ON THE DYNAMIC FINGER CONJECTURE FOR SPLAY TREES. PART II: THE PROOF

RICHARD COLE*

Abstract. The following result is shown: On an n-node splay tree, the amortized cost of an access at distance d from the preceding access is $O(\log(d+1))$. In addition, there is an $O(n)$ initialization cost. The accesses include searches, insertions and deletions.

Key words. Binary search tree, splay tree, amortized analysis, finger search tree

AMS subject classifications. 68P05, 68P10, 68Q20, 68Q25

1. Introduction. The reader is advised that this paper quotes results from the companion Part I paper [CMSS98]; in addition, the Part I paper introduces a number of the techniques used here, but in a somewhat less involved way.

The splay tree is a self-adjusting binary search tree devised by Sleator and Tarjan [ST85]. They showed that it is competitive with many of the balanced search tree schemes for maintaining a dictionary. Specifically, Sleator and Tarjan showed that a sequence of m accesses performed on a splay tree takes time $\mathrm{O}(m \log n)$, 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 *splay*(x) applied to an item x in the splay tree. *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 p is the right child of g and x is the left child of p: 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).

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 (our terminology). Node x is given weight, $wt(x)$, equal to the number of nodes in its subtree; they define the *centroid rank* of x, or simply the *rank* of x to be $rank(x) = \lfloor \log wt(x) \rfloor$ (our terminology). Each node is given a

*Computer Science Department, Courant Institute, New York University, 251 Mercer Street, New York, NY 10012-1185 (cole@cs.nyu.edu). This work was supported in part by NSF grants CCR-8702271, CCR-8902221, CCR-8906949, CCR9202900, CCR-9503309, CCR-9800085, by ONR grant N00014-85-K-0046, and by a John Simon Guggenheim Memorial Foundation Fellowship. This work was made possible, in part, by the hospitality of the Laboratoire d'Informatique, Ecole Normale Supérieure.

centroid potential equal, in units, to its centroid rank. Let δ denote $rank(g) - rank(x)$, if g is present; otherwise it denotes $rank(p) - rank(x)$. Sleator and Tarjan showed that the amortized cost of the splay step if g is present is at most 3δ units, while if g is not present the cost is at most $\delta + 1$ units (see [ST85], Lemma 1). 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.*

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 $\mathrm{O}(n)$ plus a constant times the time required by algorithm A.

DYNAMIC FINGER CONJECTURE. The total time to perform m accesses on an arbitrary n-node splay tree is $\mathrm{O}(m + n + \sum_{j=1}^{m} \log(d_j + 1))$, where, for $1 \leq i \leq m$, the jth and $(j-1)$th accesses are performed on items whose ranks differ by d_j (ranks among the items stored in the splay tree). For $j = 0$, the jth item is interpreted to be the item originally at the root of the splay tree.

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 $\mathrm{O}(n)$.

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 [STT86], [W86, T85, Su89, Luc88a, Luc88b]. [STT86] showed that the rotation distance between any two binary search trees is at most $2n - 6$ and that this bound is tight; they also related this to distinct triangulations of polygons; although connected to the splay tree conjectures, this result has no immediate application to them. [W86] provided two methods for obtaining lower bounds on the time for sequences of accesses to a binary search tree; while some specific tight bounds were obtained (such as accessing the bit reversal permutation takes time $\Theta(n \log n)$) no general results related to the above conjectures follow. [T85] 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 $\mathrm{O}(n)$. Sundar [Su89] considered various classes of rotations on binary trees. Before discussing these we need some terminology. A right rotation is a rotation between a node u and its left child v. A k-right cascade comprises a sequence of k right rotations on the following path: the path is a descending sequence of $2k$ nodes, $v_1, v_2, \cdots, v_{2k}$, where v_{i+1} is the left child of v_i, for $1 \leq i < 2k$; the rotations are between nodes v_{2i-1} and v_{2i} for $1 \leq i \leq k$. Sundar [Su89] shows the following bound on the maximum number of k-cascades in a sequence of right rotations on an n-node binary tree: $\Theta(n\alpha_{\lfloor k/2 \rfloor}(n))$, where α_k is the inverse of the kth level of the Ackerman hierarchy; in particular, for $k = 2\alpha(n)$, where α is the inverse Ackerman function, this is a $\Theta(n)$ bound. Our result will make use of this bound. Sundar uses this bound to obtain results concerning the Deque Conjecture; formulated by Tarjan [T85], it states that if a splay tree is used to implement a deque, in the natural way, then a sequence of m operations on a deque, initially of n items, take time $O(m+n)$; Sundar proved a bound of $O((m+n)\alpha(m+n))$.

In this paper, we prove a generalized form of the Dynamic Finger Conjecture, which includes inserts and deletes as well as searches among possible accesses (which is implicit in the above formulation of the conjecture). We note that our implementation of the delete operation is not the one described by Sleator and Tarjan [ST85]. The generalized form of the Dynamic Finger Conjecture states that:

Generalized Dynamic Finger Conjecture. The total time to perform m searches, insertions and deletions on an arbitrary splay tree, initially of size n, is $\mathrm{O}(m+n+\sum_{j=1}^{m}\log(d_j+1))$, where, for $1 \leq i \leq m$, the jth and $(j-1)$th accesses are performed on items whose ranks differ by d_j (ranks among the items stored in the splay tree). For $j=0$, the jth item is interpreted to be the item originally at the root of the splay tree.

Comment: The $+1$ in the log term is present to avoid problems with $\log 1$ and $\log 0$.

To measure the rank difference for an item deleted or inserted on the previous or the current operation, respectively, simply consider the immediate predecessor or successor of the item at the time of the deletion or insertion.

In the Part I paper a special case of Splay Sorting was examined. Splay Sorting is closely related to the Dynamic Finger Conjecture. It 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. This is called *Splay Sorting*. A corollary of the Dynamic Finger Conjecture is the:

Splay Sort Conjecture. Let S be sequence of n items. Suppose the ith 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 $\mathrm{O}(n+\sum_{i=2}^{n}\log(I_i+1))$.

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

Splay Sort Inversion Conjecture. Let S be sequence of n items. Suppose the ith item in S has I_i inversions in S (counting inversions both to left and right). Then Splay Sort takes time $\mathrm{O}(n+\sum_{i=1}^{n}\log(I_i+1))$.

In the Part I paper we proved 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 showed an $\mathrm{O}(n)$ bound for Splay Sorting a $\log n$-block sequence.

The key contribution of the Part I paper was the notion of lazy potentials and their analysis. 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 $\mathrm{O}(\log n)$, so essentially all the reduction in potential is wasted.) The idea of the lazy potential is to keep an old potential, ϕ_{old}, in situations in which the creation of a new (larger) potential, ϕ_{new}, may be followed by a return to the

ϕ_{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'. In the Part I paper, collections of nodes all having lazy potentials formed *lazy trees*. The nodes in a lazy tree had originally been contiguous nodes on a left (or right) path. The lazy potentials were the "correct" potentials with respect to that original path. The key to the analysis was to provide additional potentials to the nodes of the lazy tree to pay for rotations that restored them towards their original left (or right) path configuration. As it happened these potentials were linear in the size of the lazy tree and could be provided when the lazy tree was being created.

The lazy potential used here is similar to that used in the Part I paper. But, by contrast with the Part I paper, the additional potentials provided to the lazy trees may be superlinear in the size of the lazy tree. However, whenever we need to provide a superlinear potential it is superlinear by only an inverse Ackerman function and again we are able to provide this superlinear potential as the lazy trees are being created (the proof of the superlinear bound relies on the analysis of cascades provided by Sundar [Su89]). At this point it is convenient to define the version of the inverse Ackerman function, α, used in this paper. Define:

$$\begin{aligned} A_0(j) &= 2j \quad \text{for all } j \geq 1 \\ A_1(j) &= 2^j \quad \text{for all } j \geq 1 \\ A_i(j) &= \begin{cases} A_{i-1}(2) & \text{if } i \geq 2 \text{ and } j = 1 \\ A_{i-1}(A_i(j-1)) & \text{if } i \geq 2 \text{ and } j \geq 2 \end{cases} \\ \alpha(n) &= \min\{k \geq 1 | A_k(1) \geq n\} \quad \text{for all } n \geq 1 \end{aligned}$$

The other main feature of our analysis is a hierarchical partitioning of the items into blocks. A block comprises a contiguous set of items in the splay tree. The blocks at level 0 in the partition are called 0-blocks; at successively higher levels of the partition, the blocks are termed 1-blocks, 2-blocks, $\cdots$. The highest level, vis, contains a single block which holds all the items in the tree. A separate potential function is associated with each block. The accesses are divided into sequences with respect to each level of blocks. A sequence, S_i, comprises a maximal series of successive accesses all to the same i-block. We call the first access of each sequence S_{vis-1}, a *global* access. Subsequent accesses of S_{vis-1} are called *local* accesses. We will prove an $O(\log n)$ bound on the amortized cost of global accesses and appropriately smaller bounds on the cost of local accesses.

In Section 2, we analyze the global accesses. In Section 3, we introduce and analyze the local accesses. We show a bound of $O(\log(d+1) + (\log\log n)\alpha(n))$ on the cost of a local access, where d is the distance, in items, between the current accessed item and the previously accessed item. In Section 4, this bound is tightened to obtain the desired $O(\log(d+1))$ bound. The additional $O((\log\log n)\alpha(n))$ term arises because of the interaction between the different levels of blocks, as we will see later. In Section 5, we extend the bound to allow insertions and deletions. Unfortunately, the constants in the bound are rather large (10^18). In part, this may be due to the fact that we have not sought to optimize the constants, but to a larger extent is a result of the involved nature of the analysis. In Section 6, we briefly discuss whether it might be possible to improve the constants in our bounds.

