) of Invariant 3 is the top node of an abuting portion for \( u \), while node \( w \) of Invariant 4 is the bottom node of an abuting portion for \( u \).*)

Next, we make some observations about Cost3. Note that none of the nodes from marked couples on the traversed path retain small debits; so these nodes do not contribute to Cost3. Contributions to Cost3 can arise in one of two ways:

1. **Through a traversed global node \( v \) having a reduction in rank.** Then, on an already abuting portion (for \( v \)) which continues to be abuting (for \( v \)), if the portion was not traversed, we may need to remove up to two single debits in order to maintain Invariants 3 and 4. \( v \) can have at most one such abuting portion. This contributes up to \( 2sd \) to Cost3. This contribution can be a consequence of either a zig-zag rotation or a zig-zig rotation. We examine each in turn.

   a. **In a zig-zag rotation, there may be one such abuting portion for each global node in the couple.** In this case the contribution to Cost3 can be as large as \( 4sd \).

   b. **In a zig-zig rotation only the top node in the couple can retain such an abuting portion; so here the contribution to Cost3 is at most \( 2sd \).**

2. **By the creation of an abuting portion, in the case that this abuting portion is not traversed.** This can only occur for the top node \( v \) of couple and then only if both \( u \) and \( v \) are global and the rotation is a zig-zig rotation. Here, the contribution to Cost3 is at most \( 2sd \).
Next, we determine, in turn, the contribution to $Cost_2 + Cost_3$ of the truncated segment and of the first couple.

First, we consider the truncated segment. By Lemma 1 the contribution to $Cost_2$ is at most 7sd. For a contribution to $Cost_3$, the discussion of the previous paragraph shows we need only consider possibility (2) since global nodes in the truncated segment do not have a change in global rank. Likewise, possibility (2) cannot arise for the couples of the truncated segment each have at most one global node. We conclude that the truncated segment contributes at most 7sd to $Cost_2 + Cost_3$.

We turn to the leading couple. The contribution to $Cost_2$ is at most sd for each local node it contains. By the above discussion regarding contributions to $Cost_3$, the leading couple may contribute up to 4sd to $Cost_3$, but only if both nodes of the couple are global; if there is only one global node in the couple then the contribution to $Cost_3$ is at most 2sd. In any event, the total contribution of the first couple to $Cost_2 + Cost_3$ is at most 4sd. •

In case A, the cost of the rotation is paid for either by the drop in global potential, which provides at least $gp$ units, or, if there is an increase in global rank, by $gp$ units of the at least 3gp units provided for this rotation. In case B, the middle node in the sequence of three contiguous nodes is given a large debit, which pays for the rotation. So it suffices to have:

$$gp, ld \geq 6(s + 1) + 11sd$$

Now, we show how to pay for the incomplete segment, if present. Recall that we provided $(3 \log n + 1)gp$ units to pay for the present insertion. The $+gp$ term is used to pay for the incomplete segment; in addition, this term is used to pay for the incomplete rotation, if any; however, the $+gp$ term does not need to account for any increase in rank, on the part of the inserted item during the incomplete rotation, for this has already been accounted for. Clearly, the result of Lemma 2 applies here too (in fact, a tighter bound can be shown). Here too, Equation 3 suffices.

On taking equalities in Equations 1-3, we conclude:

**Lemma 3** A global insertion costs at most $(3 \log n + 1)gp$ units, where $gp, ld = 17(s + 1)$, $c = 18(s + 1)$ and $sd = s + 1$.

# 3 Local Insertions

We start by providing an overview of the analysis of a sequence of local insertions. This leads to the introduction of lazy trees which forces a reanalysis of global insertions. Following this reanalysis, the overall analysis is readily concluded.

Recall that the insertion of each block comprises one global insertion followed by a sequence of $\log n - 1$ local insertions. Each local insertion traverses a left path up to the right child of the splay tree root (see Figure 5); we call this path the local access path. By providing $e \cdot (3 \log n + 1)gp$ units to the global insertion we can treat the first $e - 1$ local insertions as if they were global insertions. Throughout these $e - 1$ local insertions, spare rotations will be accumulating (though for reasons that will become clear later it may be that no spares accumulate during the global insertion itself); after $e - 1$ local insertions, each node on the local access path, except the topmost, can be given at least $(2^{e-1} - 1)s$ spare rotations. The rotations in subsequent local insertions will be self-paying, apart from the rotation involving the splay tree root. We analyze these subsequent local insertions, called true local insertions.
In fact, we provide another $\log n \cdot gp$ units to the global insertion. Following the first $e - 1$ local insertions we provide a reserve potential to each global node on the local access path. The reserve for global node $u$ is defined as follows: let $v$ be the first proper global descendant of $u$ on the local access path, if any; let $g\cdot rank(v)$ be $v$'s global rank, if $v$ is present; otherwise, let $g\cdot rank(v) = 0$. Then $reserve(u) = gp(g\cdot rank(u) - g\cdot rank(v))$. The role of the reserve potential will become clear later.

We need one further constant, $q$, to be specified later. We choose $e$ so that

$$(2^{e-1} - 1)s \geq q + (s + 1)$$

In a true local insertion, each node $v$ removed from the local access path is given potential $q$; $s + 1$ will bound the cost of the rotation involving $v$. We note $e \geq 2$. At the end of the sequence of local insertions, the nodes remaining on the local access path, apart from the topmost node, are all also given potential $q$.

In a true local insertion, there are three types of rotation on the local access path. They all involve a couple $u, v$, with $u$ the parent of $v$. Either both $u$ and $v$ are local, or one of $u$ or $v$ is local and the other is global, or both are global. Only in the last case is a global node removed from the local access path. The rotations, apart from those involving two global nodes, cost at most $s + 1$ units, for there is no increase in global rank in such rotations and there are no debits to remove (for following the global insertion there are only large debits on the access path, which are removed by the first local access; as $e \geq 2$, the current access is not the first local access). In the case of a rotation involving two global nodes we swap their global potentials. The node remaining on the local access path still has the correct global potential, but the node $u$ removed from the path may have too small a potential; we call $u$'s present potential its lazy potential. Again, the cost of this rotation is $s + 1$ units.

Every node removed from the access path in a true local insertion is called a lazy node, whether or not it has a lazy potential. When the insertion of the current block is completed we form lazy trees. Each global node $v$ remaining on the local access path becomes the root of a new lazy tree; this lazy tree has an empty left subtree and a right subtree comprising those lazy nodes created during the insertion of the current block that are in $v$'s right subtree in the splay tree. The intuition behind the lazy tree is that if all the lazy nodes were restored to a left path then the lazy potentials would be the actual global potentials (perhaps with some swapping and shifting). Actually, difficulties are caused by the fact that the left subtree of the path may increase in size through rotations between the lazy tree and the remainder of the splay tree. So strictly speaking the intuition may be incorrect; nonetheless, it is a helpful guide. The constant potentials $q$ provided to the lazy nodes will pay for rotations that move the lazy tree towards a left path. It is straightforward to see that each new lazy node is contained in a new lazy tree. We call the lazy trees as defined above full lazy trees.

Let us return to the cost of the insertions. A global insertion, as noted above, costs $e \cdot (3 \log n + 1)gp + \log n \cdot gp$ units. We see below that the presence of lazy trees adds a further $3gp\log n$ units to the cost of the global insertions, for a total of $e \cdot (3 \log n + 1)gp + 4\log n \cdot gp$ units. A local insertion costs $c + s + 1$ units (the $s + 1$ units are used to pay for the final insertion of the rotation and the $c$ units are given to the node displaced as splay tree root, for this node becomes an additional node on an extreme path of its block). So we conclude that the cost of sorting a log $n$-block sequence is bounded by:

$$(3e \cdot gp + 4gp + c + s + 1)n + n/ \log n(e \cdot gp - c - s - 1)$$

(5)
In Section 3.2 we show how to modify the analysis of local insertions to account for the presence of lazy trees. This will require a further reanalysis of the global insertions.

3.1 The Analysis of Lazy Trees

Most of the analysis focusses on a subtree of the full lazy tree, called the truncated lazy tree or the lazy tree, for short. It is defined as follows. Consider a new full lazy tree, \( L \), created by a sequence of local insertions. Consider the set of global nodes in \( L \); the tree they induce is called the full lazy block tree. We remove the rightmost node from the full block tree; this defines the (truncated) lazy block tree for the (truncated) lazy tree. This rightmost node is called the right guard for the lazy tree. The left guard is a global node, defined later, to the left of the nodes of the lazy block tree. The root of the (truncated) lazy block tree is called the (truncated) root of the (truncated) lazy tree. We define the tree induced by the nodes of the (truncated) lazy block tree plus the left and right guards to form the large lazy block tree. Now we define the (truncated) lazy tree as follows. It comprises the nodes of the (truncated) block tree together with the following local nodes. For each node \( v \) in the (truncated) block tree we add the following nodes from \( v \)'s block to the skeleton. Let \( w \) be any descendant of \( v \) in the large lazy block tree. Those nodes in \( v \)'s block on the path from \( v \) to \( w \) in the splay tree are added to the (truncated) lazy tree. The nodes in the lazy block tree are called global nodes of the lazy tree, while the remaining nodes in the lazy tree are called local nodes. The (truncated) lazy block tree is the tree induced by the global nodes of the (truncated) lazy tree.

We define the left guard, \( w \), of the lazy tree as follows. Let \( L \) be a new lazy tree with root \( u \). Let \( v \) be the the first proper global descendant of \( u \) on the local access path, if any. Suppose \( v \) exists; if \( v \) is the root of another new lazy tree, let \( w \) be the right guard in this lazy tree, while if \( v \) is not the root of a new lazy tree, then let \( v = w \). Otherwise, let \( w \) be the root of the splay tree. In general, a node may be a right guard in one lazy tree and a left guard in a second lazy tree.

Intuitively, a lazy tree is a superblock comprising several of the blocks at hand. Rotations involving a local node within a lazy tree are handled in ways similar to those used before; the principal novelty lies in our treatment of rotations between roots of blocks within the lazy tree.

The right guard of each new lazy tree has its normal potential restored (this is paid for by its associated reserve potential, which suffices, as is shown in Lemma 4, below). Thus both the left and right guards of each lazy tree have their normal potentials.

The left (resp. right) extreme path of lazy tree, \( L \), is the path from the root of \( L \) to the leftmost (resp. rightmost) node in \( L \); it is convenient to exclude the lazy tree root from the extreme paths.

Next, we make some observations about the lazy ranks and reserve potentials. Consider the global nodes in a new lazy tree. Let \( SL \) be the set of nodes contained between the blocks of the guards of a lazy tree. For each global lazy node, \( u \), in the lazy tree, define its right neighbor \( r.n(u) \) to be the global node immediately to its right in the large lazy block tree, and define \( SL(u) \) to comprise the subset of \( SL \) strictly to the left of \( u \)'s block. Then

**Lemma 4**

(i) \( \text{lazyrank}(v) \geq \lfloor \log(\text{wt}(SL(v))) \rfloor \).

(ii) If \( v \) has no lazy global node in its right subtree, \( \text{lazyrank}(v) + 1/\text{gp-reserve}(v) \geq \text{rank}(v) \). While if \( v \) does have a lazy global node in its right subtree, then \( \text{lazyrank}(v) + 1/\text{gp-reserve}(v) \geq \text{lazyrank}(r.n(v)) \).
Proof. Both claims are readily seen by considering node \( v \) at the point at which it becomes lazy. 

It is convenient to define \( v \)’s lazy weight to be \( \text{wt}(SL(v)) \).

Next, we provide additional potentials to the nodes in the lazy tree (in addition to the potentials these nodes already carry). All local nodes on the lazy tree carry a potential of \( 2c' \) units, except for the local nodes from the leftmost block in the lazy tree, which carry a potential of only \( c' \) units; \( c' \) is a constant to be specified later. In addition, we give the following potential to the global nodes in the lazy block tree. A node of height \( h \) in the lazy block tree, other than the root, receives potential \( a \sum_{i=1}^{h} i + b \), where \( a \) and \( b \) are constants to be specified later. This potential is provided by redistributing the potential \( q \) given to each lazy node in the full lazy tree. Note that each full lazy tree has the following form: Consider the right path descending from its root; the left subtree of each node on this path, excluding the root, is a complete binary tree; in top to bottom order, these trees have strictly decreasing height. The \( q \) potentials are redistributed as follows. First, a potential of \( a \sum_{i=1}^{k} i + b \) is given to each height \( k \) node in the full lazy tree, other than the lazy tree root. The following argument shows that it suffices to provide each node with a potential of \( 4a + b \). Each node of height \( k \) needs a potential of \( \frac{1}{2}ak(k+1) + b \); it provides this by passing a charge of \( a[\frac{1}{2}(k-1)k + 2(k - 1)] \) to each of its children; if it has a parent, in turn, it receives a charge of \( a[\frac{1}{2}k(k+1) + 2k] \) from the parent; adding its own original potential of \( 4a + b \) provides exactly the required final potential (note that a leaf will pass a charge of 0 to its non-existent children). So it suffices to have

\[
4a + b \leq q
\]  

But the potentials in the full lazy tree upper bound the potentials desired in the (truncated) lazy block tree. Finally, to provide the local nodes with their potential, it suffices to have

\[
2c' \leq a + b
\]  

Nodes on the extreme paths of the lazy tree may have a path debit, either small or large, of value \( sd \) and \( ld \) units, respectively. Each local node in the lazy tree may have a lazy debit, which is huge and has value \( hd \) units, \( hd \geq ld \), a constant to be defined later.

We now give several invariants concerning lazy and path debits.

**Invariant 5** Let \( L \) be a lazy tree. Suppose node \( v \) in \( L \) has a path debit. Then \( v \) is on an extreme path of \( L \).

**Invariant 6** Let \( L \) be a lazy tree. Suppose node \( v \) in \( L \) has a lazy debit. Then \( v \) is a local node of \( L \). Also, \( v \) is not on the left extreme path of \( L \).

**Invariant 7** A node does not have both a path debit and a lazy debit.

**Invariant 8** Let \( L \) be a lazy tree. Let \( v \) be a node in \( L \) with a large path debit. Suppose that in the splay tree \( v \) is the left (resp. right) child of its parent \( u \).

(i) \( u \) is on an extreme path of \( L \).

(ii) Let \( w \) be the left (resp. right) child of \( v \) in the splay tree; then \( w \) is on an extreme path of \( L \).

(iii) Neither \( u \) nor \( w \) carry any debit.
Invariant 9 Let $u$ be the root of lazy tree $L$. Let $v$ be a child of $u$ in the splay tree. If $v$ is in $L$, then $v$ can have a small path debit only if $g$.rank$(v) < g$.rank$(u)$.

Invariant 10 Let $u$ be the root of lazy tree $L$. Consider the right (resp. left) path in the splay tree descending from $u$. Let $w$ be the last node on this path that is also on an extreme path of $L$. $w$ can have a small path debit only if $g$.rank$(w) < g$.rank$(u)$.

Invariant 11 Let $L$ be a lazy tree. Let $u$ be a local node of $L$. Suppose $u$ has a lazy debit. Let $v$ be the root of $u$'s block.

(i) Suppose that $v$ is not on the left extreme path of $L$. Then if $u$ is on the right path descending from $v$ in the splay tree, both the parent and child of $u$ on this path are local nodes in $u$'s block.

(ii) If $u$ is on the right extreme path of $L$ then its parent and child in the splay tree are local nodes in $u$'s block; this holds regardless of whether $u$ is on the right path descending from $v$ in the splay tree.

Next, we show how to incorporate lazy trees into the analysis of global insertions. A global insertion can traverse a lazy tree in one of three ways:

(a) Traverse the right extreme path of the lazy tree (or rather a topmost portion of it).

(b) Traverse the left extreme path of the lazy tree (or rather a topmost portion of it).

(c) Traverse the interior of the lazy tree and thereby split the lazy tree.

Actually, it is convenient to classify a traversal of type (a) which is to the left of the right guard to be a split (a type (c) traversal); likewise a traversal of type (b) to the right of the left guard is defined to be a split. As we will see later, local insertions only involve traversals of type (a) or (b).

A traversal of Type (c) will be paid for in two phases. First, in a preprocessing phase, the current lazy tree is partitioned into several lazy trees and/or ordinary blocks, so as to ensure that the actual splay (the second phase) comprises only traversals of Types (a) and (b). (In fact, as we will see, we need a third phase in order to pay for some of the partitioning performed in the first phase.)

We start by considering the interactions between the lazy tree and the remainder of the splay tree (which may include other lazy trees). We treat the lazy tree, as delimited by its extreme paths, in essentially the same way as a block. (The root of the lazy tree corresponds to the root of a block, while the extreme paths of the lazy tree correspond to the extreme paths of the block tree.) The root of the lazy tree behaves in the same way as the root of a block. Small path debits are created in rotations with the root and in rotations in which the parent node is on an extreme path of the lazy tree. Large path debits are created in paying for the traversal of zig-zag paths. In addition, we note that on creation, the nodes on the skeleton of the lazy tree have no debits; so there is no possibility of a node having a debit both as a block element and a lazy tree element. Segments are defined and paid for exactly as before (the only change will be in the definition of marked couples).

We now look, in turn, at the three possible ways of traversing a lazy tree.
3.1.1 Right Path Traversal

We need to consider a number of cases. So consider a couple comprising nodes $u$ and $v$, where $u$ is the parent of $v$ and at least one of $u$ and $v$ is on the extreme right path of the lazy tree. 

**Case 1.** $u$ and $v$ are both local. $u$ ceases to be on the skeleton. $u$'s $c'$ potential pays for the rotation, $s$ spares, and the removal of path or lazy debits on $u$ and $v$. So it suffices to have:

$$c' \geq s + 1 + 2hd \tag{8}$$

**Case 2.** $u$ is a local node and $v$ is a global node. By Invariant 11, $u$ does not have a lazy debit. If $u$ leaves the skeleton, the operation is paid for by $u$'s $c'$ potential, as in Case 1; Equation 8 suffices. Otherwise, $u$ is given a lazy debit; this then pays for the operation. The cost of the operation comprises the rotation, $s$ spares, and the removal of path debits, if any, from $u$ and $v$. So it suffices to have:

$$hd \geq s + 1 + \max\{ld, 2sd\} = s + 1 + ld \tag{9}$$

**Case 3.** $v$ is a local node, $u$ is the root of $v$'s block but is not the lazy block tree root. By Invariant 11, $u$ and $v$ do not have lazy debits. $v$ becomes the block root. As in Case 2, if $u$ leaves the skeleton, the operation is paid for by $u$'s $c'$ potential. Otherwise, $u$ is given a lazy debit, which pays for the operation. The cost of the operation comprises the rotation, $s$ spares, and the removal of path debits, if any, from $u$ and $v$. Here too Equations 8 and 9 suffice.

**Case 4.** $u$ and $v$ are both global nodes with lazy potentials. We need to pay for the rotation, for $s$ spares, and for the removal of path debits from $u$ and $v$, if any. In addition, we may need to reestablish Invariant 11 for node $u$; this may require the removal of up to two lazy debits, which will also be paid for by the rotation. Paying for all these is the subject of the rest of this section, following Case 5. The cost of this rotation is

$$s + 1 + ld + 2hd \tag{10}$$

**Case 5.** The remaining couples all include either the right guard or the root of the lazy tree, or a node outside the lazy tree, or they involve an increase in global rank for the inserted item. Because of Invariant 11, these couples do not include a node with a lazy debit. Thus the situation is completely analogous to the previous analysis of global insertions, except that now we view the lazy tree as a “block” and its “local” nodes are the nodes on its extreme paths; its “root” is the root of the lazy block tree; the guards, for this purpose, are not considered to be part of the lazy tree. The path debits are treated in exactly the same way as the debits used in the previous analysis; since they have the same values, it does not affect this analysis.

We define segments as before. The couples of Cases 1-4, above, each involve two “local” nodes of the lazy tree; they are all self-paying and are made into marked pairs. The one change to the previous analysis is that here a couple comprising two “local” nodes may involve an increase in global rank on the part of the inserted item (if at least one of these two “local” items is a global item with a lazy potential). We define a segment to end at such a couple, as before. The way we pay for such a segment changes slightly: the couple itself is responsible for paying for the removal of any lazy debits needed to maintain Invariant 11 (this can only occur in Case 4). The remaining costs, paying for the rotations of unmarked couples in the segment and consequential removals of path debits, are charged to $gp$ times the increase in global rank occurring at the leading couple of the segment, as before. It is a simple matter to check that the bound on the number of unmarked pairs in a segment is unchanged, as is the cost of paying for the segment.
For the remainder of Section 3.1.1 we focus on the lazy block tree. Hence when we refer to a node we mean a node in the lazy block tree; likewise a reference to a tree refers to the lazy block tree. The depth of a node in the tree is its distance from the root. The initial height of a node is called its initialHeight; the initialHeight does not change subsequently. At any time, certain nodes, called active nodes, are the nodes that pay for a traversal of the right path. Some nodes may not pay more than other nodes. This is captured by the notion of active layers. A node of initialHeight $h$ has an associated span of layers $[1, h]$; each active node $v$ of initialHeight $h$ has an active span of active layers, $(i, h]$, $0 \leq i \leq h$; for each $l$, $i \leq l \leq h$, we say $v$ is $l$-active. If the active span is non-empty we say the node is active.

Initially, only the nodes on the (extreme) right path are active. An inactive node becomes active when it first reaches the right path. Once a node becomes active it remains active, whether or not it remains on the right path; also, the active span of a node can only grow. The following invariant states several properties of the active nodes. We prove the invariant later.

In order to avoid special cases for the (extreme) left path we state our invariants with respect to a normal form of the tree, defined as follows. Given a tree, its normal form is obtained by performing a series of single rotations which move the left path nodes, one by one, to the right path; each such rotation between node $v$ and node $u$, the root of the tree, makes $v$ the root and places $u$ on the right path. Node $u$ acquires node $v$'s active span, its potentials, and its initialHeight. Such a tree is called a normal tree.

**Invariant 12** Let $H$ be the maximum initialHeight for the nodes, other than the root, present in the tree initially. Then, in the corresponding normal tree:

(i) There is exactly one $l$-active node, $1 \leq l \leq H$.

(ii) Apart from the root, every node on the right path is active.

(iii) Apart from the root, the ancestors of an active node are all active.

(iv) Let $v$ be an $l$-active node. Let $w$ be a $j$-active node. If $j < l$ then $w$ is to the right of $v$ in symmetric order, while if $j > l$ then $w$ is to the left of $v$ in symmetric order.

(v) Let inactive node $v$ have initialHeight $l$. Then its parent has initialHeight greater than $l$.

(vi) A node can become $l$-active only when it is on the right path.

When a lazy tree is created the active spans for the nodes in the corresponding lazy block tree are initialized as follows. Let $u$ be a node on the right path, of initialHeight $h$; suppose it has a right child $v$ of initialHeight $i$ (if there is no such node $v$ let $i = 0$). Then $u$ is given active span $[i + 1, h]$. Clearly, the new tree obeys Invariant 12.

Nodes are further categorized as black or white; an active node, with active span $[j, h]$, can be black with respect to each of the layers $[1, j - 1]$. In general, an active node $v$, with active span $[j, h]$, is black with respect to all the layers in some range $[i, j - 1]$, $i \geq 1$, called its black span; we say $v$ is $l$-black, for $i \leq l \leq j - 1$. If the black span is non-empty we say the node is black. Nodes are initially white at all layers. A node becomes black as a consequence of a rotation with the root. The following invariant applies to black nodes.

**Invariant 13** (i) All the nodes on the left path of a tree are fully black, i.e., a node with active span $[j, h]$ has black span $[1, j - 1]$. 

14
(ii) If a node \( u \) is \( l \)-black all \( u \)’s left ancestors, apart from the root, in the corresponding normal tree are \( l \)-black.

We define the following distances for each node \( v \), active at layer \( l \). Its right path \( l \)-distance, \( d_l(v) \), is the distance in vertices, traversed on the right path in going from \( v \) to the first \( l \)-black node, or to the root of the tree, whichever is nearer, excluding \( v \) and the first \( l \)-black node or the root, as appropriate. (A right edge is an edge from a node to its right child.) Its layer \( l \) interior right path distance, \( id_l(v) \), is the number of proper right ancestors of \( v \) below the first \( l \)-black node and below the right extreme path.

For each \( l \)-active node we maintain an \( l \)-potential, which satisfies the following invariant.

**Invariant 14** The \( l \)-potential at node \( v \) is at least

\[
a \cdot \min\{d_l(v)/d, l\} + a \cdot id_l(v)
\]

We note that initially a node has \( a \cdot l \) units of potential for each layer \( l \) at which it could become active, so when a node first becomes \( l \)-active Invariant 14 holds (see Invariant 12 (vi), also).

Immediately prior to a traversal of the right path, each node on that path is given \( s/2 \) spare rotations. Each node’s spares are subsequently provided by the rotation which involves that node. Suppose node \( v \) remains on the right path following the rotation. Let layer \( l \) be the largest layer at which \( v \) is presently active. Suppose node \( u \), the other node in \( v \)’s couple, is not \( l \)-black. If \( d_l(v) > d \cdot l \), then the spares, called the \( l \)-spares, for the nodes at depths \( (d(l - 1), d \cdot l] \) on the right path, are used to pay for \( v \)’s rotation. Otherwise, \( a/d \) units of \( v \)’s \( l \)-potential are used to pay for this rotation. Note that Invariants 12-14 continue to hold following this rotation.

Note that if node \( v \) is \( l \)-active, but is not on the right path, then its \( l \)-potential can increase, but only if \( d_l(v) > d \cdot l \). In this case, the unit increase in potential is paid for by the \( l \)-spares as in the previous paragraph.

For each \( l \)-black node we maintain an \( l \)-black potential of \( \frac{1}{2}a/d \) units. A rotation between an \( l \)-black node \( u \) and its child \( v \), where \( v \) is \( l \)-active, is paid for by the \( \frac{1}{2}a/d \) units of \( l \)-black potential at \( u \); \( u \) reduces its black span accordingly, as do all the nodes in \( u \)’s new right subtree (the subtree following the rotation). Again, it is clear Invariants 12, 13 hold following the rotation. To verify Invariant 14 we argue as follows. First, for \( j \)-active node \( w \), \( j < l \), in \( v \)’s old right subtree, \( d_j(w) \) is unchanged or reduced, since \( u \) ceases to be an ancestor of \( w \); in addition, \( id_j(w) \) is unchanged. Second, for \( j \)-active node \( w \), \( j > l \), in \( v \)’s old left subtree, \( d_j(w) + id_j(w) \) is unchanged since no node changes its \( j \)-black status; so the \( j \)-white potential of \( w \) is either unchanged or reduced. Third, for \( j \)-active node \( w \), \( j > l \), in \( u \)’s left subtree, \( d_j(w) \) and \( id_j(w) \) are unchanged since no node changes its \( j \)-black status. Reference to Invariant 12 (iv) shows that these are the only possible cases.

A black node is created in a rotation with the root (see Figure 6); we have already explained how to pay for this rotation. \( v \) becomes the new root. \( u \), the old root, acquires all of \( v \)’s potentials; \( v \) acquires the root’s global potential. If \( u \) now has minimum active layer \( l \) it acquires black span \([1, l - 1] \). The new portion of its black span is paid for by transferring \( \frac{1}{2}a/d \) potential from each of the lower layer active nodes whose distance to the root has been reduced by one, unless for some \( j \)-active node, \( w \), with \( j < l \), \( d_j(w) > j \), in which case node \( w' \) at depth \( j \) transfers a second portion of \( \frac{1}{2}a/d \) units of potential. Note that the layer \( j' \) of \( w' \) satisfies \( j' > j \) and so \( d_j(w') < j' \); thus the potential of \( w' \) is reduced by \( a/d \), owing to the
creation of the black node, and $w'$ can afford to transfer two sets of $\frac{1}{2}a/d$ units of potential to the new black node (one set for itself and one set for node $w$).

We summarize. The rotation of a couple (cost $s + 1 + ld + 2hd$ units) may be paid for by either $a/d$ units of $l$-white potential or by $\frac{1}{2}a/d$ units of $l$-black potential or by the $ds/2$ $l$-spares, where the bottom node of the couple is $l$-active. Alternatively, the $ds/2$ $l$-spares may have to provide $a/d$ units of white potential to the $l$-active node, if it is not traversed. So it suffices to have:

$$\frac{1}{2}a/d \geq s + 1 + ld + 2hd \quad \text{(12)}$$

$$ds/2 \geq \max\{(s + 1 + ld + 2hd), a/d\} \quad \text{(13)}$$

### 3.1.2 Left Path Traversal

Consider a couple, $w, v$, on the left path, with $v$ the parent of $w$; let $u$ be the parent of $v$ (see Figure 7). We have the same 5 cases as in the traversal of the right path. Cases 1-3 and 5 are handled as before. For Case 4 we proceed as follows (again, we are now concerned only with the truncated lazy block tree). If nodes $u$ and $v$ were on the right path, subtree $U$ (resp. $V$) would be the left subtree of $u$ (resp. $v$). We treat the rotation of couple $w, v$ on the left path as if it were the rotation of couple $u, v$ on the right path. So first we interchange $w$ and $v$'s potentials. Next, we pay for the rotation between $v$ and $w$ using $v$'s present $\frac{1}{2}a/d$ units of $l$-black potential, where $w$ is now $l$-active (note $w$'s present potentials correspond to those $u$ would have if on the right path). Invariants 12-14 continue to hold, as can be seen by arguments similar to those used for the right path traversal.

Notice that a left path traversal itself does not cause the spending of any spares, for all the case 4 rotations on the left path (which are the only rotations to spend spares) are paid for by black potentials.