2. The Block Structure and Global Accesses. Before embarking on the analysis of global accesses, we provide some definitions which will be needed throughout the paper. Let S be a non-empty contiguous subset of the items in the splay tree.

We define the tree, T_S, induced by S, as follows. let u and v be, respectively, the leftmost and rightmost items in S. The root, r, of T_S, is the least common ancestor of u and v in the splay tree; note that r is a node of 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. A node u is a *left descendant* of v if it is in v's left subtree or is v itself; we also say that v is a *right ancestor* of u. Right descendants and left ancestors are defined analogously. The right (resp. left) extreme path of a tree is the path from the root to the rightmost (resp. leftmost) item in the tree, excluding the root itself. The *access path* for an access comprises the traversed nodes other than the accessed node itself.

Next, we define blocks at level i, i-blocks, for $i = 1, 2, \cdots$. A 0-block comprises 1 item; a 1-block comprises c_1 items; an i-block comprises b_i $(i-1)$-blocks, for $i > 1$; c_i is the number of items in an i-block, for $i \geq 1$. For much of the paper, we choose $b_i = 2^{2^i}$, $c_i = b_{i+1}$, for $i \geq 1$. For each $i \geq 1$, we allow the rightmost i-block to be undersize. At the topmost level, level vis, a single block contains all the items in the tree. Henceforth, to simplify the discussion, we refer to an i-block as the block, an $(i+1)$-block as a superblock, an $(i-1)$-block as a subblock, and a j-block, $j < i-1$, as a miniblock. Sometimes, if confusion may arise, we refer to these blocks as i-superblocks, i-subblocks, and i-miniblocks, respectively.

We make a number of definitions with respect to the superblocks. The *superblock tree* for superblock B is the tree induced by the items in B. The *root* of the superblock tree for a superblock is called the *superblock root*. The block tree and block root for block B are defined analogously. Henceforth, when a node is identified as a block root, it is implicitly assumed not to be a superblock root. The block roots in the superblock and the superblock root itself are called *global nodes*; a global node other than the superblock root is called a *true global node*. All other nodes in the superblock are called *local nodes*. The *skeleton* of a superblock comprises the nodes on the extreme paths of its blocks. The *inner skeleton* of a superblock comprises the nodes on the skeleton other than nodes on an extreme path of the superblock. A node has *visibility* i if it belongs to the inner skeleton of an i-superblock. A node on an external path of the vis-block has visibility vis. On occasion, for precision, the skeleton (resp. inner skeleton) of an i-superblock is called the i-skeleton (resp. i-inner skeleton). Also, the global nodes in an i-superblock are said to be i-global.

Let P be a maximal contiguous path in the splay tree which is a portion of an extreme path for u's block, where u is the root of the block. If P is on the inner skeleton of u's superblock and if the top node on P is a child of u in the splay tree, then P is said to *abut* u. The *right* (resp. *left*) abutting path for u is the path to the right (resp. left) of u that abuts u, if any. Also, u is called the *parent* of P.

The nodes in each i-superblock are given centroid ranks, called the global ranks, or *g_rank*s for short, using the following weighting. Let $K_i = 2^{\lceil \log b_{i+1} \rceil}$, for $i > 0$, and $K_i = 2^{\lceil \log c_1 \rceil}$, for $i = 0$. The root is given weight $-K_i$, each true global node is given weight one, and each local node is given weight zero. The rank of v is given by $\lfloor \log wt(\text{subtree rooted at } v) \rfloor$, where we define $\log 0 = -1$. The two extreme nodes in the superblock are given additional weight K_i. We note that on an extreme path of a superblock tree, each node has the same global rank; also, this global rank is larger than that of any node on the inner skeleton. Each true global node is given a potential, called its global potential, equal in units to gp times its global rank, gp a constant to be specified later.

Each local node on the skeleton is either *labeled* or *unlabeled*. Each labeled node has potential c, c a constant to be defined later. The labels and potentials are defined with respect to each level of block to which the node belongs. For each maximal path P of unlabeled nodes, the following invariant is maintained.

Invariant 1. *Let P be a maximal path of unlabeled local nodes on an extreme path of one block. Suppose that the nodes of P are contiguous in the splay tree. Then P has an associated* path potential *of* $2c\lfloor\log(|P|+1)\rfloor$ *units.*

Each of the terms just defined, if prefixed by i will refer to a block at level i; alternatively we may use the prefixes "super," "sub" and "mini" to refer to the terms associated with superblocks, subblocks and miniblocks. Implicitly, in the discussion that follows, all references are with respect to a fixed i-superblock.

A labeled node on the inner skeleton may carry a *small* or *large* debit. Small and large debits, are worth sd and ld units, respectively, sd and ld constants to be specified later. The following invariants apply to the labels and debits.

Invariant 2. *An unlabeled local node is always adjacent, in the splay tree, to two other local nodes which are on the skeleton and in the same block.*

Invariant 3. *Only a labeled local node on the inner skeleton can have a debit.*

Invariant 4. *Let v be a local node and let v belong to 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 in block B.

(ii) v has a left (resp. right) child w in the splay tree which is a local node and in block B.

(iii) Neither u nor w carry any debit.

(Note that both u and w are on an extreme path of block B and hence on the inner skeleton.)

Invariant 5. *Let u be a true global node. Let P be a path abutting u, if any. Let v be the top node on P (v is a child of u). v can have a small debit only if $g\_rank(v) < g\_rank(u)$.*

Invariant 6. *Let u be the root of block B. Let P be a path abutting u, if any. Let w be the bottom node on P. w can have a small debit only if $g\_rank(w) < g\_rank(u)$.*

To avoid special cases it is convenient to redefine the access path for an access to exclude the splay tree root r in the event that r is involved in an incomplete rotation (Case 1 of the Splay step). 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.

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 nodes on the truncated segment all belong to the inner skeleton of one superblock; in addition, the accessed item is part of the inner skeleton of this superblock immediately prior to the traversal of the current segment.

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

(iii) For each couple, the rotation does not change the total global potentials.

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

(v) Define a node to be *staying* if it is involved in a zig-zag rotation or it is the lower node in a couple. (Intuitively, the staying 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 labeled staying local nodes in the truncated segment, following its traversal.

The topmost segment is said to be *incomplete* if it satisfies conditions (i)-(v) 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. Each marked couple is involved in a zig-zig rotation. For each type below, suppose the segment includes a couple, u, v, with u the parent of v.

Type 1. See Figure 1. Suppose that u is the root of v's block; then both u and v are marked.

Fig. 2.1. *Type 1 nodes*

Type 2. See Figure 2. Suppose that u is a local node and let x be the root of u's block. Further suppose that u is on the left (resp. right) abutting path for x. Let v be the left (resp. right) child of u; if u does not have a debit and if v is global then both u and v are marked.

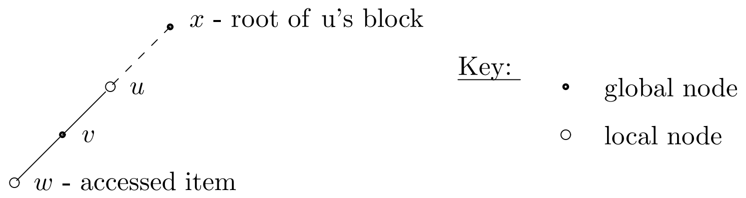

Fig. 2.2. *Type 2 nodes*

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 2.1. *(See [CMSS98], Lemma 2.1.) A truncated segment comprises at most 10 unmarked nodes, of which at most 7 are local.*

Over the course of the access we maintain the following invariant.

Invariant 7. *See Figure 3. Let w be the accessed item. Let x be w's right (resp. left) child. Suppose that x is on the inner skeleton of its superblock. Let P be the maximal portion of the extreme left (resp. right) path in x's block descending from x, contiguous in the splay tree. Let y be the bottom node on P (x is the top node). Then neither x nor y carry any debits. w does not carry a debit either.*

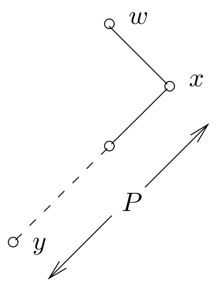

FIG. 2.3. *Invariant 7*

To initialize Invariant 7 at the start of the access may require the removal of up to five small debits or up to one large debit and three small debits; this costs at most

$$\max\{ld + 3sd, 5sd\} \tag{2.1}$$

As we will see, the access will remove the debits from all the traversed nodes (though it may also cause large debits to be placed on some of the traversed nodes).

To understand the cost of individual rotations is is helpful to consider when abutting paths affect Invariants 5 and 6. We define an *untouched* abutting path to be one whose top node is not traversed in the current access and whose top node was not a child of the accessed item at the start of the access; any other abutting path is said to be *touched*. We define an abutting path to be *clean* if it obeys Invariants 5 and 6 with respect to its parent, and to be *dirty* otherwise. Because of Invariant 7, a node that receives a touched abutting path receives a clean abutting path.