### 3.1.3 Splitting the Lazy Tree

A split of the lazy tree occurs when the inserted item lies in value between the left guard and the right guard in the lazy tree. Since a split causes a zig-zag rotation within the lazy tree, and hence in the splay tree other that at the splay tree root, a split can occur only during a global insertion.

For this section, we assume that the open intervals spanned by the guards of each lazy tree are disjoint. In Section 3.2, we analyze the general case.

We view the split as occurring in three phases. In Phase 1 no rotations are performed, but certain nodes are marked as promoted (a node is promoted by increasing its lazy potential to its normal global potential). In fact the nodes are not promoted yet, but the splay will proceed as if they had been promoted. The effect of the promotions is to partition the original lazy tree into several new lazy trees. Phase 3 pays for these promotions. This ensures that in Phase 2, the actual splay, only extreme paths of lazy trees are traversed. However, there will be one difference in paying for Phase 2 as compared to the previous traversals. Any (apparently) promoted node, whose global rank drops during the splay does not use $gp$ units of its global potential to pay for the associated segment, for it does not yet have its global potential; instead paying for the segment becomes a charge to be paid for by the promotion (a charge of at most $gp$ units). It is called the segment charge; the segment charge is paid for directly by the promoted vertex. Phase 3 pays for the remaining costs of the promotions, at most $3gp\log n$ units.
Phase 3 uses the following imaginary tree. Consider performing all the zig-zag operations of the insertion but replacing each of the zig-zig operations by two single rotations performed in bottom to top order. This creates two paths, called access paths, descending from the inserted item, the root of the imaginary tree; one path, the left path, to its left, descends to the right, the other path, the right path, to its right, descends to the left. The items on the access paths are exactly the items that will be traversed in the splay operation. We provide each global node on the access paths with an imaginary global rank, namely the global rank it has in the imaginary tree. The global ranks of the other global nodes are the same in the imaginary tree and the actual tree. Each of the access paths is traversed from bottom to top; for each global node at which there is a jump in imaginary global rank the following potential is provided: for the left path, \( gp \) times the jump in rank, and for the right path, \( 2gp \) times the jump in rank.

We now discuss which nodes are promoted in a split and how this is paid for. There are a number of cases.

**Case 1.** Let \( v \) be a node, other than the root, in the lazy block tree. Suppose \( v \) is not on the left extreme path. Further suppose that \( v \) is accessed from its left child, \( u \). Then \( v \) is promoted. \( v \)'s promotion is paid for by \( gp \) times the jump in global rank from \( u' \), \( v \)'s left child in the imaginary tree, to \( v \) (for, by Lemma 4, \( v \)'s lazy weight is at least the weight of its left subtree in the actual tree and hence also in the imaginary tree; so \( v \)'s lazy rank is at least as large as \( u \)'s global rank). (Comment. The node \( u' \) may be missing; i.e., \( v \) is the bottom node on the right path.)

Let \( w_1 \) be \( v \)'s right child, if any, in the lazy block tree, and let \( w_2, w_3, \ldots, w_k \) be the maximal left path descending from \( w_1 \) in the lazy block tree. \( w_1, w_2, \ldots, w_k \) are also promoted. Between them, these promotions require just \( gp \) times the jump in global rank from \( u' \) to \( v \). For, by Lemma 4, \( w_i \)'s lazy rank is at least \( w_{i+1} \)'s normal rank, for \( 1 \leq i < k \). Finally, \( w_k \)'s lazy rank together with the jump in rank from \( u' \) to \( v \) is at least \( v \)'s normal rank (for the lazy weight of \( w_k \) includes the weight of \( v \)'s left subtree). The nodes \( w_i \) become the roots of new lazy trees. Let \( g_i \) be the right guard for \( w_i \) (the rightmost node in the subtree of the old lazy block tree rooted at \( w_i \)), for \( 1 \leq i < k \). \( g_i \) is promoted; this is paid for by its reserve potential (for note that \( g_i \) has an empty right subtree in the lazy block tree and see Lemma 4). Also \( g_{i+1} \) becomes the left guard for the new lazy tree tree rooted at \( w_i \), for \( 1 \leq i < k \); the tree rooted at \( w_k \) uses \( v \) as its left guard.

So Case 1 occasions charges of \( 2gp \) times the jump in rank to node \( v \), a node on the right path.

**Case 2.** A global node \( v \) on the left extreme path is accessed from its right child, \( w \). Then \( v \) is promoted. \( v \) becomes the root of the lazy block tree for a new lazy tree. \( gp \) times the jump in global rank from \( w' \), \( v \)'s right child in the imaginary tree, to \( v \) suffices to pay for this promotion. (Comment. Again, node \( w' \) may be missing.)

In addition, we need to provide a new left guard for the remaining portion of the old lazy tree. So let \( v' \) be \( v \)'s parent in the lazy block tree. Let \( w_1 \) be \( v' \)'s right child, if any, in the lazy block tree, and let \( w_2, w_3, \ldots, w_k \) be the maximal left path descending from \( w_1 \) in the lazy block tree. If \( v' \) is not the root of the lazy tree, then all of \( v', w_1, w_2, \ldots, w_k \) are promoted, while if \( v' \) is the root then only \( w_1, w_2, \ldots, w_k \) are promoted. For \( 1 \leq i \leq k \), \( w_i \) becomes the root of a new lazy tree. Let \( g_i \) be the right guard for \( w_i \) (the rightmost node in the subtree of the old lazy block tree rooted at \( w_i \)), for \( 1 \leq i \leq k \); each node \( g_i \) is promoted (as in Case 1, its reserve potential pays for the promotion). \( g_{i+1} \) is the left guard for the new lazy tree tree rooted at \( w_i \), for \( 1 \leq i < k \); the tree rooted at \( w_k \) uses \( v' \) as its left guard. \( v' \) becomes the root.
of an ordinary block. If \( v' \) is not the root of the lazy tree, \( g_1 \) becomes the left guard for the remainder of the old lazy tree, while if \( v' \) was the root, then \( g_1 \) did not need promoting as it was already the right guard of the lazy tree.

In the imaginary tree, \( v' \)'s lazy weight is at least its normal weight; so its promotion comes for free. As in Case 1, the jump in global rank from \( v' \)'s left child to \( v' \) suffices to raise \( w_k \)'s lazy rank to \( v' \)'s normal rank and hence to \( w_1 \)'s normal rank; the promotions of \( w_i \), for \( 1 < i \leq k \) are handled as before.

The promotions of Case 2 are also performed if just the left guard, \( l \), is separated from the remainder of the lazy tree (through being accessed from its right child), but \( v' \) is defined to be the leftmost node in the lazy block tree (note \( v' \) will be traversed in the access). This applies even if \( l \) is the root of the lazy tree.

Case 2 occasions a charge of \( gp \) times the jump in rank at \( v \), which is on the left path, and a charge of \( gp \) times the jump in rank at \( v' \), which is on the right path.

Case 3. Consider the new leftmost lazy tree created by the promotions of Cases 1 and 2. We may have to provide it with a new right guard. So let \( x \) be the rightmost global node in the leftmost new lazy tree. Note that \( x \) is traversed in the current access.

Case 3a. \( x \) is not the root of its new lazy tree. \( x \)'s reserve potential is used to raise its lazy potential to that of the next global node, \( y \), to its right in the (old) lazy tree (i.e., the modified lazy potential of \( x \) also includes the weight of \( y \)'s left subtree); it may be that \( y \) is the right guard for the old lazy tree in which case it does not presently carry a lazy potential (so here, by \( y \)'s lazy potential, we mean \( y \)'s lazy rank multiplied by \( gp \)). \( x \)'s modified lazy potential is at least as large as its normal potential in the imaginary tree so the promotion is already complete.

Case 3b. \( x \) is the root of its new lazy tree. Let \( w \) be \( x \)'s left child in its new lazy block tree (if \( x \) does not have such a child, then the lazy tree comprises only one node and can be treated as a normal block henceforth). \( w \) is promoted, becoming the root of a new lazy tree while \( x \) becomes its right guard.

Either \( x \) was already the root (Case 1) or its promotion has already been paid for (Case 2). \( w \)'s promotion is paid for as follows. \( w \) adds its reserve potential to its lazy potential; \( w \)'s modified lazy potential is at least as large as \( x \)'s normal potential in the imaginary tree. Hence \( w \)'s promotion is paid for by its reserve potential alone.

The promotions of Case 3 are performed even if only the right guard, \( r \), in the lazy tree is separated from the rest of the lazy tree (through being accessed from its left child). This applies even if \( r \) is the root of the lazy tree.

Case 3 occasions no charge.

The promotions between them cost at most \( 3gp \log n \), there being a charge of \( 2gp \log n \) to the right path and of \( gp \log n \) to the left path.

We show how to reestablish Invariants 5-11 by removing debits on the extreme paths of the new lazy trees.

Invariant 5. The path debit, if any, is removed from each promoted node. This is charged to the promoted node. Also, each local node that ceases to be an extreme path node, and hence also ceases to be on a lazy tree, pays for the removal of its path debit, if any.

Invariant 6. The lazy debit, if any, is removed from each local node whose block is no longer part of a lazy tree. This includes those local nodes whose block roots become guards for a new lazy tree. We call such blocks normal blocks. This is charged to the node itself.

Invariant 8. For each promoted node, the large path debits, if any, are removed from its parent and child (if any) on the extreme path. This is charged to the promoted node. (Note
that the reestablishment of Invariant 5 has already removed the large debit, if any, from itself.)

Invariant 9. The small path debit, if any, is removed from each extreme path node whose
parent is promoted. This is charged to the promoted node.

Invariant 10. For each promoted node \( u \), the path debit, if any, is removed from the corre-
sponding node \( w \), if present. This is charged to the promoted node.

Note that reestablishing Invariants 5 and 8-10 involves the removal of at most two large
path debits and one small path debit.

Invariant 11. We need to consider several cases.

Case 1. For all nodes that become a guard, their blocks become normal, as mentioned earlier,
and these blocks' local nodes are responsible for paying for the removal of their lazy debits
(for they are no longer lazy tree nodes).

Case 2. For each block that becomes the leftmost block in a new lazy tree we reduce the
potential of each local node from \( 2c' \) to \( c' \). This pays for the removal of lazy debits from these
nodes.

Case 3. We consider violating nodes in a new lazy tree. The only such nodes to violate
Invariant 11 must be on a new right extreme path. (For a new left extreme path is either a
new leftmost block, in which case its local nodes were dealt with in Case 2, or it is a tail of
the old left extreme path and so already satisfied the invariant.) There are two possibilities
for a local node \( x \) on a new right extreme path, which still violates Invariant 11(i).

Case 3.1. \( x \) is adjacent to a node \( y \) in an block, \( B_y \), where \( B_y \) either has become the leftmost
block in a new lazy tree or it has become a new normal block. The removal of \( x \)'s lazy debit
is charged to node \( y \).

Case 3.2. \( x \)'s right child is in the same lazy tree but is the root of a different block. Let \( z \)
be the root of \( x \)'s block. First, suppose that \( z \) had not been on the left extreme path of the
lazy tree prior to the split. Let \( y \) be the nearest right ancestor of \( x \). Then \( y \) is between \( x \) and
\( z \), for if not \( z \) would have been on the right path descending from \( z \), prior to the split, and
by Invariant 11, would not be carrying a lazy debit. So \( y \) is in a block, \( B_y \), where \( B_y \) either
has become the leftmost block in a new lazy tree or it has become a new normal block. In
addition, for each such node \( y \), there is at most one node \( z \) of Case 3.2, namely the first node,
on the right path descending from \( y \)'s left child, to have a global child. The removal of \( x \)'s
lazy debit is charged to node \( y \).

Second, suppose that \( z \) had been on the left extreme path of the lazy tree prior to the split.
If \( z \) now violates Invariant 11, \( z \) must be the promoted root of \( x \)'s new lazy tree. The removal
of \( x \)'s lazy debit is charged to \( z \).

Note that a global node may be charged for the removal of three lazy debits (as node \( y \))
or one lazy debit (as node \( z \)) but not both.

Each local lazy tree node in a block that becomes either normal or newly leftmost is charged
for the removal of at most four lazy debits (from itself and the nodes of Case 3) or one path
debit and three lazy debits (removed from the same nodes). This is charged to the node’s \( c' \)
potential. So it suffices to have:

\[
c' \geq \max\{4hd, ld\} = 4hd
\]

Each promoted global node is charged for the removal of at most three lazy debits (from the
nodes of Case 3), one small path debit and two large path debits; it may also have to pay a
segment charge. So it suffices to have:

\[
b \geq 3hd + 2ld + sd + gp
\]
We conclude this Section by showing how to reestablish Invariants 12-14. Following the promotions of Phase 1, Invariants 12 (i) and (ii) need not hold. So we create nodes active at new layers according to the following rule. Consider a new lazy tree. Let \( H \) be the maximum initial height of any node in the new tree, apart from its root (the initial heights are those defined with respect to the original tree; they are not redefined with respect to the new lazy tree). Suppose there is no \( l \)-active node for some \( l \leq H \). Then, in the corresponding normal tree, the lowest node \( v \) on the right path whose span includes \( l \) becomes \( l \)-active (these new active layers are then translated back into the lazy tree at hand). Below, we show that Invariants 12-14 hold once more (incidentally, this implicitly shows that the rule for creating new \( l \)-active nodes is well defined).

Clearly Invariant 14 still holds for \( d_l(v) \) does not increase. Next we consider Invariant 13. In each new lazy tree the black status of the nodes is unchanged. So Invariant 13 still holds.

Finally, we show Invariant 12 is reestablished by the creation of new active layers. First, we consider the situation prior to the creation of new active layers. Clearly, Invariant 12 (iii) and (v) still hold; (iv) holds likewise, since the symmetric order of the nodes in each subtree of each new lazy tree is unchanged, as is the relative ordering of the subtrees. We note that each new lazy tree has a span of active layers, possibly empty, of the form \((i,h)\), \(0 \leq i \leq h\), where \( h \) is the largest initial height (as provided initially) of any node in the new lazy tree, apart from the root. By Invariant 12 (v), (iii) and (iv), the right path in each new normal lazy tree, from top to bottom, comprises a sequence, possibly empty, of active nodes with decreasing initial heights. It is now readily seen that the rule for creating new active layers restores Invariant 12 (i)-(iv). Finally, it is evident Invariant 12 (vi) has been obeyed in the creation of new active layers.

### 3.2 Multiple Level Lazy Trees

Because a local access path may include nodes of a current lazy tree we may seek to make the root of a lazy tree a global node carrying a lazy potential in a new lazy tree. We therefore generalize the form of the lazy trees. Now, a "block" in a lazy tree may itself be another lazy tree.

In order to distinguish the ages of the different lazy trees, we number the blocks, in insertion order, by \( 1, 2, \ldots \). A lazy tree is labeled by the number of its corresponding creating block. When a lazy tree is split its parts keep the same label.