LEMMA 2.2. *Let u be a true global node. Let P be an abutting path. If P is dirty, u must be on the inner skeleton of its superblock.*

Proof. By definition, P is on the inner skeleton. By construction, every node on an extreme path of a superblock has larger global rank than any node on the inner skeleton. So for P to be dirty, u must be on the inner skeleton of its superblock. ∎

LEMMA 2.3. *Let u be a node on the access path. Suppose that before the rotation involving u, u was j-global and that after the rotation it is k-global, $k > j$. Then, after the rotation, u's abutting paths are clean.*

Proof. We note this must be a zig-zig rotation and u must be the lower node in the couple. So of the at most two abutting paths u acquires, one is touched; the untouched one, if present, had been an abutting path for its parent, t (t and u are in the same couple). The only node added to this abutting path is t, which by assumption carries no debit. u acquires t's k-global rank and so the untouched abutting path remains clean. ∎

LEMMA 2.4. *Let u be a node on the access path. Suppose that before the rotation involving u, u was j-global and that after the rotation it is k-global, $k < j$. Then, after the rotation, u's abutting paths are clean.*

Proof. u remains on the j-skeleton and hence is not on the inner k-skeleton. The result follows from Lemma 2.2. ∎

LEMMA 2.5. *Let u be a node on the access path. Suppose that both before and after the rotation u is a true j-global node. Further suppose that u is on the inner i-skeleton before the rotation, where $i > j$. Then either*

(i) u is on the inner j-skeleton after the rotation; then u's j-rank decreases.

(ii) u is not on the inner j-skeleton after the rotation; then u has no dirty abutting paths.

Proof. The conclusion of (i) follows from the definition of the j-ranks. The conclusion of (ii) follows from Lemma 2.2. ☐

COROLLARY 2.6. *Let u be a node on the access path. Following the rotation, u can have a dirty abutting path only if:*

(i) u was a true j-global node both before and after the rotation; also, u is on the inner skeleton of its j-superblock after the rotation. And,

(ii)

(a) Either u's j-global rank decreases.

(b) Or u receives a new untouched abutting path.

Proof. Condition (i) summarizes Lemmas 2.2–2.4. There are only two ways in which u can acquire a dirty abutting path. First, an already abutting path becomes dirty; for this to happen, u's j-rank must decrease. Second, u receives a new abutting path; in order for the new abutting path to be dirty it must be untouched (as already pointed out in the discussion following Invariant 7). ☐

Next, we investigate possibility (ii)b of Corollary 2.6 further.

LEMMA 2.7. *Consider the rotation involving couple u and v, where v is a child of u. Following the rotation, v does not have a new dirty untouched abutting path. u can receive a new dirty untouched abutting path only if both:*

(i) the rotation is a zig-zig rotation, and

(ii) v is in the same j-superblock as u but in a distinct j-block, where u is a true j-global node.

Proof. In a zig-zag rotation the new abutting paths are all touched.

If v acquires a new dirty abutting path, v is a true j-global node both before and after the rotation. Also, for u to be in v's new abutting path, u must be in the same j-block as v; but then v was not a true j-global node before the rotation. So v does not have a new dirty untouched abutting path.

If u acquires a new dirty abutting path, u is a true j-global node both before and after the rotation. Again, if v is in the same j-block as u, u is not a true j-global node after the rotation. So v is in a distinct j-block. Suppose that u is on the inner i-skeleton before the rotation, where $i \geq j$. If $i > j$, by Lemma 2.5, u is on the inner j-skeleton after the rotation, and so v must be in the same j-superblock as u. While if $i = j$, as u was on the inner i-skeleton before the rotation, v must be in the same i-superblock as u. ☐

LEMMA 2.8. *Let u be a node on the access path. If u acquires a dirty abutting path then u is in the leading couple of its segment. In addition, the visibility of w, the accessed item, is unchanged following the rotation involving u.*

Proof. If condition (ii)a of Corollary 2.6 applies, then clearly u is in the leading couple. So suppose that condition (ii)b applies. Consider Lemma 2.7; let v be the other node in u's couple (v is u's child). v is in the same j-superblock as u but in a distinct j-block. Also, the rotation involving u and v is zig-zig. There are two possibilities. First, v is j-global; then the rotation must reduce u's j-global rank or increase w's j-global rank; in either event u is in the leading couple. Second, v is not j-global; in order for v to be in a distinct j-block from u, w cannot be on the inner skeleton of B, u's j-superblock. So this rotation raises w's visibility, violating condition (i) of the definition of a truncated segment; again, u must be in the leading couple. Finally, suppose w's visibility increases due to the rotation involving u. Then, u becomes a true k-global node, $k < j$, after the rotation; by Lemma 2.5, u does not acquire any dirty abutting paths. ◻

LEMMA 2.9. *Following the access, the accessed item's abutting paths are clean.*

Proof. This is an immediate consequence of Lemma 2.2, for the accessed item is not on the inner skeleton of any of its blocks. ◻

In the analysis of global accesses, 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 accesses.

In order to pay for the rotations we provide the following potential. In each rotation, for each unit increase in rank on the part of the accessed item, w, we provide $3gp$ units of potential. It is used as follows. Consider a rotation involving couple u, v, where u is the parent of v. Suppose that w is currently the root of its block. Suppose that this rotation increases the global rank of w by I. Each of w and v (if global and in the same superblock as w) may increase their global potentials by up to $gp \cdot I$ units; this is a total of at most $2gp \cdot I$ units of potential. If $I > 0$, the remaining at least gp units of potential at hand will pay for the segment that ends at couple u, v (as we will see, gp units always suffice). When w becomes the root of its superblock it interchanges potentials with the old root of the superblock (it may be that in one double rotation w in turn swaps potentials with v and then with u).

Further potential will also be provided; a total of $gp + vis \cdot vp + (ld + 3sd)$ units, vp being a constant to be specified (the role of the latter potential will become clear towards the end of this section).

Next, we show how to pay for the rotations (and associated spares) along a segment. First, each marked couple pays for its rotation (and spares), as follows. Recall that the marked couples are all involved in zig-zig rotations.

CASE 1. (Type 3.) The coupled nodes are both local nodes in the same i-superblock (see Figure 4). Nodes u and v are on the inner skeleton in the same superblock; without loss of generality, v is the left child of u, and w, the node being accessed, is the left child of v. Node u is j-global and node v is k-global, $j, k < i$. There are several subcases depending on the relative values of j and k.

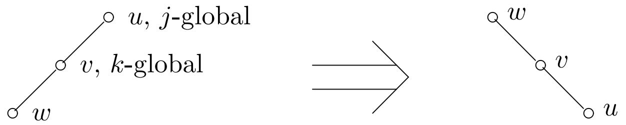

FIG. 2.4. *Type 3 nodes*

CASE 1.1. $j > k$. After the rotation, v becomes j-global and u becomes k-global. u and v exchange potentials. Since v is adjacent to a j-global node it must be h-labeled, for $k < h \leq j$; it transfers the h-labels and associated potentials to node u. u is given a small j-debit; as it is now a j-local node on an inner j-skeleton, this obeys Invariant 3. By Lemma 2.2 and Corollary 2.6, Invariants 5 and 6 continue to hold. The small debit pays for the rotation and s spares. Thus it suffices that:

$$sd \geq s + 1 \tag{2.2}$$

Next, we remove the debits present at u and v, if any. If both nodes are i-labeled, then u spends its associated potential c; this pays for the removal of debits from both nodes. If only one node is i-labeled, as we show later, the traversal of a portion, or all, of the i-unlabeled path associated with the i-unlabeled node results in the i-labeling of that i-unlabeled node. So we proceed in the same way in this case. If neither node is i-labeled then they do not carry debits. So it suffices that:

$$c \geq \max\{2sd, ld\} \tag{2.3}$$

CASE 1.2. $j < k$. Then u and v are in the same k-superblock, but are in distinct k-blocks. Here, v remains k-global and u j-global. u is given a small k-debit; again, as u is now a k-local node on an inner k-skeleton, Invariant 3 is satisfied. Again, by Lemma 2.2 and Corollary 2.6, Invariants 5 and 6 continue to hold. Here too, Equation 2.2 suffices. The removal of debits is handled as in the previous case.

CASE 1.3. $j = k$. The j-rank of u decreases, and this pays for the removal of small or large debits from u and v, and of up to two small debits from nodes which now violate Invariants 5 and 6. So it suffices that

$$gp \geq \max\{2sd, ld\} + 2sd \tag{2.4}$$

CASE 2. (Type 1.) u is i-global and v is j-global, $j < i$ (see Figure 1). This is similar to Case 1.1, above. However, here v cannot have a debit before the rotation, by Invariant 5, since $g\_rank(u) = g\_rank(v)$ (this follows from the fact that the global rank of the accessed item does not change). Here u is given a small i-debit, for it remains on the inner i-skeleton. Note that following the rotation, $g\_rank(u) < g\_rank(v)$. Invariants 3, 5 and 6 continue to hold.

CASE 3. (Type 2.) v is i-global and u is j-global, $j < i$ (see Figure 2). This is similar to Case 1.2, above. But here, u does not have a small debit by assumption (since the couple was marked). It is given a small debit following the rotation. As before, Invariants 3, 5 and 6 continue to hold.

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)-(v), except in the case of an incomplete topmost segment, which is handled subsequently. The rotation involving the first couple falls into at least one of the following cases.

CASE A. The rotation reduces the total global potential of the nodes in the first couple, or it increases the global rank of the accessed node.

CASE B. The rotations along the segment creates a sequence of three contiguous local nodes from the same block.

CASE C. The rotation increases the visibility of the accessed item.

Each case involves three costs.

*Cost*1. The rotation and spares for each unmarked couple, including the leading couple: $\leq 6(s+1)$.

*Cost*2. Removal of small debits from the unmarked couples; *Cost*2 is analyzed below.

*Cost*3. Removal of small debits for local nodes that now violate Invariant 5 or 6; *Cost*3 is analyzed below.