Let \( L \) be a lazy tree; its skeleton plus its lazy block tree comprise the nodes on \( L \). The nodes on \( L \) are also said to belong to \( L \). A node may be on an extreme path of several lazy trees. For each lazy tree to which a node belongs it carries a separate potential. However, a node may carry only one debit, as before.

Now, however, a new lazy tree, as well as including blocks may include old lazy trees. We define a new lazy tree, \( L_{\text{new}} \), to contain an old lazy tree, \( L_{\text{old}} \), if the root, \( r \), of \( L_{\text{old}} \) is on \( L_{\text{new}} \). We write \( L_{\text{old}} \subseteq L_{\text{new}} \). If there is no tree \( L_{\text{mid}} \) with \( L_{\text{old}} \subseteq L_{\text{mid}} \subseteq L_{\text{new}} \), then \( L_{\text{old}} \) is treated as a "block" of \( L_{\text{new}} \). The root of \( L_{\text{old}} \) is treated as the "root" of this "block"; the "root" of \( L_{\text{old}} \) is a "global" node on \( L_{\text{new}} \). All other nodes on \( L_{\text{old}} \) are "local" nodes of this "block", unless they are not even on \( L_{\text{new}} \). This matters when defining the potentials for nodes on \( L_{\text{new}} \). Suppose that \( L_{\text{old}} \) is contained in \( L_{\text{new}} \); then, apart perhaps for its root, any node on \( L_{\text{old}} \) that is also on \( L_{\text{new}} \) must be a "local" node on \( L_{\text{new}} \); in fact, only the root and nodes on an extreme path of \( L_{\text{old}} \) can be on \( L_{\text{new}} \), but it need not be the case that even all these nodes are on \( L_{\text{new}} \).
In addition, each guard of a lazy tree may be the root of another lazy tree, rather than being the root of a block. More precisely, suppose that \( g \) is the right (resp. left) guard for lazy tree \( L \) on creation of \( L \), and \( L_{old} \) is the newest lazy tree rooted at \( g \) at this time (\( L_{old} \) is older than \( L \)). Then the root of \( L_{old} \) remains the right (resp. left) guard for \( L \) until one or both of \( L \) and \( L_{old} \) are split. Note that the root of \( L_{old} \) may at some point become the root of a newer lazy tree, \( L_{new} \); however, the root of \( L_{new} \) does not take over the guard role for \( L \).

The following invariant characterizes the overlap of lazy trees.

**Invariant 15** (i) Let \( L_a \) and \( L_b \) be two lazy trees of the same age. Then the two open intervals defined, respectively, by the guards of \( L_a \) and of \( L_b \) are disjoint.

(ii) Let \( L_{old} \) be a lazy tree and let \( r \) be its root. Let \( L_{new} \) be another, newer, lazy tree. Then

(a) If \( r \) lies strictly between the guards of \( L_{new} \) then the guards of \( L_{old} \) lie between the guards of \( L_{new} \). In addition, let \( u \) and \( v \) be the “global” items in the large lazy block tree for \( L_{new} \) straddling \( r \) (\( r \) may or may not be a “global” item on \( L_{new} \)). Then \( L_{old} \) plus its guards lies between \( u \) and \( v \). (Recall that the large lazy block tree for lazy tree \( L \) comprises the lazy block tree for \( L \) plus the guards for \( L \).)

(b) If \( r \) is strictly outside the closed interval defined by the guards of \( L_{new} \) then the open intervals defined by the guards of \( L_{old} \) and \( L_{new} \), respectively, are disjoint.

(c) Suppose that \( r \) is the right (resp. left) guard for \( L_{new} \). Let \( d_{new} \) be the rightmost (resp. leftmost) “global” item of \( L_{new} \). Then the left (resp. right) guard of \( L_{old} \) is either equal to or to the right (resp. left) of \( d_{new} \).

Thus, in some sense, an older lazy tree is either contained in a newer lazy tree or is disjoint from it. Before giving the next invariant, a few definitions are helpful. Let \( L \) be a lazy tree and let \( u \) be a “global” node on \( L \) or a guard for \( L \). \( v \) is an \( L \)-neighbor of \( u \) if \( v \) is a “global” node on \( L \) or a guard for \( L \) and \( u \) and \( v \) enclose no other node “global” in \( L \). Let \( u \) be an ancestor of its \( L \)-neighbor \( v \); the \((u,v)\)-neighbor path comprises those items on the path from \( u \) to \( v \) in the splay tree which are in the range \((u,v)\), if \( u < v \), or \((v,u)\), if \( v < u \).

**Invariant 16** Let \( L \) be a lazy tree.

(i) Apart possibly for its root, all of \( L \)'s “global” nodes carry their lazy potential.

(ii) Each of \( L \)'s “local” global nodes is a “global” non-root node in some older lazy tree.

(iii) Let \( u \) and \( v \) be \( L \)-neighbors, with \( u \) the ancestor of \( v \). Let \( N \) denote the \((u,v)\)-neighbor path. If \( N \) includes a global node, there is a lazy tree \( L_{old} \), older than \( L \), rooted at \( u \), such that every node in \( N \) is on \( L_{old} \). Further suppose that \( u \) is a left (resp. right) ancestor of \( v \). Then the right (resp. left) guard for \( L_{old} \) is either \( v \) or a left (resp. right) descendant of \( v \).

By inspection plus induction, Invariants 15 and 16 are true on creation of a lazy tree \( L_{new} \); also, they remain true as the extreme paths of the lazy trees are traversed.

The newest tree \( L_{old} \) of Invariant 16(iii) is called the cover lazy tree for the \((u,v)\)-neighbor path; if \( u \) is the left (resp. right) guard for lazy tree \( L \), \( L_{old} \) is called the left (resp. right) guard cover tree for \( L \), and is denoted \( CL \). Also, if the left (resp. right) guard, \( u \), for lazy tree, \( L \), has no descendants in \( L \), the newest lazy tree, \( L_{old} \), older than \( L \), rooted at \( u \), if any, is called the left (resp. right) guard cover tree for \( L \).
Corollary 1 Using the notation of Invariant 16, the nodes of $N$ are all on an extreme path of the corresponding cover lazy tree.

Lemma 5 Let $L_1$ be a lazy tree and let $g$ be its right (resp. left) guard. Let $CL_1$ be $L_1$'s right (resp. left) guard cover tree. If any. Suppose that $g$ is on lazy tree $L_2$ also, where $L_2$ is newer than $CL_1$. Then $L_2$ is newer than $L_1$.

Proof. The lemma is immediate if $g$ has no descendants in $L_1$. So suppose that $g$ has a descendant in $L_1$. Without loss of generality, suppose that $g$ is the right guard for $L_1$. Let glob be the left $L_1$-neighbor of $g$. Suppose, for a contradiction, that $L_2$ is not newer than $L_1$. $L_1$ and $L_2$ satisfy Invariant 15. As $g$ is on $L_2$, they must satisfy Invariant 15(ii)c; so $g$ is the root of $L_2$. By Invariant 15(ii)a applied to $CL_1$ and $L_2$, the left guard, $g_{l_2}$, of $L_2$ is equal to or to the left of the left guard, $g_{cl_1}$, of $CL_1$. By Invariant 15(ii)c applied to $L_1$ and $L_2$, $g_{l_2}$ is equal to or to the right of glob. So $g_{l_2}$ is in the range $[gob,g_{cl_1}]$. By Invariant 16, $g_{cl_1}$ is either glob or a right descendant of glob. Thus $g_{l_2}$ is also either glob or a right descendant of glob. Since each node $w$ in the range $(gob,g)$ on the path from $g$ to glob is on the left extreme path of $CL_1$, and is also on the path from $g$ to $g_{l_2}$, each such node $w$ is also on the left extreme path of $L_2$. So $L_2$, rather than $CL_1$, would provide the left cover guard tree for $L_1$, contradicting the definition of $CL_1$. Thus $L_2$ is newer than $L_1$. •

Lemma 6 Let $v$ be a non-root node on lazy tree $L$. If $v$ is a “local” node on $L$, suppose that it is a right (resp. left) descendant of its nearest ancestor in its “block” on $L$; while if $v$ is a “global” node on $L$, suppose that it is the right (resp. left) child of its parent in $L$. Also, suppose that $v$ is on lazy tree $L_{old}$, older than $L$. Then $v$ is either the root of $L_{old}$ or is on the right (resp. left) extreme path of $L_{old}$.

Proof. The proof uses an induction on the age of $L$. If $v$ is “global” in $L$ then it is the root of $L_{old}$. So suppose that $v$ is “local” in $L$. Let $u$ be the nearest left (resp. right) ancestor of $v$, “global” in $L$. Let $w$ be the leftmost (resp. rightmost) “global” node in $L$ to the right (resp. left) of $u$. Consider the $(u,w)$-neighbor path in $L$; it includes $v$. Apply Invariant 16(iii) to this path to obtain the cover lazy tree $L_1$. If $L_1 = L_{old}$, by Corollary 1, we are done. If not, the result follows by induction on the triple $L_1, L_{old}$ and $v$ ($L_1$ is replacing $L$). •

Invariants 5-11 should be interpreted with respect to the newest lazy tree to which the node carrying the debit belongs.

We need to reconsider the analysis of global accesses (the first access for each block) for the form of the lazy trees has become more involved. As before, the traversals are classified into three types, respectively, those that traverse a right extreme path, a left extreme path, and splits. A split of lazy tree $L$ is an access that lies strictly between the guards of $L$. By means of splits we guarantee that in each lazy tree encountered in a traversal, only extreme paths are traversed.

The analysis of a right extreme path traversal proceeds essentially as before. Each couple is paid for as before, at the level of the newest lazy tree to whose extreme path it belongs. In addition, for each traversed node, $v$, we maintain $v$'s potentials with respect to each lazy tree to which $v$ belongs. Maintaining the white and black potentials might appear problematic, for it may involve the spending of $k$-spares, $k \geq 1$. But we note that for each global node, $v$, there is only one lazy tree with respect to which $v$ carries white and/or black potentials (for in all newer lazy trees that contain $v$, $v$ is “local”). So there is only one lazy tree with whose white and black potentials $v$ is concerned, and this is the only lazy tree to which $v$ contributes.
spares or from which \( v \) draws spares. So the white and black potentials can be maintained as before. Otherwise, the analysis of a right path traversal is unchanged. The same is true for a left path traversal.

Finally, we explain how to perform a split. For each split lazy tree, our goal in a split is to promote the same nodes as in Section 3.1.3; however, this may prove too expensive because of the recursive containment of lazy trees. So, sometimes, instead of promoting a node \( v \), “global” on lazy tree \( L \), we will promote a node \( w \) on the \((u, v)\)-neighbor path in \( L \), where \( w \) is global. The details of this process follow; they are quite intricate.

A promotable node on lazy tree \( L \) is a “global” node of \( L \) other than the root. Let \( v \) be a node on lazy tree \( L \). Suppose \( v \) is accessed from its child \( u \). The traversal of node \( v \) is a promoting traversal if \( v \) is a “global” node on \( L \) and either:

(i) \( v \) is a promotable node not on the left extreme path of \( L \) and \( u \) is \( v \)'s left child. Or,

(ii) \( v \) is a promotable node on the left extreme path of \( L \) and \( u \) is \( v \)'s right child.

Let \( v \) be a traversed global node. If in some lazy tree to which \( v \) belongs the traversal is promoting, then \( v \) is promoted. The provision of \( gp \log n \) potential to the access suffices to pay for these promotions. For from Lemma 4, we conclude that for each such promoted node \( v \), its lazy rank, \( lazyrank(v) \), satisfies \( lazyrank(v) \geq \lfloor \log wt(u) \rfloor \) (note that to apply Lemma 4 to the left extreme path the normal form of the lazy tree should be considered); so \( gp \) times the jump in rank from \( u \) to \( v \) suffices to pay for the increase of \( v \)'s lazy potential to \( gp \cdot rank(v) \), its normal potential.

Remark 1 \( v \)'s traversal is promoting in at most one lazy tree.

We promote the following additional nodes on the split paths. Let \( r \) be the root of \( L \).

Case a. The access is to the right of \( r \). Let \( v \) be the following “global” node on \( L \), if present: the rightmost “global” node to the left of the accessed item. \( v \) is promoted if it is traversed. The promotion is paid for as follows: \( v \)'s reserve potential is added to its lazy potential; this is at least as large as its normal potential in the imaginary tree (see Lemma 4).

Case b. The access is to the left of \( r \). Let \( v \) be the following “global” node on \( L \), if present: the lowest “global” node on the left extreme path of \( L \) to the right of the accessed item. \( v \) is promoted if it is traversed. We note that \( v \)'s lazy potential is already its normal potential in the imaginary tree. Let \( u \) be the following “global” node on \( L \), if present: the rightmost “global” node to the left of the accessed item, if it is not on the left extreme path. \( u \) is promoted if it is traversed. \( u \)'s promotion is paid for as follows: \( u \)'s reserve potential is added to its lazy potential; this is at least as large as its normal potential in the imaginary tree.

There is no further charge for the promotions of Cases a and b. No other traversed nodes are promoted.

The promoted nodes on lazy tree \( L \) (apart from node \( v \) of Case (a) or node \( u \) of Case (b)) are all to become the roots of new lazy trees formed from \( L \). However, we will create further new lazy trees by promoting additional nodes; on top of this, we have yet to provide guards to the new lazy trees, which may entail further promotions.

A few more definitions are helpful. A vertex \( v \) is a true split point for lazy tree \( L \) if \( L \) is split and at least one of the following is true:

(a) \( v \) is a promotable node on \( L \) and \( v \)'s traversal is promoting.
(b) $v$ is the lowest traversed node on the left extreme path of $L$ to be accessed from its right child; also, $v$ is not "global" on $L$.

(c) $v$ is the lowest node on $L$, not on the left extreme path of $L$, which is accessed from its left child and is a right descendant of its nearest ancestor in its "block"; in addition, $v$ is not "global" on $L$.

(d) $v$ is the root of $L$, and if to the right (resp. left) of the accessed item, is the leftmost (resp. rightmost) "global" item in the portion of $L$ to the right (resp. left) of the accessed item.

(e) $v$ is the vertex $v$ of Case (b) for $L$.

If $v$ of (a) is on the left extreme path of its lazy tree, and there is a corresponding node $u$ in Case (b) above, $v$ is said to be a special split point.

Next, we define guard split points for lazy tree $L$.

(a) Let $g$ be the right (resp. left) guard for $L$; let $glob$ be the leftmost (resp. rightmost) "global" item in $L$; suppose that the access is in the range $(g, glob)$ (resp. $(glob, g)$). Suppose that $glob$ is not traversed (so $g$ is an ancestor of $glob$). Let $CL$ be the guard cover tree for $L$, rooted at $g$ (if there is no such lazy tree $CL$ we intend the block rooted at $g$). Then $v$, the lowest traversed node in $CL$ on the path from $g$ to $glob$ is a direct guard split point for $L$.

(b) If $v$ is a special split point, the corresponding node $u$ of Case (b) of the promotions is a virtual guard split point. Node $v$ of Case (a) of the promotions is also a virtual guard split point.

(c) $v$ is an indirect guard split point if $v$ is a true split point for an older lazy tree $L_{old}$, $v$ is a guard for $L$, and $v$ has a descendant in $L$.

**Lemma 7** If $v$ is an indirect guard split point for lazy tree $L$, then it is a true split point for the corresponding guard cover tree, $CL$.

**Proof.** Suppose that $v$ is on the right (resp. left) split path. Let $L_{old}$ be the tree for which $v$ is a true or direct guard split point. Suppose that $CL \neq L_{old}$ (otherwise the result is immediate). Then, by Invariant 15(ii)a applied to $CL$ and $L_{old}$, $L_{old}$ is to the right (resp. left) of $u$, the left (resp. right) $CL$-neighbor of $v$ (for by Lemma 5, $CL$ is newer than $L_{old}$). Since $L_{old}$ is split, $v$ is a type (d) true split point for $CL$. □

**Lemma 8** Let $v$ be the root of lazy tree $L$. Suppose that $L$ is split and $v$ is traversed. Then following the above promotions, $v$ carries its normal potential.

**Proof.** Suppose, for a contradiction, that $v$ is the root of $L$ but does not carry its normal potential. Then $v$ must be a "global" non-root node on a newer lazy tree, $L_{new}$ (since it does not have its normal potential). In $L_{new}$ the traversal of $v$ was not promoting, by assumption; nonetheless, $L_{new}$ must have been split (for by Invariant 15 applied to $L$ and $L_{new}$, the open interval spanned by $L_{new}$'s guards includes the corresponding interval for $L$, which in turn contains the inserted item).

Suppose that $v$ is accessed from its left (resp. right) child. We consider various cases.

**Case 1.** $v$ is on the left (resp. right) extreme path of $L_{new}$. From Invariant 15(ii)a, since $L$ ($=L_{old}$) is split, $v$ must be the leftmost (resp. rightmost) "global" item of $L_{new}$ to the right
(resp. left) of the accessed item. So \( v \) is promoted in Case (b) (resp. in Case (a)) for \( L_{\text{new}} \). So Case 1 does not arise.

**Case 2.** \( v \) is on the right (resp. left) extreme path of \( L_{\text{new}} \). Then the access to \( v \) is promoting in \( L_{\text{new}} \). So Case 2 does not arise.

**Case 3.** \( v \) is not on an extreme path of \( L_{\text{new}} \). If \( v \) is accessed from its left child the access to \( v \) is promoting in \( L_{\text{new}} \). So suppose that \( v \) is accessed from its right child. As in Case 1, \( v \) must be the leftmost “global” item of \( L_{\text{new}} \) to the right of the inserted item. Then, according as the access is to the right or left of the root of \( L_{\text{new}} \), \( v \) is promoted in Case (a) or (as \( u \) in) Case (b) for \( L_{\text{new}} \). So Case 3 does not arise.

But this exhausts the possibilities. The lemma follows. •

**Lemma 9** Let \( v \) be a split point for lazy tree \( L \). Following the above promotions, if \( v \) is global, \( v \) now carries its normal potential.

**Proof.** Suppose that \( v \) is a global node on the right (resp. left) split path. Now suppose, for a contradiction, that \( v \) does not carry its normal potential, at present. If \( v \) is a direct guard split point or an indirect guard split point then it is also a split point for an older lazy tree \( L' \). (That a direct guard split point is a split point for some tree \( L' \) can be seen as follows: By Invariant 16(ii), \( v \) is a “global” non-root node in some lazy tree \( L' \). By Corollary 1, \( v \) is on the right (resp. left) extreme path of \( CL \), the right (resp. left) guard cover tree for \( L \), and by Lemma 6, \( v \) must be on the right (resp. left) extreme path of \( L' \). But \( v \) is the first node accessed on the right (resp. left) path of \( L' \) and it is “global” on \( L' \); hence its traversal is promoting and so \( v \) is a true split point for \( L' \).) Then, by an inductive argument, we can suppose that \( v \) has been promoted. For \( v \) a true split point we argue as follows: (i) If \( v \) is of type (a) or (e) it was promoted. (ii) If \( v \) is of type (d), by Lemma 8, \( v \) now carries its normal potential. (iii) If \( v \) is of type (b) or (c), \( v \) is a “local” node on \( L \) and yet it is global; so \( v \) must be a “global” non-root node on some older lazy tree \( L_{\text{old}} \) (by Invariant 16); but \( v \) is the first vertex on the right (resp. left) extreme path of \( L_{\text{old}} \) to be traversed. (For, if \( v \) is on the right split path, \( v \) is of type (c), and so \( v \) is a right descendant of the nearest ancestor in its “block” in \( L \); by Lemma 6, \( v \) is on the right extreme path of \( L_{\text{old}} \). While if \( v \) is on the left split path, \( v \) is of type (b), and so \( v \) is a left descendant of the nearest ancestor in \( L \), which is in the same “block” as \( v \); again, by Lemma 6, \( v \) is on the left extreme path of \( L_{\text{old}} \).) As \( v \) is a “global” non-root node on \( L_{\text{old}} \), \( v \)’s traversal is promoting with respect to \( L_{\text{old}} \). So if \( v \) is a true split point, it carries its normal potential following the split. Finally, if \( v \) is a virtual guard split point, it was promoted and so carries its normal potential. •

**Lemma 10** Let \( L_1 \) and \( L_2 \) be two lazy trees both having split point \( v \) on the right (resp. left) split path. Let \( g_1 \) be a “global” node on \( L_1 \) and \( g_2 \) be a “global” node on \( L_2 \), both in \( v \)’s right (resp. left) subtree. Suppose that \( g_1 \) is a proper ancestor of \( g_2 \). If \( L_1 \) is older than \( L_2 \), then \( g_1 \) is a left (resp. right) ancestor of \( g_2 \), while if \( L_1 \) is newer than \( L_2 \) and \( g_1 \) is not the root of \( L_2 \), then \( g_1 \) is a right (resp. left) ancestor of \( g_2 \).

**Proof.** Since both \( L_1 \) and \( L_2 \) have left (resp. right) guard equal to or to the left (resp. right) of \( v \) and right (resp. left) guard to the right (resp. left) of \( v \), one of Invariant 15(ii)a or (ii)c is satisfied by \( L_1 \) and \( L_2 \). But now the result is immediate. •

**Lemma 11** Let \( L_1 \) and \( L_2 \) be two lazy trees both having split point \( v \) on the right (resp. left) split path. Suppose that the root, \( r_1 \), of \( L_1 \) is in \( v \)’s right (resp. left) subtree. Suppose further
that \( r_1 \) carries its normal potential. Then the root, \( r_2 \), of \( L_2 \) is not in \( r_1 \)'s right (resp. left) subtree.

**Proof.** Suppose, for a contradiction, that \( r_2 \) is in \( r_1 \)'s right (resp. left) subtree. Let \( g_2 \) be the left (resp. right) guard for \( L_2 \); since \( v \) is a split point for \( L_2 \), \( g_2 \) is either \( v \) or a left (resp. right) ancestor of \( v \). Let \( glob_2 \) be the leftmost global node in \( L_2 \). Then \( r_2 \) is on the path from \( g_2 \) to \( glob_2 \). So \( r_1 \) is on the path from \( g_2 \) to \( glob_2 \), and in the range \((g_2, glob_2) \) (resp. \((glob_2, g_2) \)).

Now apply Invariant 16(iii) to the nodes \( g_2 \) and \( glob_2 \) and tree \( L_2 \). This yields a lazy tree \( L_{\text{old}} \), rooted at \( g_2 \), to which all the nodes on the path from \( g_2 \) to \( glob_2 \) in the range \((g_2, glob_2) \) (resp. \((glob_2, g_2) \)) belong; this includes \( r_1 \). But this implies \( r_1 \) does not carry its normal potential (by Invariant 16).

**Lemma 12** Let \( L_1 \) be a lazy tree for which \( v \) is a split point on the right (resp. left) split path. Let \( L_2 \) be another, newer lazy tree, to which \( v \) belongs. If \( v \) is a split point for \( L_2 \) also, then \( L_2 \) has no "global" nodes both to the left (resp. right) of \( v \) and to the right (resp. left) of the accessed item.

**Proof.** If \( v \) is a guard split point for \( L_2 \), the result is immediate. Otherwise, we start by defining a tree \( L_3 \), older than \( L_2 \), where \( L_3 \) is split by the access (i.e., the access is strictly between \( L_3 \)'s guards). If \( v \) is an indirect guard split point for \( L_1 \), then \( CL_1 \), the left (resp. right) guard cover tree for \( L_1 \) is split by the access. In addition, \( CL_1 \) is older than \( L_1 \) and hence older than \( L_2 \). Let \( L_3 \) denote \( CL_1 \) is this case; otherwise, let \( L_3 = L_1 \). \( L_3 \) is split by the access. Let \( u \) and \( w \) be the two "global" nodes or guards of \( L_2 \) straddling \( v \). By Invariant 15, \( L_3 \) and its guards are enclosed by \( u \) and \( w \). As \( L_3 \) is split by the access, \( u \) is to the left (resp. right) of the accessed item. The result follows.

**Lemma 13** Let \( v \) be the root of split lazy tree \( L \), where \( v \) is not a split point for \( L \). Then every tree for which \( v \) is a split point is newer than \( v \).

**Proof.** Suppose that \( v \) is on the right (resp. left) split path. For any lazy tree, \( L_1 \), for which \( v \) is a split point, either \( L_1 \) is split and \( v \) is on \( L_1 \), or the left (resp. right) cover guard tree for \( L_1 \), \( CL_1 \), is split, in which case \( v \) is on \( CL_1 \). Let \( L_2 \) denote this split tree. \( v \) is on \( L_2 \). \( L_2 \) is no newer than \( L_1 \). If \( L_2 \) were newer than \( L_2 \), then, from Invariant 15(ii)a, we conclude that \( v \) would be the leftmost (resp. rightmost) "global" item in \( L \) to the right (resp. left) of the accessed item; but then \( v \) would be a split point for \( L \).

**Lemma 14** Suppose \( v \) is the split point of Case (b) for \( L \). Then every other tree for which \( v \) is a split point is older than \( L \).

**Proof.** \( v \) is on the right split path, so must be on the left extreme path for \( L \). By Lemma 6, on any newer lazy tree, \( L_{\text{new}} \), \( v \) must be a "local" node which is a left child of its parent in the lazy tree. By inspection, \( v \) is not a split point for \( L_{\text{new}} \).

**Lemma 15** Let \( u \) be a virtual guard split point for lazy tree \( L \). Let \( v \) be the corresponding special split point, if any. Then

(i) Any other lazy tree which has \( u \) as a split point is older than \( L \).

(ii) Any other lazy tree which has \( v \) as a split point is newer than \( L \).
Proof. Suppose $L_{new}$ is a lazy tree, newer than $L$, for which $u$ is a split point. If $u$ is on $L_{new}$, then by Lemma 6, $u$ is on an extreme path of $L$, which is not the case. The only other possibilities are that $u$ is a direct or indirect guard split point for $L_{new}$, but here again, $u$ must be the root or on the extreme path of the cover guard tree for $L_{new}$, and hence on the extreme path of $L$.

Next, suppose $L_{old}$ is an lazy tree, older than $L$, for which $v$ is a split point. $v$ is on the left split path. If $L_{old}$ is split, from Invariant 15(ii) we conclude $u$ must be the root of $L_{old}$ (for the access is to the right of $u$). But then $v$ is either the left guard of $L_{old}$ or to the left of this guard. So $v$ could only be an indirect guard split point for $L_{old}$; this happens only if $v$ is a true split point for a yet older lazy tree $L_{old'}$, which we have just argued cannot happen. ●

In order to understand the effects of the current promotions we examine the split paths in the imaginary tree. In the remainder of this section, we denote by $L'$ a new lazy tree created from $L$, using further indexing to distinguish among the trees $L'$, if need be: $L'$ is called an $L$-derived lazy tree, or $L$-derived tree, for short. Let $v$ be a global node on the right (resp. left) split path.

Comment. The cases with $v$ on the left and right split paths are very similar, although not identical.

Suppose $v$ is a global node which is a split point. By Lemma 9, $v$ now carries its normal potential. Suppose $v$ is on the right (resp. left) split path. Consider $v$'s right (resp. left) subtree in the splay tree. Suppose that $v$ is a split point, other than a special split point, for lazy trees $L_{old'} = L_1, L_2, \ldots, L_j = L_{new}$, where $i_1 < i_2 < \cdots < i_j$, and where these trees have a “global” node in $v$'s right subtree. Let $L_{old} = L_1, L_2, \ldots, L_{i_1} = L_{new}$, where $i_1 < i_2 < \cdots < i_j$, be the subset of these trees for which $v$ is a true split point. For trees $L_{old}, \ldots, L_{i_j}$, $v$ is on the right (resp. left) extreme path or is the root. (For by Lemma 6, $v$ is on an extreme path or is the root. So suppose that $v$ is on the left (resp. right) extreme path for some tree $L_a$ among $L_{old}, \ldots, L_{i_j}$. Then $v$ is a true split point for $L_a$, of type (e); so $v$ is “global” on $L_a$. But then $v$ would be “local” on any newer lazy tree, $L_b$. By Lemma 6, $v$ must be a left (resp. right) descendant of its nearest ancestor in the same “block” in $L_b$. By inspection, $v$ is not a split point for $L_b$. But $v$ must be a split point for $L_b = L_{new}$. This contradiction shows that $v$ is not on the left (resp. right) extreme path of $L_a$.)

We define a staircase path, $ST_v$, along which further promotions will occur. $ST_v$ is a path in $v$'s right (resp. left) subtree. Consider the right (resp. left) path in $L_{new}$ descending from $v$; let $w$ be the first node on this path that is “global” in one of $L_{old}, \ldots, L_{new}$ (see Figure 8). $w$ is promoted. $w$ is called the staircase point for $v$. Suppose that the oldest lazy tree to which $w$ belongs is $L_{i_j}$; denote $w$ by $w_{j_1}$. Let $w_{j_1}'$ be the leftmost (resp. rightmost) “global” node on $L_{i_j}$, descendant from $w_{j_1}$, and let $w_{j_1}''$ be the left (resp. right) child of $w_{j_1}'$. We continue the above process (selecting a point $w$) with respect to $w_{j_1}''$ and the trees $L_{old}, \ldots, L_{i_{j_1-1}}$. Eventually, suppose that a sequence of points $w_{j_1}, \ldots, w_{j_k}$ is created. We define the path $ST_v$ to comprise the nodes on the path from $v$ to $w_{j_k}$, excluding $v$, which are on one (or more) of $L_{old}, \ldots, L_{new}$.