We remind the reader that we remove all debits from the traversed nodes (though we may add new debits).

LEMMA 2.10.

(i) For each segment in which the visibility of the accessed item is unchanged, $Cost2 + Cost3 \leq 11sd$.

(ii) For all other segments, $cost2 + Cost3 \leq 7sd + \max\{2sd, ld\}$

Proof. Lemma 2.10(i) is proven in [CMSS98], where it is Lemma 2.2. To prove (ii), we note that by Lemma 2.1, the truncated segment contributes at most $7sd$ to $Cost2$. By Lemma 2.8 it contributes nothing to $Cost3$. The removal of debits from the leading couple costs at most $\max\{2sd, ld\}$. Again, by Lemma 2.8 there is no contribution to $Cost3$. □

LEMMA 2.11. *Invariant 7 remains true throughout the access.*

Proof. This follows immediately from the fact that the traversed nodes have no debits following their traversal. Thus w's children do not carry debits (the node x of Invariant 7). Consider node y; this node changes only when the new node x has a larger visibility than the old node x; but then the new node y is the new node x. □

LEMMA 2.12. *A global access costs at most* $(3\sum_{i=1}^{vis} k_i + 1)gp + vp \cdot vis + 2c \cdot vis + (ld + 3sd)$ *units, where* $sd = s + 1$, $ld, c = 17(s + 1)$, $gp = 19(s + 1)$, $vp = 30(s + 1)$, *vis is the number of levels of blocks, and k_i is the i-rank of nodes on an $i+1$)-extreme path. (Note* $\sum_{i=1}^{vis}(k_i + 1) \le \log n + 3vis + 1$.*)*

Proof. 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 for the accessed item, by gp units of the at least $3gp$ units provided for this rotation. In case B there are at least three labeled staying local nodes; the middle node among these three labeled nodes is given a large debit, which pays for the rotation (note that the three labeled nodes need not be contiguous but they all lie on a contiguous path on the skeleton; the middle node obeys Invariant 4). In Case C, the cost of the rotation is charged to the new visibility level. (vp units are provided per level for a global access to pay for each instance of Case C.)

Thus it suffices to have:

$$gp, ld \ge 6(s + 1) + 11sd \tag{2.5}$$

$$vp \ge 6(s + 1) + 7sd + \max\{2sd, ld\} \tag{2.6}$$

Recall that an i-superblock comprises b_{i+1} i-blocks; let k_{i+1} denote the i-rank for nodes on an $(i + 1)$-extreme path. Clearly, $\sum_{i=1}^{vis} k_i \le \log n + 2vis + 1$; with $b_i = 2^{2^i}$, we get $\sum_{i=1}^{vis} k_i \le \log n + vis + 1 \le 2 \log n$. We use $3(k_{i+1} + 1)gp + vp$ units of potential to pay for rotations in which the accessed item traverses the inner skeleton of its i-superblock and for which, on completion, the accessed node has reached an extreme path of its i-superblock. The $3(k_{i+1} + 1)gp$ units pay for rotations in which the i-global rank of the accessed item increases, as explained earlier. The vp units pay for the rotation in which the accessed item reaches an extreme path of its i-superblock. Overall, we need

$$\sum_{i=1}^{vis}(3(k_i + 1)gp + vp) + \max\{ld + 3sd, 5sd\} \tag{2.7}$$

units to pay for global accesses so far (see Equation 2.1).

Now, we show how to pay for the incomplete segment, if present. We provide an additional gp units to pay for this segment; in addition, this term is used to pay for the incomplete rotation, if any; however, the additional gp term does not need to account for any increase in global rank, on the part of the accessed item, during the incomplete rotation, for this has already been accounted for. The result of Lemma 2.10 applies here too (in fact, a tighter bound can be shown). Here too, Equation 2.5 suffices.

We have yet to explain how to maintain the potential of an unlabeled path and the invariants concerning labels, Invariants 1 and 2. If a maximal unlabeled path is traversed in its entirety and in addition the child, on the extreme path, of this unlabeled path, is also traversed, then the two old endpoints of the path are labeled (or rather, if an old endpoint is in a couple with another unlabeled node, then the new endpoint is given the label) and the remainder of the path is essentially halved in length; the sufficiency of the potential is readily verified (for a path of length at most 4, the potential suffices to label each node on the path; for a path of length $2k+2$, $k>2$, we have $2\lfloor\log(2k+3)\rfloor \geq 2+2\lfloor\log(k+1)\rfloor$, and for a path of length $2k+3$, $k \geq 2$, we have $2\lfloor\log(2k+4)\rfloor \geq 2+2\lfloor\log(k+2)\rfloor$).

If only a portion of an unlabeled path is traversed then the path was accessed through an increase in the visibility of the accessed item. Consider an unlabeled path that is part of a left extreme path of its block (the case of a right extreme path is analogous). At most two unlabeled nodes need to be labeled to maintain Invariants 1 and 2, namely the new top of the unlabeled path and the old root of the block. This costs at most $2c$ units. This charge is levied once for each unit increase in visibility on the part of the accessed item. Over the whole access this is a cost of $2c \cdot vis$ units.

On taking equalities in Equations 2.2-2.6, the lemma follows from Equation 2.7. □

Unfortunately, the potential, as described, may be of size $\Theta(n \log\log n)$ initially. (For example, consider a zig-zag path descending from the root.) To reduce this to linear size, we modify the rules for creating unlabeled paths, and modify the analysis accordingly.

For each block, each of its extreme paths is partitioned into chains. A chain for block B, at level i, comprises a maximal sequence of nodes, $v_1, v_2, \cdots, v_k$, from B, such that the path P in the splay tree from v_1 to v_k does not include any j-global node, $j \geq i$. Each chain, C, is given a path potential of size $2c\lfloor\log(|C|+1)\rfloor + c$ units and a reserve path potential of $2 \cdot 2c\lfloor\log|C|\rfloor$. The role of the reserve path potential will become clear in Section 4. In addition, each node adjacent on P to a node that is not part of the chain is given an additional *zig-zag* potential of c units; this potential is provided once, and not for each block to which the node belongs.

Lemma 2.13. *The cost of the path and reserve path potentials is at most*

$$3\sum_{j=1}^{vis}\lceil\frac{n}{c_j}\rceil 4c[\lfloor\log c_i\rfloor + 3\sum_{i=1}^{j} 4c\lfloor\log c_i\rfloor] + 2c\cdot(j+1)$$

This is bounded by $144c \cdot n$ if a j-block has size $c_j = 2^{2^{j+1}}$, for $j \geq 1$ (with the possibility of one smaller block at each level j).

Proof. The path potential for each chain, apart from the bottommost chain on each extreme path, is charged to the j-global node terminating the chain at the bottom. The remaining at most two chains in each block are charged to the block itself. Each j-global node is charged for at most two chains on the extreme paths of i-blocks, for each $i \leq j$. A chain of an i-block has length at most $c_i - 1$, so the path potential for an i-chain is at most $2c\lfloor\log c_i\rfloor + c$. Hence the charge to a j-global node is bounded by $\sum_{i=1}^{j} 12c\lfloor\log c_i\rfloor + 2c$. Likewise, the charge to a j-block is bounded by $12c\lfloor\log c_j\rfloor + 2c$. Summing over all j-global nodes, j-blocks, and all j, gives a total charge of $\sum_{j=1}^{vis}\lceil\frac{n}{c_j}\rceil 12c[\lfloor\log c_j\rfloor + \sum_{i=1}^{j}\lfloor\log c_i\rfloor] + 2c\cdot(j+1)$.

For $c_i = 2^{2^{i+1}}$ this is bounded by $\sum_{i=1}^{vis}\lceil\frac{n}{c_j}\rceil 72c\cdot 2^j + 2c\cdot(j+1)$ which is bounded by $72c\cdot(n+2\log n) + c\cdot(vis+1)(vis+2) \leq 144c\cdot n$, assuming $n \geq 16$ ($vis = \log\log n$).

For $n < 16$, $vis = 1$, so it suffices to give potential c to each node on an extreme path of the splay tree for a total of cn potential. □

LEMMA 2.14. *A global access costs at most* $(3\sum_{i=1}^{vis}(k_i+1)+1)gp + vp \cdot vis + 2c \cdot vis + (ld+3sd)$ *units, where* $sd = s+1$, $ld, c = 17(s+1)$, $gp = 19(s+1)$, $vp = 30(s+1)$, *vis is the number of levels of blocks, and* k_i *is the* i*-rank of nodes on an* $(i+1)$*-extreme path. (Note* $\sum_{i=1}^{vis}(k_i+1) \le \log n + 3vis + 1$.*) In particular, maintaining the chain potentials requires a charge of at most* $2c$ *for each unit increase in visibility on the part of the accessed item.*

Proof. It suffices to explain how to modify the analysis of Lemma 2.12 to account for chains. There are two ways a chain can be traversed: in its entirety or in part. We consider each in turn.

If a chain is traversed in its entirety, following the traversal it forms a path for which the associated path potential is sufficient to allow the previous analysis to be performed; indeed this path potential is too large by c units. These c units are used to label the old root of the block if the root of the block is changed by the traversal. All the couples which include just one node of the chain are paid for by the associated zig-zag potential of c units, or to put it in terms of marking, these couples are marked.