For each tree $L_i$, for $1 \leq h \leq j'$, we define its pseudo root, $r_{i'h}$, to be the topmost “global” node of $L_{i'h}$ in $v$'s right subtree (by assumption, there is such a node); we then define the intersection point with $ST_v$, $e_{h}$, as follows: If the pseudo root, $r_{i'h}$, of $L_{i'h}$ is on $ST_v$ then $e_h = r_{i'h}$. Otherwise, $e_h$ is the promoted node $w_{j_k}$, where $L_{i_{j_k}}$ is the newest tree at least as old as $L_{i'h}$ to have a promoted node on $ST_v$, if any. It need not be the case that the intersection point exists.
Lemma 16 For each $1 \leq h \leq j'$,

(i) Suppose that $e_h$ exists and is not on $L_{i_h}$. Then the “global” nodes on $L_{i_h}$ to the right (resp. left) of $v$ are proper right (resp. left) descendants of $e_h$.

(ii) Suppose that $e_h$ exists and is on $L_{i_h}$. For $h' < h$, if $e_h$ is not on $L_{i_{h'}}$ and if $e_{h'}$ exists, the “global” nodes on $L_{i_{h'}}$ to the right (resp. left) of $v$ are proper left (resp. right) descendants of $e_h$.

(iii) The portion of $ST_v$ descendant from $e_h$ is contained in $e_h$’s left (resp. right) subtree, for $1 \leq h < j'$.

(iv) If $e_h$ does not exist then $L_{i_h}$ is older than $L_{old}$.

Proof. We consider three types of $h$; type (i), those $h$ for which $v$ is a true split point of $L_h$; type (ii), those $h$ for which $v$ is a guard split point and $e_h$ exists; type (iii), those $h$ for which $v$ is a guard split point and $e_h$ does not exist.

First, we prove the lemma restricted to type (i) $h$. Let $h' < h$ and consider $L_{i_h'}$ and $L_{i_{h'}}$, where $h$ and $h'$ are type (i). We note that the first node, $x$, if any, on the path from $v$ to $r_h'$ which does not belong to $L_{i_{h'}}$ must be $l$-global. As $x$ is on $L_{i_{h'}}$, it must be “global”, but not a root, on some $l$-lazy tree, $L$, newer than $L_{i_{h'}}$ but no newer than $L_{i_h}$. As $L$ is newer than $L_{i_{h'}}$, its root is $v$ or to the left (resp. right) of $v$, by Invariant 15 applied to $L$ and $L_{i_{h'}}$. Further, if $L \neq L_{i_{h'}}$, $v$ must either be the root of $L$ or be on the right (resp. left) extreme path of $L$ (by Lemma 6) and hence $v$ is a true split point for $L$ (since $L_{i_{h'}}$ is split), so $L$ is one of $L_{old}, \ldots, L_{new}$. We conclude that $ST_v$ cannot branch right (resp. left) at $x$, for if it reaches $x$, $x$ will be promoted. It follows that the portion of $ST_v$ from $v$ to the staircase point, excluding the staircase point itself, is on all of $L_{old}, \ldots, L_{new}$. Suppose that the staircase point is on $L_{i_{a+1}}, \ldots, L_{new}$; then the continuation of $ST_v$, beyond $L_{i_{a+1}}$, is on $L_{old}, \ldots, L_{i_{a}}$ (for, by assumption, there is a “global” node on the right, resp. left, extreme path of the subtree of each of these lazy trees contained in $v$’s right, resp. left, subtree). Also, the staircase point is $r_{a+1} = e_{a+1} = \cdots = e_{new}$. Performing the argument inductively, we conclude that $e_h$ exists and is on $L_{i_{h'}}$ for each type (i) $h$. (ii) for type (i) $h$ and $h'$ follows by induction on $j' - h$ and by Lemma 10.

We extend (ii) for type (i) $h$ to include type (ii) $h' < h$. We use the following claim (shown later in the proof).

Claim. Let $h$ be type (i) or (ii) and $h' < h$ be type (ii) or (iii). Then $r_{h'}$ is not a proper ancestor of $e_h$.

By the claim, $r_{h'}$ is a descendant of $e_h$, for $h' < h$, and (ii) for type (i) $h$ follows by Lemma 10.

Proof of Claim. Suppose, for a contradiction, that $r_{h'}$ is a proper ancestor of $e_h$. As $h' < h$, by Invariant 15, applied to $L_{i_{h'}}$ and $L_{i_{h'}}$, $r_{h'}$ is a proper left (resp. right) ancestor of $e_h$. Again by Invariant 15, applied to $L_{i_{h'}}$ and $L_{i_{h'}}$, $r_{h'}$ is the root of $L_{i_{h'}}$ (for otherwise two “global” nodes on $L_{i_{h'}}$ would straddle a “global” node on $L_{i_{h'}}$). By Invariant 16, applied to $L_{i_{h'}}$, $r_{h'}$ does not carry its normal potential prior to the split. So $r_{h'}$ must be “global” on some $l$-lazy tree, $L$, older than $L_{i_{h'}}$, and newer than $L_{i_{h'}}$. We show that $v$ is a true split point for $L$. We note that the root of $L$ must be $v$ or to the left (resp. right) of $v$ (by Invariant 15 applied to $L_{i_{h'}}$ and $L$). Furthermore, $v$ must be the root or on the right (resp. left) extreme path of $L$, as $L$ is older than $L_{i_{h'}}$ (by Lemma 6). We have three cases to consider, according to $v$’s type as
a split point for \( L_{i_h'} \). First, we observe that \( v \) is not a virtual guard split point for \( L_{i_h'} \) (for then, by Lemma 15, there would not be a newer tree \( L_{i_h'} \) with split point \( v \)). Second, if \( v \) is an indirect guard split point for \( L_{i_h'} \), then the left (resp. right) guard cover tree, \( CL_{i_h'} \), for \( L_{i_h'} \) is split; \( CL_{i_h'} \) is rooted at \( v \); as \( CL_{i_h'} \) is split and is older than \( L \), \( v \) must be a true split point for \( L \). Third, suppose that \( v \) is a direct guard split point for \( L_{i_h'} \). Then \( v \) is on \( CL_{i_h'} \).

As \( L \) is newer than \( CL_{i_h'} \), the root of \( L \) must be to the left (resp. right) of \( v \) and hence \( v \) is a true split point for \( L \). So in any event \( v \) is a true split point for \( L \). But then \( L \) is one of the trees \( L_{old}, \ldots, L_{new} \), and so \( r_{h'} \) would have been promoted, which contradicts \( e_h \) being a proper right (resp. left) descendant of \( r_{h'} \).

Next, we prove the lemma for type (ii) \( h \).

**Case 1.** Suppose that the path from \( v \) to \( r_{h'} \) has a global node in the range \((v, r_{h'})\) belonging to one of \( L_{old}, \ldots, L_{new} \). Let \( r_{h''} \) on \( L_{i_{h''}} \) be the topmost such node. If \( ST_v \) includes \( r_{h''} \) it was promoted. Then \( e_h = r_{h''} \) and (i) is immediate for \( h \). If \( ST_v \) does not include \( r_{h''} \) then either the bottom of \( ST_v \) is in a subtree to the left (resp. right) of \( r_{h''} \) (but then consider the node \( x \), the lowest node common to \( ST_v \) and the path from \( v \) to \( r_{h''} \); as \( ST_v \) contains left, resp. right, at \( x \), \( x \) is promoted; this contradicts the definition of \( r_{h''} \)). Otherwise, the bottom of \( ST_v \) is in a subtree to the right (resp. left) of \( r_{h''} \). But then the path \( ST_v \) and the path from \( v \) to \( r_{h''} \) diverge at a node, \( x \), “global” on a tree, \( L \), newer than \( L_{i_{h''}} \), yet older than \( L_{old} \) (for \( x \) is not promoted). But this is a contradiction as \( L_{i_{h''}} \) is no older than \( L_{old} \).

**Case 2.** Suppose that the path from \( v \) to \( r_{h'} \) does not have a “global” node from any of \( L_{old}, \ldots, L_{new} \) in the range \((v, r_{h'})\).

**Case 2.1.** Suppose that \( ST_v \) reaches \( r_{h'} \). Then \( e_h = r_{h'} \). By construction, \( r_{h''} \), for type (i) \( h' \), \( h' < h \), is a descendant of \( r_{h'} \), and (ii) follows for such \( h' \) by Lemma 10. For type (ii) \( h' < h \), \( r_{h''} \) is also a descendant of \( r_{h'} \) by the claim. Now, (ii) follows for type (ii) \( h' \) by Lemma 10.

**Case 2.2.** Suppose that \( ST_v \) does not reach \( r_{h'} \). Then \( ST_v \) must diverge from the path to \( r_{h'} \) at some node \( x \). The divergence cannot be a branch to the left (resp. right), for \( x \) would be a promoted node which meets the conditions of Case 1. (\( x \) is “global” on a lazy tree, \( L \), older than \( L_{i_{h'}} \). If \( x \) is not promoted, then \( L \) is newer than \( L_{new} \), so \( L \) is split. As \( L \) is older than \( L_{i_{h'}} \), by Invariant 15 applied to \( L \) and \( L_{i_{h'}} \), \( v \) must either be the root of \( L \) or be on the right (resp. left) path descending from the root of \( L \), for otherwise two global nodes of \( L \) would straddle a global node of \( L_{i_{h'}} \); so \( v \) is a true split point for \( L \). So, in fact, \( L \) is one of \( L_{old}, \ldots, L_{new} \), and \( x \) is promoted, as claimed.) If the divergence were a branch to the right (resp. left), it would be at a global node, which, since it is not promoted, cannot be “global” on any of \( L_{old}, \ldots, L_{new} \). So it must be “global” on an older lazy tree. But \( L_{i_{h'}} \) is yet older and so would not be of type (ii). So Case 2.2 does not arise.

Now we prove (iii). If \( e_h \) is on \( L_{i_{h'}} \), (iii) follows from (ii) immediately. If \( e_h \) is not on \( L_{i_{h'}} \), it is on some older tree \( L_{i_{h'}} (h' < h) \). (iii) for \( h \) follows from (iii) for \( h' \).

(iv) is immediate.

The following promotions are performed. Along \( ST_v \), in descending order, for each lazy tree \( L_{i_k} \), among \( L_{new}, \ldots, L_{old} \), the following nodes are promoted: if \( v \) is on the right split path, all the “global” nodes of \( L_{i_k} \) on \( ST_v \) are promoted; while if \( v \) is on the left split path, the first “global” node of \( L_{i_k} \) on \( ST_v \) is promoted.

We are now ready to specify the new lazy trees formed by these promotions. Again, suppose that \( v \) is on the right (resp. left) split path.

**Case 1.** For each lazy tree \( L \) among \( L_{old'}, \ldots, L_{new'} \), the following new lazy tree \( L' \) is created. (\( v \) is a split point for \( L \).)
Case 1.1. \( v \) is the root of \( L \). From the definition of true split points we conclude that \( v \) has an empty left (resp. right) subtree in the portion of its lazy block tree to the right (resp. left) of the accessed item. Thus the new lazy tree, \( L' \), rooted at \( v \) is trivial (for it does not include the subtree rooted at the staircase point, \( w \)). \( v \) may also be the root of one other lazy tree, \( K' \), which is split, but for which \( v \) is not a split point; this is discussed further, in Case 2, below.

Case 1.2. \( v \) is a split point for \( L \) but not a true split point; \((v \) is not the root of \( L \)). If \( v \) is a direct or indirect guard split point let \( pr \) be the pseudo root of \( L \) and \( r \) the root. \( r \) becomes the root for \( L' \). If \( pr \) is on \( ST_v \) or if the intersection point of \( L \) with \( ST_v \) does not exist, then \( v \) is the new left (resp. right) guard; otherwise, the new left (resp. right) guard is the intersection point of \( L \) with \( ST_v \). The right (resp. left) guard for \( L \) becomes the right (resp. left) guard for \( L' \). Otherwise, \( v \) is a virtual guard split point (node \( v \) of Case (a) or node \( u \) of Case (b) of the promotions). \( v \)'s role is to attempt to provide the right guard for the new lazy tree rooted at, respectively, the root of \( L \), \( w \), or at the corresponding special split point, \( w \). Again, let \( pr \) be the pseudo root of \( L \). \( w \) becomes the root for \( L' \). If \( pr \) is on \( ST_v \) or if the intersection point of \( L \) with \( ST_v \) does not exist, then \( v \) is the new right guard; otherwise, the new right guard is the intersection point of \( L \) with \( ST_v \). The left guard for \( L \) becomes the left guard for \( L' \).

We now form new lazy trees rooted at promoted nodes on \( ST_v \). Let \( u \) be a promoted node on \( ST_v \). Let \( Lu_{\text{old}} \) and \( Lu_{\text{new}} \) be the oldest and newest lazy trees, respectively, among \( L_{\text{new}}, \ldots, L_{\text{old}} \), to which \( u \) belongs. For each lazy tree, \( Lu_{\text{i}} \), among \( L_{\text{new}}, \ldots, L_{\text{old}} \), \( u \) becomes the root of a new lazy tree, \( L_{u_{\text{i}}} \), defined as follows.

Case 1.3. \( v \) is on the right split path. Then \( L_{u_{\text{i}}} \) acquires the following node, \( g_{u_{\text{i}}} \), as its right guard: If the right guard of \( L_{u_{\text{i}}} \) is a descendant of \( u \), then this right guard becomes \( g_{u_{\text{i}}} \). Otherwise, if \( u \)'s right subtree in the lazy block tree for \( L_{u_{\text{i}}} \) is non-empty, \( g_{u_{\text{i}}} \) is the rightmost node in this subtree; \( g_{u_{\text{i}}} \) is promoted by adding its reserve potential to its lazy potential. If neither of these apply, \( L_{u_{\text{i}}} \) is trivial; in this case it is convenient to define \( g_{u_{\text{i}}} = u \). On the path \( ST_v \), let \( u_1, u_2, \ldots, u_k \) be the sequence of promoted nodes all from the same old right lazy tree, \( L_{u_{\text{i}}} \), among \( L_{\text{new}}, \ldots, L_{\text{old}} \). For brevity, we write \( g_h \) for \( g_{u_{h_{\text{i}}}} \), for \( 1 \leq h \leq k \). Then, for \( 1 \leq h < k \), \( g_{h+1} \) becomes the left guard for \( L_{u_{h_{\text{i}}}} \); \( v \) becomes the left guard for \( L_{u_k} \).

Case 1.4. \( v \) is on the left split path. Then \( u \) is on the left extreme path for \( L_{u_{\text{i}}} \). For if \( v \) was a true split point for \( L_{u_{\text{i}}} \), conditions (a) and (b) imply \( v \) is on the left extreme path for \( L_{u_{\text{i}}} \); condition (d) means \( v \) is the root of \( L_{u_{\text{i}}} \); conditions (c) and (e) do not arise for \( v \) is accessed from its right child. If \( v \) is on the left extreme path or is the root of \( L_{u_{\text{i}}} \), then \( u \) must be on the left extreme path of \( L_{u_{\text{i}}} \). We define the new right guard, \( g_i \), for \( L_{u_{\text{i}}} \), as follows. If \( u \)'s right subtree in the lazy block tree for \( L_{u_{\text{i}}} \) is non-empty then \( g_i \) is the rightmost node in this subtree; \( g_i \) is promoted by adding its reserve potential to its lazy tree potential. Otherwise, \( g_i = v \). The left guard for \( L_{u_{\text{i}}} \) is provided by the left guard for \( L_{u_{\text{i}}} \).

We note that \( v \) may be the left, or right, guard for several new lazy trees, up to one new lazy tree formed from each of \( L_{\text{old}}, \ldots, L_{\text{new}} \).

We have yet to pay for the promotions of the nodes on path \( ST_v \). This requires at most \( gp \) times the jump in rank, in the imaginary tree, from node \( v \)'s left (resp. right) child to \( v \). For consider a promoted node, \( u \) on path \( ST_v \). Suppose that \( u \) was promoted as a "global" node on lazy tree \( L \). Let \( u' \) be the left (resp. right) child of \( u \) in the splay tree. If \( v \) is on the right split path, \( u \)'s lazy weight is at least \( wt(u') \) (see Lemma 4). While if \( v \) is on the left split path, \( u \) is on the left extreme path for \( L \) (see Case 1.4); so \( u \)'s lazy weight is again at least \( wt(u') \) (apply Lemma 4 to the normal form of the lazy tree). Hence the lazy potential for a node, \( u \), on \( ST_v \) is at least the normal potential of the next promoted node on \( ST_v \) in descending order (for it is contained in \( u \)'s left, resp. right, subtree). Let \( x = w_j \) be the
bottom promoted node; suppose that $x$ was in lazy tree $L_x$. In the imaginary tree, if $v$ is on the right split path, $\text{lazyrank}(x) \geq \lfloor \log \text{wt}(\text{leftchild}(v)) \rfloor$ (see Lemma 4); while if $v$ is on the left split path, $\text{lazyrank}(x) + 1/gp \cdot \text{reserve}(x) \geq \lfloor \log \text{wt}(\text{rightchild}(v)) \rfloor$ (apply Lemma 4 to the normal from of $L_x$). Hence, adding $x$’s lazy potential plus its reserve potential to $gp$ times the jump in rank from $v$’s left child to $v$ provides at least $v$’s normal potential in the imaginary tree, which is at least $w_x$’s normal potential.

**Case 2.** We now consider new lazy derived trees $L'$ rooted at node $v$, where $v$ is not a split point for $L$, or is a special split point for $L$.

**Case 2.1.** $v$ is the root of split lazy tree $L$. By Lemma 8, $v$ now carries its normal potential. $v$ becomes the root of a new $L$-derived lazy tree, $L'$. The right (resp. left) guard for $L$ becomes the right (resp. left) guard for $L'$. Let $v'$ be the following node: if $v$ is on the right split path, $v'$ is the node $v$ of Case (b) for $L$. If $v'$ is a split point let $w'$ be the staircase point for $v'$, if any. $L'$ acquires the following left guard, $g_l$. If the right subtree of $v'$ in $L$ contains a node “global” on $L$, then the right guard of the $L$-derived lazy tree rooted at $w'$ also becomes the left guard for $L'$; otherwise, $g_l = v'$. We note that on the path from $v$ to $g_l$ there are no promoted nodes in the range $(v, g_l)$. If $v$ is on the left split path, $v'$ is the node $v$ of Case (a) for $L$; $v'$ is a virtual guard split point. This case was already handled in Case 1.2, above.

**Case 2.2.** $v$ is a special split point. Let $u$ be the corresponding virtual guard split point. Then a new lazy tree $L'$ rooted at $v$ is formed. Its right guard was created in Case 1.2 for node $u$; its left guard is the left guard for $L$.

We call the promotions performed in Cases 1 and 2 the **staircase promotions** with respect to $v$. The total cost of the staircase promotions for global nodes to the right of the accessed item is at most $gp \cdot \log n$; likewise, the promotions to the left of the accessed item cost at most $gp \cdot \log n$.

So the promotions add at most $3gp \cdot \log n$ to the cost of an access at level $l$ (recall the charge of $gp \cdot \log n$ for the promoting traversals).

Next, we show that Invariants 15 and 16 continue to hold following the above partitionings of the lazy trees.

**Lemma 17** Invariant 15 continues to hold for each pair of lazy trees following the partitionings of a split step.

**Proof.** We start with a few observations.

(i) Each promoted item is on a lazy tree whose root is traversed.

(ii) Define the near right (resp. left) guard of lazy tree $L$, to be the leftmost (resp. rightmost) node in the block of the right (resp. left) guard. For each new $L$-derived lazy tree $L'$, the near guards of $L'$ enclose an interval which is a proper subset of the interval enclosed by the near guards of $L$.

(iii) The only change that might be made to a lazy tree whose root is not traversed is that it might acquire one new guard; this new guard will be inside the interval previously spanned by its guards (by a new guard, we do not intend the situation in which the guard is a different node in the same block, or more generally, on the same unsplit cover guard tree). We call such lazy trees **defensive lazy trees**.

For the purposes of this proof, we suppose that the promotions are performed in three phases. In Phase 1, we promote the virtual guard split points, thereby partitioning each lazy
tree with such a split point into two new lazy trees (the virtual guard split point provides
the left guard for one new lazy tree and the right guard for the other). Clearly, the invariant
continues to hold for if a node is used to partition tree $L$ by being promoted, it is a "global"
node on $L$.

In Phase 2, we perform all the remaining promotions, except that for each sequence of
promoted nodes on each staircase path, "global" on the same lazy tree, only the topmost node
is promoted: the (temporary) $L$-derived lazy tree, $L'$, rooted at this topmost node is given as
guards the two extreme guards for the sequence of actual $L$-derived lazy trees with roots on
this staircase path. We argue below that the invariant continues to hold.

In Phase 3, the remaining promotions are performed. Again, for each lazy tree, $L$, from
phase 2, only nodes "global" on $L$ are used to partition $L$, so the invariant continues to hold.

Now, we show that following the promotions of Phase 2, the invariant still holds.

Let $L_1$ and $L_2$ be two lazy trees prior to the split. Let $L'_1$ and $L'_2$ be two lazy trees created
in the split, $L'_1$ from $L_1$ and $L'_2$ from $L_2$ (it may be that a tree $L_i$, $i = 1, 2$, is partitioned into
more than one lazy tree, $L'_i$).

Suppose that $L_1$ and $L_2$ satisfied Invariant 15(i) or (ii)b; then the trees $L'_1$ and $L'_2$ also
satisfy Invariant 15(i) or (ii)b. Similarly, if $L'_1$ and $L'_2$ are on opposite sides of the accessed
item, or if they are both $L$-derived lazy trees they satisfy one of Invariant 15(i) or (ii)b.

We consider a number of further possibilities for $L_1$ and $L_2$. Suppose that $L'_1$ is formed by
the action at point $v$ on the right (resp. left) split path; i.e., suppose $L'_1$ has at least one node
in $v$'s right subtree. In Cases 1-4, we consider pairs $L_1$ and $L_2$ in which both $L_1$ and $L_2$ are
modified. In Case 5, we consider the remaining possibilities. We define $L$ to be boring if $L$
is older than $L_{old}$ and both $L$ and $L_{old}$ have split point $v$, and further $v$ is not a special split
point for $L$.

Case 1. $L_1$ is boring. If $L_1$ and $L_2$ are both boring, then they satisfy Invariant 15(ii)a.
The only change in obtaining $L'_1$ and $L'_2$ is that the left (resp. right) guard of $L_1$ (resp. $L_2$)
is changed to $v$; so $L'_1$ and $L'_2$ also satisfy Invariant 15(ii)a. If $L_2$ is not boring and $v$ is a
non-special split point for $L_2$, then any changes in $v$'s right (resp. left) subtree that are part
of $L'_2$ must be promotions of nodes to the right (resp. left) of or equal to the right (resp. left)
guard of $L_1$, by Invariant 15 (since all such nodes are on $L_{old}, \ldots, L_{new}$). So if $L'_2$ has left
(resp. right) guard $v$, $L_1$ and $L_2$ satisfy Invariant 15(ii)a; otherwise, they satisfy Invariant
15(ii)b. Finally, if $L_2$ is not boring and $v$ is not a split point for $L_2$ or is a special split point
for $L_2$, then the subtree of $L'_2$ in $v$'s right (resp. left) subtree is equal to the subtree of $L'_2$ in
$v$'s right (resp. left) subtree. As $L_1$ and $L_2$ satisfied Invariant 15(ii)a, so do $L'_1$ and $L'_2$.

Case 2. $L_1$ is defensive but not boring. So $v$ is the split point for $L_1$.

Case 2.1. $v$ is the left (resp. right) guard for $L'_1$. Let $e_1$ be intersection point with $ST_v$ for
$L_1$; $e_1$ is the pseudo root of $L_1$ (see Case 1.2 of the staircase promotions). Let $r'_2$ be the pseudo
root for $L_2$ and let $L'_2$ be the $L_2$-derived lazy tree with nodes in $v$'s right subtree.

Case 2.1a. $L'_2$ is rooted at $v$. If $v$ is a true split point, other that a special split point, for
$L_2$, then $L'_2$ is trivial. Otherwise, $v$ is the node of Case 2 of the staircase promotions. Then
the portion of $L_2$ in $v$'s right (resp. left) subtree is identical to the portion of $L'_2$ in $v$'s right
(resp. left) subtree; as $L_1$ and $L_2$ satisfied Invariant 15(ii)a or (ii)c, $L'_1$ and $L'_2$ satisfy Invariant
15(ii)c (by Lemma 13 and 15, respectively, $L_1$ is newer than $L_2$).

In the remaining subcases of Case 2.1, we suppose that $L'_2$ is not rooted at $v$.

Case 2.1b. $r'_2$ is a descendant of $e_1$ (including $e_1$ itself). Suppose that $L_2$ is older than $L_1$.
$L'_2$ comprises a rightmost (resp. leftmost) portion of $L_2$ plus possibly a new root and a new
left (resp. right) guard, $g_{l_2}$; $g_{l_2}$ is either $v$ or to the right (resp. left) of $v$. The left (resp. right)
guard of \(L_2\) was not to the right (resp. left) of \(v\). As \(L_1\) and \(L_2\) must have satisfied Invariant 15(ii)a or (ii)c, \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)a.

Next, suppose that \(L_2\) is newer than \(L_1\). Then the pseudo root of \(L_2'\), \(r_2'\), is a right (resp. left) descendant of \(e_1\). If \(r_2'\) is a proper descendant of \(e_1\), then, by Invariant 16, \(e_1\) was promoted by the split (for if \(e_1\) already carried its normal potential, \(r_2'\) could not be a proper descendant of \(e_1\)). Then, if \(L_2\) is defensive, \(e_1\) provides the left (resp. right) guard for \(L_2'\), otherwise, \(e_1\) is the root of \(L_2'\) and \(v\) is the right (resp left) guard (for by Invariant 15 applied to \(L_1\) and \(L_2\), there is no “global” node on \(L_2\) in \(e_1\)’s left, resp. right, subtree). As \(L_1\) and \(L_2\) satisfied Invariant 15(ii)a, \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)c or 15(ii)a, according as \(L_2\) is defensive or not. So suppose that \(e_1 = r_2'\). Then, by Invariant 15(ii)a, applied to \(L_1\) and \(L_2\), \(e_1\) must be the leftmost (resp. rightmost) “global” node on \(L_2\) to the right of \(v\), so the new left (resp. right) guard for \(L_2\) is \(v\). As \(L_1\) and \(L_2\) satisfied Invariant 15(ii)a, \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)a.

Case 2.1c. \(r_2'\) is a right (resp. left) descendant (not necessarily proper) of a node on \(ST_v\), which is a proper ancestor of \(e_1\). If \(L_2'\) is defensive, with guard \(v\), we reverse the roles of \(L_1\) and \(L_2\) and appeal to the first paragraph of Case 2.1b. If \(L_2'\) is defensive, but with left (resp. right) guard on \(ST_v\), then \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)b. The only other possibility is that \(v\) is a split point for \(L_2\), on \(L_2\). Then \(L_1\) and \(L_2\) satisfy Invariant 15(ii)a; so \(L_1'\) and \(L_2'\) will satisfy one of Invariant 15(ii)a, (ii)b, (ii)c, according as the left (resp. right) guard of \(L_2'\) is \(v\), a proper ancestor of the root of \(L_1'\), and the root of \(L_1'\).

Case 2.1d. \(L_2\) is the tree of Case 2 of the staircase promotions, where the \(v\) here corresponds to the \(v'\) of Case 2.1 of the staircase promotions. We defer this to Case 4, below. Note that if \(L_2'\) is the tree of Case 2.2 of the staircase promotions then the Phase 1 promotion makes the portion of \(L_2\) containing \(L_2'\) into a defensive lazy tree; this case is handled by Cases 2.1a-c, above.

Case 2.2. \(L_1'\) is a defensive lazy tree with node, \(u\), on \(ST_v\) as left (resp. right) guard. Suppose that \(L_2'\) is also a defensive tree. The case in which \(L_2'\) has \(v\) as left (resp. right) guard has already been dealt with in Case 2.1. If \(L_2'\) also has left (resp. right) guard \(u\), then as \(L_1\) and \(L_2\) satisfied Invariant 15(ii)a, so \(L_1'\) and \(L_2'\). Otherwise, the left (resp. right) guard of \(L_2'\) is either a proper ancestor or a proper descendant on \(ST_v\) of \(u\). As \(L_1\) and \(L_2\) satisfied Invariant 15(ii)a, \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)b.

Next, suppose that \(L_2'\) is not defensive. If \(L_2\) is older than \(L_1\), then the root of \(L_2'\) is a left (resp. right) descendant of \(u\) and \(L_1\) and \(L_2\) satisfied one of Invariant 15(ii)a or (ii)c; so \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)b or (ii)c, according as the root of \(L_2'\) is a proper descendant of \(u\), or is equal to \(u\). If \(L_1\) is older than \(L_2\), then the root of \(L_2'\) is an ancestor of \(u\) and \(L_1\) and \(L_2\) satisfy Invariant 15(ii)a. So \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)a or (ii)b, the former arising if the root of \(L_2'\) is \(u\) or if the left (resp. right) guard of \(L_2'\) is \(u\) or \(v\), and the latter arising otherwise.

Case 3. \(v\) is a true split point for \(L_1\) and \(L_2\). Suppose that \(L_2'\) is an older tree than \(L_1'\); then \(L_1\) and \(L_2\) satisfy Invariant 15(ii)(a). If \(v\) is the left (resp. right) guard of \(L_1'\), \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)a. Otherwise, the root of \(L_2'\) is a proper left (resp. right) descendant of the left (resp. right) guard for \(L_1'\); \(L_1'\) and \(L_2'\) satisfy Invariant 15(ii)b. Finally, suppose that \(L_2'\) is a newer tree than \(L_1'\); then interchange the roles of \(L_1\) and \(L_2\) in the previous three sentences.