Next, consider a chain which is only partially traversed. Suppose, without loss of generality, that this chain is on the right extreme path of its block. Let v_j be the bottommost node on the chain to be traversed. If v_j is accessed from its left child, then the rotation involving v_j results in an increase in the visibility of the accessed item. This increase in visibility pays for the rotation involving node v_j. Again, the old root of the block may need to be labeled, which costs c units; this is also charged to the increase in visibility. The path potential is associated with the remainder of the chain on and below node v_j. The rotations along the traversed portion of the chain are paid for as in the traversal of the full chain. In addition, the nodes on the traversed portion cease to be on an extreme path of their block and so do not need an associated path potential on completion of the traversal. There may be several chains including v_j which are partially traversed, but the number of these chains is bounded by the increase in visibility. More precisely, let v_j be h-global, and suppose it is on an inner i-skeleton. Then every chain including v_j but not ending at v_j must be on an extreme path of an l-block, $h < l \le i$; thus there are at most $i - h$ such chains. The increase in visibility for the accessed item, in its rotation with v_j is at least $i - h$. Hence each unit increase in visibility is charged c for the labeling of the root of one partially traversed chain of the type described in this paragraph.

Finally, suppose v_j is accessed from its right child. Then there is a node w on the path between v_j and v_{j+1} which is accessed from its right child. The rotation involving w results in an increase in the visibility of the accessed node. This increase in visibility is charged for the additional potential of c units needed to label the old root of v_j's block. Again, the rotations along the traversed portion of the chain are paid for as in the traversal of the full chain. The path potential continues to be associated with the remainder of the chain starting at node v_{j+1}. We note that there may be many chains including nodes v_j and v_{j+1}, each on an extreme path for a different level of block. But the level of node w must be smaller than that of these blocks. To be specific, let w be h-global. Then every chain including v_j and and v_{j+1} must lie on an l-block extreme path, with $h < l$. Suppose further that v_j and v_{j+1} are on an inner i-skeleton. Then the increase in visibility of the accessed item in rotation with node w is at least $i - h$, and there are at most $i - h$ chains straddling w, each of which may require a charge of c to label its old root.

Thus an increase in visibility which places the accessed item on the inner skeleton of an i-superblock results in a charge of up to $2c$ to visibility level i. $\square$

REMARK 1. Later, the form of the blocks will be generalized to allow more complex situations involving local nodes. Specifically, we will allow local nodes to carry other debits. For the above analysis to continue to apply, it will suffice that

(i) If a local node on its block's right (resp. left) path carries a new debit then its parent and right child (resp. left child) are both local nodes of the block.

(ii) A couple containing two local nodes must pay for the removal of all debits on the couple's nodes; the $s+1$ rotations will be paid for as before.

3. Local Insertions. For each level i of blocks, the access sequence S is partitioned into maximal subsequences S_l; each subsequence comprises accesses in the same i-block. Recall that an $(i+1)$-block contains b_{i+1} i-blocks and c_{i+1} items. We allocate amortized time $O(\log b_{i+1} + |S_l|\alpha(c_{i+1}))$ to pay for the cost of the sequence S_l. Specifically, the first access to an i-block has amortized cost $O(\log b_{i+1})$ and subsequent accesses have amortized cost $O(1+\alpha(c_{i+1}))$. Note that an access occurs in vis sequences; thus the cost of an access comprises the sum of vis terms. So suppose that an access is to a new i-block, but to the same $(i+1)$-block, with respect to the previous access. Then the cost of the access is

$$O(\sum_{h=0}^{i} \log b_{h+1} + \sum_{h=i+1}^{vis} [1+\alpha(c_{h+1})]) = O(\log b_{i+1} + (\log\log n)\alpha(n)) \tag{3.1}$$

(assuming $b_i = 2^{2^i}$ which implies $vis = \lceil \log\log n \rceil$). In Section 4, we show how to reduce the $O((\log\log n)\alpha(n))$ term to $O(1)$.

It is possible to have an access close to the previous access be in a different high level block (if both accesses are close to the boundary between the blocks). In Section 3.7, we will see how to reduce this excess cost and obtain an amortized time for the access proportional to the logarithm of the distance between successive accesses (plus an additional $O(\log\log n\alpha(n))$ term). For now, we focus on the cost of the access at a single level, i. To be specific, we focus on the rotations during which the accessed item is on the inner $(i+1)$-skeleton.

In the present analysis spare rotations will be spent in several ways. We spend the spare rotations in portions of size s', s' a constant to be specified. Recall that s spares are provided per couple. As we will see in Section 4, it suffices to have $s \geq 3s'$.

3.1. Forming Lazy Trees. As in the analysis of splay sorting $\log n$-block sequences, we construct lazy trees. However, their form is more involved here.

For each level of blocks a separate analysis is performed, as in the analysis of global accesses. Here we focus on the traversal of the inner skeleton of superblock B, an $(i+1)$-block. For each item, the analysis of its traversal of the inner skeleton of superblock B is performed as part of the analysis of a maximal sequence S of accesses to the same block B' of superblock B.

We start with a few definitions. Let b be the number of blocks in superblock B and let c' be the number of items in B. Following the first access of S, we define the *right access path* as follows: Let $right\_B$ denote the set of elements in superblock B to the right of block B'. Consider the tree induced by the nodes of $right\_B$; its left extreme path is the *right access path* (see Figure 5). The *left access path* is defined analogously. In the splay tree, an access path may comprise more than one contiguous segment of nodes. Note that as the accesses of sequence S proceed, the access paths become shorter. Lazy trees are formed from the nodes originally on the access paths.

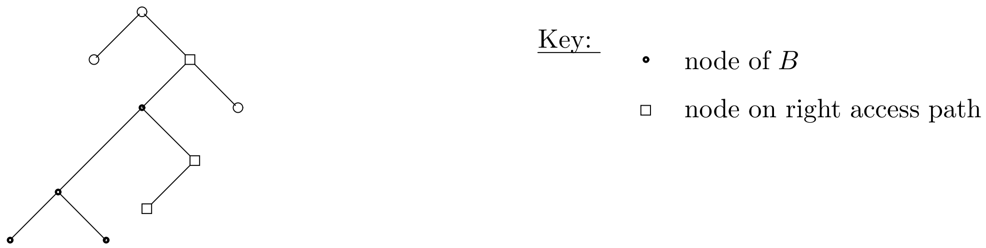

FIG. 3.1. *The right access path*

Let $l\_child(v)$ denote the left child of v. For each global node, v, on the right access path, other than the bottom global node, define $jump(v) = g\_rank(v) - g\_rank(l\_child(v))$; for the bottom global node, define $jump(v) = g\_rank(v) + 1$. $jump(v)$ is defined analogously for global nodes on the left access path.

Consider an access in sequence S; there are three possibilities. First, no node on either access path is traversed, in which case no cost is incurred for the traversal of the inner skeleton of superblock B. Second, the right access path (or rather a top portion of it) is traversed. Third, a top portion of the left access path is traversed.

Our goal is to accumulate spares on the nodes of the access paths, so that following e' accesses we can assume that each node on the access paths, apart from the top node, has accumulated at least $2^{e'-1} - 1$ spares, e' a constant to be specified. We need to be careful, however; for higher portions of the access paths may have been traversed more frequently and so have accumulated more spares. We proceed as follows.

It is convenient to ensure that on each traversal, for each couple, its two nodes have been traversed the same number of times. This need not be true. So we introduce *pseudo-traversals*. Initially, each node on an access path has been pseudo-traversed zero times. Each time a node is traversed, its pseudo-traversal count is incremented, up to a maximum of e' pseudo-traversals. In addition, immediately prior to each access of sequence S, apart from the first access, for each couple in the upcoming access, for each lower node in the couple, its pseudo-traversal count is increased to the count of the higher node. We note that the pseudo-traversal counts are non-decreasing from bottom to top of the access paths (for each time a node is traversed so are all its ancestors on its access path).

Immediately following the first access in S, we provide each global node, v, on an access path with *jump potential* $2gp \cdot e' jump(v)$. Summing over the two access paths, we obtain a cost of $4 \cdot e' gp(\log 2b + 1)$ units charged to the first access of sequence S; the first access itself costs a further $3gp(\log 2b + 1)$ units. As further traversals of the access paths occur, Invariants 8 and 9 below are maintained.

INVARIANT 8. *Each global node, v, on an access path, apart from the first node, carries a jump potential of $2gp \cdot (e' - t)jump(v)$ after having been pseudo-traversed $t \leq e'$ times.*

INVARIANT 9. *Each node on an access path, global or not, apart from the first node, carries a spare potential of $s' \cdot (2^{(t-1)} - 1)$ units after having been pseudo-traversed $t \leq e'$ times.*

Clearly the invariants are true following the first access of sequence S.

First, we explain how to maintain the invariants following the increase in pseudo-traversal counts (immediately prior to an access). Clearly, Invariant 8 is unaffected for the jump potentials can only decrease. To maintain Invariant 9, an additional spare potential of $s' \cdot (2^{q-1} - 2^{p-1})$ is provided to a node whose pseudo-traversal count increases from p to q. This is a total of at most $s' \cdot 2^{e'-1}$ spare potential per access for the pseudo-traversal counts are non-decreasing along the access path (from bottom to top).

Next, we show how to pay for an access and how to restore Invariants 8 and 9 following the access. Consider one couple of this access, where the couple contains nodes u and v, and u is the parent of v. Further suppose that at least one of u and v is on the access path.

The rotation is paid for as in a global access, except that the jump potentials provide the additional potential needed to pay for the segment associated with a rotation which increases the global rank of the accessed item. There is one special case: if the rotation increases the visibility of the accessed item, then the increase in visibility is charged for the segment associated with the rotation, and the increase in the global rank of the accessed item is not charged. In addition, we are no longer seeking to provide a global potential to the accessed item, w. For w eventually acquires the global potential of the old root, x, of w's block, and x then acquires the sub or mini potential of w. More precisely, it will suffice, at each global node, y, on the access path, to provide $2gp \cdot jump(y)$ units of potential, taken from y's jump potential.

It remains to explain how Invariants 8 and 9 are restored following the rotation. We start with Invariant 9. u gives v all its spare potential; in addition, v receives s' of the spares allocated to the couple u, v. This gives v spare potential $s'[2 \cdot (2^{(t-1)} - 1) + 1] = s'(2^t - 1)$, as needed, since t will be incremented.

Invariant 8 needs restoring only if both u and v are global nodes; then u gives v its remaining jump potential. To show this suffices we argue as follows. Without loss of generality, suppose that both u and v are on the right access path. First, we note that $g\_rank(l\_child(v))$ is unchanged following the rotation involving v (for w, the accessed item, is in block B' and so its left subtree contains no global items, while w's right subtree remains a subtree of v). Likewise, $new\_g\_rank(v)$, $g\_rank(v)$ following the rotation involving v, is exactly $g\_rank(u)$ before this rotation. So $new\_jump(v)$, the value of $jump(v)$ following the rotation, is given by $g\_rank(u) - g\_rank(l\_child(v))$, which is $[g\_rank(u) - g\_rank(v)] + [g\_rank(v) - g\_rank(l\_child(v))]$, or $jump(u) + jump(v)$, as claimed. Hence, following the rotation, v has jump potential $2gp \cdot (e' - (t+1))jump(v)$ (using the new value for $jump(v)$); as t is incremented following the rotation, Invariant 8 continues to hold.

Once a node on an access path, other than the top node, has been pseudo-traversed e' times, and hence carries spare potential $s'(2^{e'-1} - 1)$, when it is rotated off the access path it becomes a lazy node and is given q_1 units of potential, q_1 a constant to be specified. In addition, if both the nodes u and v in the couple are global, they swap their global potentials; the potential now carried by u, the node leaving the access path, is called its *lazy potential.* Each couple pays for its rotation, spare rotations and associated segment charge, if any, by a charge of gp units to the spare potential of the node leaving the access path, instead of by the creation of debits. So it suffices to have

$$s'(2^{e'-1} - 1) \geq gp + q_1 \tag{3.2}$$

When a rotation removes the top node of an access path it too becomes a lazy node; however it is given $a_3\alpha(c') + b_3$ potential, where a_3 and b_3 are constants to be specified. So following the first access of sequence S, the charge per access for the traversal of nodes on the inner skeleton of block B is bounded by $s' \cdot 2^{e'-1} + a_3\alpha(c') + b_3$.

We note that a lazy node, on creation, carries no debit. For the rotation creating the lazy node is paid for by the gp charge to the node's spare potential (see Equation 3.2).

In fact, we provide another $2gp \cdot (\log 2b + 1)$ units to the first access of sequence S. This is used as follows. For each global node, u, remaining on an access path, as soon as u acquires a proper global descendant, v, on the access path, which has been pseudo-traversed e' times, we provide u a *reserve potential*, defined as follows. Define the *marker_rank* for a global node, v, on the access path to be the global rank of the node following exactly e' pseudo-traversals. Define the *reserve_rank* for global node u to be: $reserve\_rank(u) = marker\_rank(u) - marker\_rank(v)$, and the reserve potential for u to be gp times its reserve_rank. The role of the reserve potential will become clear later.

As in the analysis of a global access, the cost of the segment that involves an increase in visibility on the part of the accessed item is at most $vp+2c$, per unit increase in visibility. The presence of lazy trees, as we will see, adds a further $4gp(\log 2b + 1)$ to the cost of the first access of sequence S. So, focusing on superblock B alone, the cost of the first access in S is bounded by

$$(4e' + 9)gp(\log 2b + 1) + vp + 2c \tag{3.3}$$

and the cost of a subsequent access of S is bounded by

$$s' \cdot 2^{e'-1} + a_3\alpha(c') + b_3 + vp + 2c \tag{3.4}$$

Also, for each access, there are fixed costs of (see Equation 2.1):

$$ld + 3sd \tag{3.5}$$

Each node removed from the access paths is called a *lazy node*, whether or not it has a lazy potential. Those lazy nodes with a lazy potential are called *heavy* nodes; the other lazy nodes are called *light* nodes. When the accesses of sequence S are completed we form *lazy trees*. Each global node v remaining on the right access path becomes the root of a new *right lazy tree*. Its left subtree is empty; its right subtree comprises the lazy nodes, created during the accesses of sequence S, in v's right subtree in the splay tree. An analogous definition is made with respect to the left access 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 *initial lazy trees*. (We will be adding and removing a few nodes from the initial lazy tree in order to obtain other lazy trees, which are the lazy trees that are actually analyzed.) As we will see, the local nodes on the access paths, which have been pseudo-traversed e' times, may be added to the new lazy trees; these nodes too are called light nodes of their lazy trees. We will need to ensure these nodes also carry no debits and have an associated potential of q_1 units. But this is ensured by requiring:

$$s'(2^{e'-1} - 1) \geq q_1 + \max\{sd, ld, hd, ed\} \tag{3.6}$$

where hd and ed are the values of other debits (huge and enormous debits) introduced later.

Remark 2. We will need to generalize this analysis to take account of the presence of lazy trees on the access paths. The present analysis, with Equation 3.2 replaced by Equation 3.7 below, will continue to apply if the following conditions hold.

(i) Each node on the access path, which has been pseudo-traversed e' times, has spare potential $(2^{e'-1} - 1)s$.

(ii) A lazy tree node, on creation, carries no debit.

(iii) Following the first e' pseudo-traversals of a node v on the access path, if v's couple includes just one node on the extreme path of a lazy tree, the cost of the rotation of this couple is the same as in the present analysis.

(iv) Following the first e' pseudo-traversals of a node v on the access path, if v's couple includes two nodes on the extreme path of a lazy tree, the cost of the rotation of this couple is at most $\max\{hd, ed\}$ units. The additional cost of the couple is covered by the spare potential if:

$$s'(2^{e'-1} - 1) \geq gp + q_1 + \max\{hd, ed\} \tag{3.7}$$

(i) and (ii) will be achieved as here. (iii) will be seen to be true shortly. (iv) will be handled later, in Lemma 3.1.

Next, we describe the potential provided to the lazy trees and show how to include the lazy trees in the overall analysis.

3.2. The Analysis of Lazy Trees. In the following subsections, unless we specify otherwise, the definitions given apply to right lazy trees. Analogous definitions hold for left lazy trees. We focus on the right lazy trees; we discuss the left lazy trees only where their presence affects the analysis (the only place this arises is in Section 3.6).

Much of the analysis focuses on a subtree of the initial lazy tree, called the *truncated lazy tree* or the *lazy tree*, for short. It is defined as follows. Consider a new initial lazy tree, L, created by the sequence S of accesses. Consider the set of heavy nodes in L; the tree they induce is called the *initial lazy block tree*. We remove the rightmost node from the initial lazy 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 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) lazy block tree together with the following light nodes. For each node v in the (truncated) lazy block tree we add the following nodes from v's block to the (truncated) lazy tree. 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 light nodes added to the lazy tree form its *skeleton*.

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 portion of the right access path which has been traversed at least e' times, 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 for one lazy tree and a left guard for a second lazy tree.

Intuitively, a truncated lazy tree is a megablock comprising several of the blocks at hand. The megablock is considered to be at level $i - 1$, the level of the blocks it comprises. In its interactions with the remainder of the splay tree the megablock will behave in the same way as a block. The root of the lazy tree corresponds to the block root and behaves like a global node; the other nodes of the lazy tree, called *local* nodes of the lazy tree, correspond to the local nodes from the block. The lazy tree local nodes may carry small and large debits according to the invariants specified for blocks; a further *lazy* debit may be carried as specified later.

The right guard of each new lazy tree has its global potential restored; this is paid for by adding its reserve potential to its lazy potential, which suffices by Invariant 24, in Section 3.5. In addition, Invariants 5 and 6 may need to be restored; this requires the removal of at most two small debits. These are paid for by the potential $a_3 + b_3$ associated with each lazy node on paths P_1, P_2, P_3 (see Sections 3.3.3 and 3.3.4); these paths always include the right guard. Thus both the left and right guards of each lazy tree carry their global potentials.

We need one more definition: a *rightist tree* is a tree in which the depths of the leaves, from left to right, are non-decreasing. Now, we describe the structure of a typical lazy tree; Consider a newly constructed initial right lazy tree, IL, and the associated lazy tree L. The main difference between L and IL is that IL may have (many) light nodes which are not in L. The tree induced by the nodes of L comprises its left extreme path (a sequence of light nodes), its root, and a path, P, descending to the right, together with the left subtrees of the nodes on P; each of these left subtrees is a rightist tree (strictly speaking, if v is the root of such a left subtree, then the tree rooted at v in IL is a rightist tree). It is helpful to partition P, the left extreme path and the root into four portions, BL, P_1, P_2 and P_3. BL comprises the light nodes on the skeleton from the leftmost block in L (initially, BL comprises the light nodes on the left extreme path); P_1 is the maximal top contiguous portion of P incident on the root of the lazy tree, minus those nodes in BL, P_2 is the second highest contiguous portion of P below P_1, and P_3 is the remainder of P below $P2$. It is also helpful, for each subtree of P, a rightist tree, R, to partition R into its top right path RP, and the left subtrees of RP, also rightist subtrees (again, strictly speaking, it is the corresponding subtrees in IL which are rightist); let SR denote a typical such rightist subtree (*small rightist subtree*). Now, to analyze a lazy tree, we analyze rotations in each of its six types of components separately; the six types of components are:

(i) BL, comprising nodes of type 1.
(ii) The path P_1, comprising nodes of type 2.
(iii) The path P_2, comprising nodes of type 3.
(iv) The path P_3, comprising nodes of type 4.
(v) Each path RP, comprising nodes of type 5.
(vi) Each rightist subtree SR, comprising nodes of type 6.