Case 4. \(L_1'\) is a split tree, but \(v\), the root of \(L_1\), is not a split point for \(L_1\). (See Case 2.1 of the staircase promotions.) We note that there cannot be two trees of this type both rooted at \(v\) (for on the newer tree, \(v\) would be a true split point of type (d) ). To the right (resp left) of \(v\), \(L_1\) is identical to \(L_1'\). Let \(L_2\) be a tree with split point \(v\); by Lemma 13, \(L_2\) is newer than
$L_1$. So the bottom promoted node $w_k$ on $ST_v$ is either the right (resp. left) guard for $L_1$ or is to the right (resp. left) of this right (resp. left) guard. Thus for those trees $L'_2$ that have left (resp. right) guard $v$, $L'_1$ and $L'_2$ satisfy Invariant 15(ii)c, while for all other trees $L'_2, L'_1$ and $L'_2$ satisfy Invariant 15(ii)b.

Next, let us consider the point $v'$ from Case 2.1 of the staircase promotions. All the split trees, $L_2$, other than $L_1$, with split point $v'$, are older than $L_1$ (by Lemma 14). $L_1$ and $L_2$ satisfy Invariant 15(ii)a. Let $g$ be the left guard for $L'_1$. Recall that $g$ is either $v'$ or the rightmost item in $L_1$ in the right subtree of $v'$. So if the root of $L'_2$ is to the right of $g$, then $L'_1$ and $L'_2$ satisfy Invariant 15(ii)a; otherwise, if the root of $L'_2$ is to the left of $g$, $L'_1$ and $L'_2$ satisfy Invariant 15(ii)b; finally, if the root of $L'_2$ is $g$, $L'_1$ and $L'_2$ satisfy Invariant 15(ii)c.

**Case 5.** $L_1$ does not have a split point. Since all the promoted nodes were “global” nodes on trees which had a split point, the only node on $L_1$ that might be promoted is its root. Also, the access must be outside the open interval defined by $L_1$’s guards.

Let $L_2$ be an older lazy tree. First, suppose that $L_1$ and $L_2$ satisfied Invariant 15(ii)a; then $L_2$ also does not have a split point. As $L_1$ and $L_2$ are unchanged they continue to satisfy Invariant 15(ii)a. Second, suppose that $L_1$ and $L_2$ satisfied Invariant 15(ii)c; let $v$ be the root of $L_2$ (which is also the right, resp. left, guard of $L_1$). While $L_2$ may be split, the $L_2$-derived tree, $L'_2$, which includes $S_v$, the right (resp. left) subtree of $v$, is unchanged on the (possibly empty) portion of $S_v$ contained in $L_1$ (for otherwise $v$ would be a split point for $L_2$ and hence an indirect guard split point for $L_1$, entailing a, possibly trivial, modification of $L_1$). So Invariant 15(ii)c applies to $L_1$ and $L'_2$, and Invariant 15(ii)b applies to $L_1$ and any other $L_2$-derived tree.

Next, let $L_2$ be a newer lazy tree. Suppose that $L_1$ and $L_2$ satisfy one of Invariants 15(ii)a or (ii)c. Let $L'_2$ be an $L_2$-derived tree for which the open interval defined by its guards overlaps the open interval defined by $L_1$’s guards. If a new guard and/or root for $L'_2$ comes from a tree no older than $L_1$, then it is either the root of $L_1$ or outside the open interval spanned by $L_1$’s guards. While if the new guard and/or root for $L'_2$ is from a lazy tree older than $L_1$, by the argument of the previous paragraph, the new guard and/or root is outside the open interval spanned by $L_1$’s guards. So if the root of $L_1$ is not a guard of $L'_2$, then $L_1$ and $L'_2$ satisfy Invariant 15(ii)a; while if the root of $L_1$ is a guard of $L'_2$, then $L_1$ and $L'_2$ satisfy Invariant 15(ii)c. $L_1$ and any other $L_2$-derived tree satisfy Invariant 15(ii)b.

**Lemma 18** Invariant 16 continues to hold for each lazy tree following the partitionings of a split step.

**Proof.** We use the following claim: there are no promotions on each new $(u, v)$-neighbor path in each new $L$-derived lazy tree $L'$; the claim is shown later on in this proof. It follows that for each promoted node, $u$, for each lazy tree, $L$, to which $u$ belonged, if $u$ was not already a guard or the root of $L$, then $u$ becomes a root or a guard of a new $L$-derived lazy tree, $L'$, or ceases to be on any $L$-derived lazy tree. We conclude that each “global” non-root node in a new lazy tree carries its lazy potential (for it was not promoted); i.e., (i) continues to hold.

(ii) continues to hold, for if a “global” node is not promoted it remains “global”.

We now show (iii). Let $u, v$ be a pair of $L'$-neighbors. $u$ and $v$ are on a $(t, w)$-neighbor path in $L$ (possibly $t = u, w = v$). Without loss of generality, suppose that $t$ is an ancestor of $w$. Apply Invariant 16(iii) to the $(t, w)$-neighbor path; For the oldest tree $L_1$ provided by the lemma, either $u$ provides the root of a new lazy tree $L'_1$ which fulfills Invariant 16; or the new $L_1$-derived lazy tree, $L'_1$, rooted at $u$ must be trivial, so there are no “global” nodes of

34
$L_1$ on the portion of the $(t, w)$-neighbor path descending from $u$; in fact, this right (resp. left) extreme path of $L_1$ continues to the right (resp. left) guard, $w_1$, of $L_1$ without encountering another node “global” on $L_1$; $w_1$ is a left (resp. right) descendant of $w$ (possibly $w = w_1$). Let $t_1$ (resp. $w_1$) be the “global” node on $L_1$ on the neighbor path immediately above $u$ (possibly $t_1 = t$). Now apply the argument iteratively to the $(t_1, w_1)$-neighbor path in $L_1$ (the iterations terminate since there are only finitely many distinct ages of trees; formally, the proof could be cast as an inductive argument).

We complete the proof by showing the claim. Let $L'$ be a new lazy tree created by the action at split point $v$. Suppose that $v$ is a split point on the right (resp. left) split path.

Case 1. The pseudo root, $r'$, of $L'$ is not on $ST_v$. If $e$, the intersection point of $L$ with $ST_v$ exists, then $r'$ is a proper right (resp. left) descendant of $e$; by Lemma 16, the path $ST_v$ continues to the left (resp. right) of $e$ and hence there are no promotions in $e$’s right (resp. left) subtree; in addition, there are no promotions on the path from $v$ to $e$ in the range $(v, e)$ (resp. $(e, v)$). The claim follows for $L'$. Likewise, if $e$ does not exist, $L$ is older than $L_{old}$ and hence there are no promotions in the range $(v, r')$ (resp. $(r', v)$); also, there are no promotions in $r'$’s subtree. Again, the claim follows for $L'$.

Case 2. The pseudo root, $r'$, of $L'$ is on $ST_v$. Then $r'$ is the intersection point of $L$ with $ST_v$. Promotions of nodes in $r'$’s left (resp. right) subtree are of “global” nodes on $L$ or of nodes on trees older than $L$; the latter set of nodes are not on $L$. The only node that might be promoted in $r'$’s right (resp. left) subtree is the right (resp. left) guard for the newest lazy tree, $L'_{new}$, rooted at $r'$; if $L_{new} \neq L$, this node is to the right (resp. left) of $L$, and otherwise it is a “global” node on $L$. Again, there are no promotions on the path from $v$ to $e$ in the range $(v, e)$ (resp. $(e, v)$). The claim follows for $L'$.

Case 3. $v$ is a virtual guard split point for $L$. Let $L'$ be the $L$-derived lazy tree which contains nodes in $v$’s left subtree, if any. The argument of Cases 1 and 2 shows that the portion of $L'$ in $v$’s left subtree satisfies the claim.

Case 4. $v$ is a special split point for $L$. Promotions in $v$’s left subtree, by Lemma 15, are of nodes “global” in lazy trees newer than $L$ and so are either at or to the left of the left guard for $L$ (see Invariant 15); the claim follows for the portion of $L$ including $v$ and its left subtree. Let $v'$ be the corresponding virtual guard split point. There are no promotions on the left split path between $v'$ and $v$; so it remains to consider the left subtree of $v'$. But this was dealt with in Case 3.

Case 5. $v$ is the root of $L$ but is not a split point for $L$. As in Case 4, the claim holds for the portion of $L'$ including $v$ and its right (resp. left) subtree. There are no promotions on the path from $v$ to either the promoted node $w = v$ of Case (a) (for $v$ on the left split path) in the range $(v, w)$, or to the promoted node $w = v$ of Case (b) (for $v$ on the right split path) in the range $(w, v)$, so the claim continues to apply to these paths. For $w$ of Case (a), we have already shown in Case 3 that the claim holds for the left subtree of $L'$ rooted at $w$. For $w$ of Case (b), the left guard, $g_l$, for $L$ is either $w$ or the rightmost “global” node on $L$ in the right subtree of $w$. It is clear there is no promoted node on the path from $g'$’s $L$-right neighbor, $glob$, to $g$, in the range $(g, glob)$ ($glob$ is equal to or to the left of $v$). In addition, there are no promoted nodes to the right of $g$ on $L$ in the right subtree of $v'$. This proves the claim for the portion of $L'$ to the left of $v$.

Now we need to show how to restore Invariants 5-14. Invariants 12-14 are restored exactly as before. It remains to show how to restore Invariants 5-11. For each node that was on a newest lazy tree $L$, but is not on an $L$-derived lazy tree, we charge the removal of its debit, if any, to the node’s potential associated with lazy tree $L$. So we are only concerned with nodes.
that remain on a lazy tree of the same age. For such nodes, the invariants are all restored exactly as in Section 3.1.3. At first sight, Invariant 10 might appear to cause some problems; but we note that for each promoted node \( u \), there is at most one lazy tree for which Invariant 10 might not hold: the newest lazy tree to which \( u \) belongs (for recall that each Invariant is interpreted with respect to the newest tree to which the nodes in question all belong). So to restore Invariant 10 for a newly promoted node \( u \) requires the removal of at most one small debit. Likewise, Invariant 11 might appear to cause some difficulties. The charging proceeds as follows. The removal of each lazy debit is charged to the potential associated with the lazy tree to which the debit belongs; except for node \( z \) of Case 3.2 it should be clear that for each lazy tree to which a node \( y \) of Case 3 belongs, there are at most four lazy debits to remove, as before. For node \( z \), a promoted node, the charge to the promotion occurs only if \( z \) is a "global" node on the left extreme path of its lazy tree; but this can apply to at most one lazy tree to which \( z \) belongs. So again, the charge to node \( z \) is bounded as in Case 3.2.

So a node \( v \) is charged only in the following situations: (i) if it is a promoted node; (ii) if it was on a lazy tree \( L \) and is not on an \( L \)-derived tree; (iii) if it was on a lazy tree \( L \), and it was not in the leftmost block for \( L \), but it is now in the leftmost block for its current \( L \)-derived lazy tree. We note that \( v \) may be charged with respect to several lazy trees to which it belonged and to which one of these conditions applies; this causes no problems since \( v \) carries a separate potential for each lazy tree to which it belongs.

Each promoted node may have to pay a segment charge, as in Section 3.1.3. We conclude that Equations 14 and 15 of Section 3.1.3 suffice. So the presence of multiple levels of lazy trees does not alter the previous analysis of an global access, the first access of a block.

We turn to the local accesses for a given block. The first \( e - 1 \) local accesses are treated as global accesses, as before. It remains to consider the true local accesses. Each couple in such an access costs at most \( q + s + 1 \) units; the \( q \) units provide the potential for the new lazy node, the single unit pays for the rotation itself, and the \( s \) units provide \( s/2 \) spares to each of the nodes in the couple. We note that each node in the couple carries no debit (for they have already been traversed at least twice). If the nodes in the couple are from the same (old) lazy tree, we need to maintain the lazy tree potentials; this is done essentially as before (Sections 3.1.1 and 3.1.2). The only detail is that for a couple comprising two nodes from the same lazy tree, we need to maintain the lazy tree potentials; to do this the procedure for the traversal of an extreme path of a lazy tree is followed, except that no debits are created. So the bound of Section 3 on the cost of a local access is unchanged. We note that the nodes forming the new lazy tree carry no debits when the new lazy tree is created.

### 3.3 Concluding the Analysis

Clearly, a local insertion only performs left path traversals on lazy trees. As noted in Section 3.1.2, such a traversal does not cause any spare rotations to be spent. So, as claimed at the start of Section 3, spare rotations accumulate over the first \( e - 1 \) local insertions into a block.

We choose \( s = 1 \). By Lemma 3 we have \( g = ld = 34 \), \( c = 36 \). On taking equality in Equations 4, 6, 9, 12, 13, 14, 15, we obtain \( hd = 36 \), \( d = 432 \), \( b = 212 \), \( c' = 144 \), \( a = 2 \cdot 216^2 = 93,212 \) (incidentally, these values satisfy Equations 7 and 8) and \( q + 4 \leq 2^{19} \), by Equation 4 it suffices to set \( e = 20 \). From Equation 5, we conclude:

**Theorem 1** The number of rotations for sorting a log \( n \)-block sequence is bounded by \( 2300n + O(n/\log n) \). (Recall that the number of rotations dominates the overall cost of the sort.)
4 Acknowledgements

We thank Rajamani Sundar for his very careful reading of the paper which uncovered several serious omissions in an earlier draft.

References


key: • global node
  o local node

**Figure 1**

**Figure 2**

**Figure 3**

**Figure 4**

x is the root of u’s block
Figure 5

- `r` is root of splay tree and of `w`'s blade; `w` is inserted item.
- `·` global node
- `○` local node

Figure 6

- `u` is root of lazy tree
- `v` has active span `[l, l']`
- `v` becomes root of lazy tree
- `u` acquires active span `[l, l']`
- Blade span `[1, l-1]`

Figure 7

- Corresponds to
Figure 8

key:  
- node on L new
- right guard for L new
- node not on L new

(1) Suppose trees $L_{i_1}$, $L_{i_2}$, $L_{i_3}$, $L_{i_4}$ are split.
Suppose $v$ belongs to trees $L_{i_1} = L_{i_1}'$, $L_{i_2} = L_{i_3}'$.

Nodes are labeled by tree(s) to which they belong.

key:  
- "global" node on $L_i$ or $L_{i_2}$
- "local" node on $L_i$ or $L_{i_2}$
- node on $L_{i_2}$ or $L_{i_4}$

Comment: In general, a node may be "local" on $L_{i_2}$ and "global" on $L_i$.
On the dynamic finger conjecture for splay trees. Part I

C.2