The following invariant describes the distribution of the six types of nodes.

INVARIANT 10. *Let L be a right lazy tree. Let v be a node in L. The following nodes all have type equal to or less than type(v):*

(a) The nodes on the left extreme path of L, other than v's ancestors.
(b) The left ancestors of v in L, apart from the root of L.

In particular, traversing the extreme paths of the lazy tree, from the bottom of the right extreme path to the bottom of the left extreme path, including the root of the lazy tree, yields, in order:

(i) A path from a tree SR.
(ii) A portion of a path RP.
(iii) A portion of P_3.
(iv) A portion of P_2.
(v) A portion of P_1.
(vi) A portion of BL.

The portions need not be contiguous portions of the original paths and some or all of the portions may be empty. There are only two ways in which the type of a node can change: first, a node can leave the lazy tree; second, when the lazy tree is split, a node may move into a component BL.

The structure of a left lazy tree is entirely analogous.

Our analysis is presented in three parts. First, for each component of the lazy tree, we describe the potentials associated with its nodes. Second, we analyze a traversal of an extreme path of a lazy tree. Third, we analyze a split of a lazy tree. For the moment, this discussion will exclude the possibility of multiple lazy trees contained within one and another; in Section 3.6 we show how to accommodate this possibility.

Before entering into the details of the potentials it is helpful to introduce the normal form of a (right) lazy tree L. It is obtained by performing the following series of single rotations: one by one, the left path nodes are moved to the right path; each such rotation, between node v and node r, the root of the lazy tree, makes v the root and places r on the right path. r and v interchange potentials and all lazy tree properties. The resulting tree is called the *normal tree* for L.

3.3. The Potentials for the Components of the Lazy Tree.

3.3.1. The Rightist Subtrees SR. We use the potential defined for the lazy trees present in the result on splay sorting $\log n$-block sequences, where it is called the *lazy complete tree potential*. To define this potential we need to introduce the *component lazy block tree*. It is the tree induced by the heavy nodes in the component SR together with the root of the lazy tree. A node of height h in the component lazy block tree, other than the root, is given potential $\frac{1}{2}a_1\sum_{i=1}^{h} i(i+1)+b_1$, a_1 and b_1 being constants to be specified, while a light node in the subtree SR is given potential c_1, c_1 being another constant to be specified.

Now, we verify that there is enough potential at hand to initialize the trees SR with their correct potential. Let ISR denote the subtree in the initial lazy tree corresponding to SR (the subtree with the same root as SR). We start by giving a node of height k in ISR potential $\frac{1}{2}a_1\sum_{i=1}^{k} i(i+1)+b_1$, regardless of whether the node is heavy or light. Since, for each heavy node, its height in ISR is at least as large as its height in the component lazy block tree, the heavy nodes will receive sufficient potential. To ensure adequate potential for the light nodes (potential c_1) it suffices to ensure

$$c_1 \le a_1 + b_1 \tag{3.8}$$

We explain how to provide the initial potential of $\frac{1}{2}a_1\sum_{i=1}^{k} i(i+1)+b_1$ to the nodes of ISR. We exploit the fact that the path IRP (the right path in IL corresponding to RP) together with its left subtrees is also a rightist subtree. A non-leaf node, v transfers all but b_1 of the charge for its potential to the following node, w, to its right (see Figure 6): Suppose v's first right ancestor, u, is reached by following e edges to the left and then an edge to the right. Then w is defined to be the descendant of u reached from u's right child by following $e+1$ left edges (edges to left children). The charge, $\frac{1}{2}a_1h(h+1)$, for a node, v, at height h, is transfered to a node, w, at *minimum height* at least $h-1$, where the minimum height of a node is the length, in vertices, of the path to its leftmost descendant. Thus a node, v, at minimum height k receives a charge of at most $\frac{1}{2}a_1(k+1)(k+2)$. This charge can be distributed evenly among the nodes by passing a charge of $a_1\cdot(\frac{1}{2}k^2+\frac{5}{2}k-\frac{7}{2})$ to each of v's children, if v has minimum height greater than 1 (note that v will receive a similar charge of at most

$a_1 \cdot (\frac{1}{2}(k+1)^2 + \frac{5}{2}(k+1) - \frac{7}{2})$ from its parent, for v's parent has minimum height at most $k+1$); also, v keeps a charge of $\frac{15}{2}a_1 + b_1$ locally. A node at minimum height 1 receives, at most, a transfered charge of $3a_1$, a charge of $\frac{7}{2}a_1$ from its parent, and a direct charge of $a_1 + b_1$; this is a total charge of at most $\frac{15}{2}a_1 + b_1$. So we see that each node is charged at most $\frac{15}{2}a_1 + b_1$. Finally, we note that the charges are transfered only among nodes in the trees ISR.

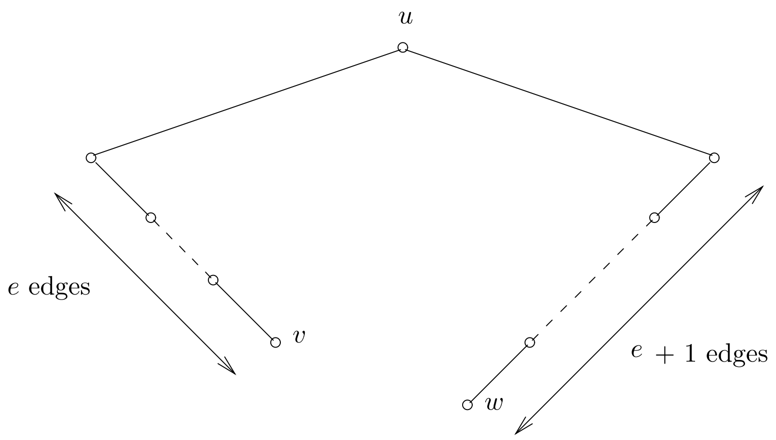

FIG. 3.2. *The node w receiving potential*

So, as the lazy tree is being constructed, it suffices to provide each node in a tree ISR with potential $\frac{15}{2}a_1 + b_1$; this is guaranteed by having

$$q_1 \geq \frac{15}{2}a_1 + b_1 \tag{3.9}$$

3.3.2. The Paths RP. Consider a path, RP. Initially, it comprises a contiguous sequence of nodes, descending to the right. As it is traversed, roughly speaking, it will become a binary tree, at which point the potential for rightist lazy trees becomes adequate. Thus our goal is to provide a potential that progressively converts to a rightist lazy tree potential as the path is repeatedly traversed. As we will see, an initial potential of $a_2 + b_2$ for each node suffices, where a_2 and b_2 are constants to be specified. So it suffices to have

$$q_1 \geq a_2 + b_2 \tag{3.10}$$

3.3.3. The paths $\mathbf{P_1}$ and $\mathbf{P_2}$. They are treated in exactly the same way as the paths RP. To provide sufficient initial potential here it suffices to have

$$a_3\alpha(c_l) + b_3 \geq a_2 + b_2$$

In turn, to satisfy this equation, the following suffices

$$a_3 + b_3 \geq a_2 + b_2 \tag{3.11}$$

In fact, we modify the statement of Section 3.1 that $a_3\alpha(c_l) + b_3$ potential is provided; instead only $a_2 + b_2$ potential is provided, which still suffices. The reason for this modification will become clear in Section 4.

3.3.4. The path $\mathbf{P_3}$. This path cannot be treated in the same way as the paths RP for it does not necessarily comprise a contiguous sequence of nodes in the splay tree.

We maintain the following potential; each node on the current right extreme path of the tree induced by the root of the lazy tree plus P_3, with one class of exceptions, carries a potential of c_2, c_2 a constant to be specified. The exceptional nodes are formed by sequences of $4\alpha(c_l)$ contiguous nodes whose parent and right child in the splay tree are both from P_3; such nodes need not carry the c_2 potential. More precisely, each maximal contiguous sequence of nodes on the right extreme path of the tree induced by P_3, from bottom to top, has an initial segment of length one and a final segment of length at least one and at most $4\alpha(c_l)$ in which each node has a potential of c_2; the remainder of the sequence is partitioned into segments of length exactly $4\alpha(c_l)$, and in each such segment either all or none of the nodes carry a potential of c_2. Each node on the current extreme left path of the tree induced by the root of the lazy tree plus P_3 carries a potential of c_2, with no exceptions.

In addition, P_3 has an associated potential, $Pot(P_3)$; before specifying this potential we need to introduce some other definitions and results. A k-right cascade is a sequence of k single left rotations (a rotation between a node and its right child) on a set of k contiguous couples on a right path. A *right k-cascade sequence* is an intermixed sequence of k-right cascades and arbitrary left rotations. We recall Sundar [Su89] showed that there can be at most $8m$ $2\alpha(m)$-right cascades in a right $2\alpha(m)$-cascade sequence on the normal form of an m-node binary tree. We define $Pot(P_3) = 4c_2\lambda\alpha(c_l)$, where λ denotes the maximum number of $2\alpha(c_l)$-right cascades in a right $2\alpha(c_l)$-cascade sequence that can be performed on the normal form of the tree induced by the nodes of P_3 and the root of the lazy tree, with the root of the lazy tree removed from the normal form (i.e., the cascades never include the root node); this tree comprises at most c_l nodes. The initial values of λ is at most $8|P_3|$, and so $Pot(P_3) \leq 32c_2\alpha(c_l)|P_3|$.

Finally, each node in P_3 carries an additional potential of $4c_2\alpha(c_l) + c_2$. Thus it suffices to provide each node in P_3, on creation, with potential $36c_2\alpha(c_l) + 2c_2$; so it suffices to have

$$a_3\alpha(c_l) + b_3 \geq 36c_2\alpha(c_l) + 2c_2$$

That is, it suffices to have

$$a_3 \geq 36c_2 \tag{3.12}$$

$$b_3 \geq 2c_2 \tag{3.13}$$

3.3.5. BL, the Skeleton Nodes in the Leftmost Block. Each node in the leftmost block of the lazy tree carries an additional potential of c_3 units. Nodes in the other blocks carry an additional potential of $c_3 + c_4$ units. In order to provide this potential initially, we augment each of q_1, $a_3 + b_3$, and b_3 by $c_3 + c_4$ units. So it suffices to have

$$q_1 \geq \max\{\frac{15}{2}a_1 + b_1, a_2 + b_2\} + (c_3 + c_4) \tag{3.14}$$

$$a_3 + b_3 \geq a_2 + b_2 + (c_3 + c_4) \tag{3.15}$$

$$b_3 \geq 2c_2 + (c_3 + c_4) \tag{3.16}$$

3.3.6. Debits on the Local Nodes of the Lazy Tree. Nodes on the extreme paths of the lazy tree may have a small or large debit; these debits satisfy Invariants 3–6, where lazy tree L replaces block B in the statement of the invariants. Also, each node in the lazy tree can be considered to be labeled. A labeling potential is not needed here, however, for the removal of nodes from an external path of the lazy tree is always paid for in some other way, as we shall see. However, so that that we can apply Invariants 3–6, we will consider the nodes of a lazy tree to be labeled. In addition, each light 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. Lazy debits satisfy the following invariants.

INVARIANT 11. *Let L be a right (resp. left) lazy tree. Suppose node v in L has a lazy debit. Then v is a light node of L. Also, v is not on the left (resp. right) extreme path of L. Finally, v does not carry a small or large debit.*

INVARIANT 12. *Let L be a right (resp. left) lazy tree. Let u be a light 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 (resp. right) extreme path of L. Then if u is on the right (resp. left) 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 and are in the same component as u.

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

INVARIANT 13. *BL and P_3 do not carry lazy debits.*

In addition, each node in the lazy tree may have a *border* debit, which is enormous and has value ed; ed is a constant to be defined later. Border debits satisfy the following invariants.

INVARIANT 14. *Let L be a right (resp. left) lazy tree. Suppose node v in L has a border debit. Then v is not the root of L and is not on the left (resp. right) extreme path of L. Further, if v is on the right (resp. left) extreme path of L then both v's parent and right (resp. left) child in the splay tree are in the same component of L.*

INVARIANT 15. *A node has at most one debit.*

Next, we show how to incorporate lazy trees into the analysis of accesses. An access 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 block of the right guard to be a split (a type (c) traversal); likewise a traversal of type (b) to the right of the block of the left guard is defined to be a split.

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, as a block. We note that Invariant 12 ensures that a node on an extreme path, carrying a lazy debit, satisfies condition (i) of Remark 1. In addition, we note that on creation, the nodes of the lazy tree have no debits; so Invariants 3–6 all hold at this point.

Next, we need to show that condition (ii) of Remark 1 is satisfied. That is we have to show that couples involving two nodes on an extreme path of the lazy tree, which will be marked, can pay for the removal of their debits and $s+1$ rotations. This is done in Section 3.4.

Segments are defined and paid for essentially as before. The one difference occurs if either node of the leading couple of the segment carries a lazy debit; then the couple pays for its own rotation and the removal of its debits as part of the lazy tree, as we see later; but this can only reduce the total remaining cost of the segment and so the bounds of the previous analysis are still valid.

We need to mention one detail about couples containing the root, u, of a lazy tree and another node, v in the same lazy tree. Following the rotation, v becomes the root of the lazy tree; v and u interchange roles and potentials (so if v had been light, resp. heavy, u becomes light, resp. heavy).

In Section 3.5 we explain how a lazy tree is split.

3.4. Extreme Path Traversal. We consider all rotations involving two nodes on an extreme path. So consider a couple comprising nodes u and v, where u is the parent of v. We distinguish the following types of couples.

CASE 1. u and v are both nodes of P_3. If both u and v carry c_2 potentials then the potential associated with u, which ceases to be on an extreme path of P_3, is used to pay for the rotation, s spares and the removal of any debits, which do not include lazy debits (see Invariant 13). So it suffices to have

$$c_2 \geq (s+1) + \max\{2ed, ld\} \tag{3.17}$$

If one or both of u and v do not carry a c_2 potential then they are part of a contiguous sequence of $4\alpha(c_l)$ nodes lacking this potential. This sequence undergoes a $2\alpha(c_l)$-right cascade (or perhaps this sequence of nodes shifted by one). The cascade provides c_2 potential to each of these $4\alpha(c_l)$ nodes; we can then handle the rotation between u and v as before. The $4c_2\alpha(c_l)$ cost of the cascade is charged to P_3 (recall that P_3 had been given sufficient potential to pay for the maximum possible number of such cascades).

CASE 2. u and v are both nodes of BL. The c_3 potential associated with u, which ceases to be on BL, is used to pay for the rotation, s spares and the removal of any debits, which do not include lazy debits (see Invariant 13). So it suffices to have

$$c_3 \geq (s+1) + \max\{2ed, ld\} \tag{3.18}$$

CASE 3. u and v are in distinct components. Then u is given a border debit. This pays for the rotation, s spares, and the removal of debits from u and v (note that by Invariants 14 and 12 u and v carry neither lazy nor border debits). So it suffices to have

$$ed \geq (s+1) + ld \tag{3.19}$$

CASE 4. u and v are in the same tree SR, a *rightist lazy tree*. Again, without loss of generality, we are only considering right lazy trees. The analysis is similar to that in splay sorting $\log n$-block sequences but the possible presence of border debits changes the cost of the various rotations.

CASE 4.1. u and v are both light nodes. u ceases to be on the skeleton. u's c_1 potential pays for the rotation, s spares, and the removal of debits on u and v. So it suffices to have

$$c_1 \geq s + 1 + \max\{2hd, 2ed, ld\} = s + 1 + \max\{2hd, 2ed\} \tag{3.20}$$

CASE 4.2. u is a light node and v is a heavy node. By Invariant 12, u does not have a lazy debit. If u leaves the skeleton, the operation is paid for by u's c_1 potential, as in Case 4.1; Equation 3.20 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 the small, large or border debits, if any, from u and v. So it suffices to have

$$hd \geq s + 1 + \max\{ld, 2ed\} = s + 1 + 2ed \tag{3.21}$$

CASE 4.3. v is a light node, u is the root of v's subblock but is not the lazy tree root. By Invariants 11 and 12, u and v do not have lazy debits. v becomes the block root. As in Case 4.2, if u leaves the skeleton, the operation is paid for by u's c_1 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 the path or border debits, if any, from u and v. Here too Equations 3.20 and 3.21 suffice.

CASE 4.4. u and v are both heavy nodes. We need to pay for the rotation, for s spares, and for the removal of path or border debits from u and v, if any. In addition, we may need to reestablish Invariant 12 for the nodes on the path newly abutting u; this may require the removal of up to two lazy debits, which will also be paid for by the rotation. So the cost of this rotation is bounded by

$$s + 1 + 2hd + 2ed \tag{3.22}$$

The analysis of this case is identical to that of the corresponding case in the splay sorting of $\log n$ block sequences [CMSS98]; it is omitted in part. However, in order to understand the analysis of Case 5, it is helpful to repeat it in part.

For the remainder of this case, until we indicate otherwise, when we refer to a tree we intend the component lazy block tree. So, for instance, when we refer to a node we mean a node in the component lazy block tree.

We start with some definitions. The *depth* of a node is its distance from the root. The height of a node at the time of the creation of the lazy tree is called its *creation-height* (a leaf has creation-height 1); the creation-height does not change subsequently.

Consider how an extreme path traversal appears to the tree. A top contiguous portion of the nodes on this path are all *touched* (these are the nodes contained in couples of the splay tree). Some (arbitrary) subset of disjoint pairs of touched nodes form couples. Each touched node is provided with $\frac{s'}{2}$ spare units of potential (this is the node's share of the portion of s' spares provided to its couple in the splay tree). The spare potential of the touched nodes, together with any changes to the potentials of the touched couples will pay for the rotations of the touched couples.

At any time, certain nodes, called *active* nodes, are the nodes that pay for a traversal of an extreme path. Some nodes may pay more than other nodes. This is captured by the notion of *active layers*. A node of creation-height h has an associated *span* of layers $[1, h]$; each active node v of creation-height h has an *active span* of *active layers*, $(g, h]$, $0 \leq g \leq h$; for each k, $g \leq k \leq h$, we say v is k-*active*. If the active span is non-empty we say the node is *active*.

Initially, only the nodes on the 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.

A rotation of the splay operation involving a heavy node, v, and the root, r, of the lazy tree causes the two nodes to swap all their potentials. Also r acquires v's creation-height and active span.

The following invariant states several properties of the active nodes. We prove the invariant later. In order to avoid special cases for the left path we state our invariants with respect to the normal form of the tree, where the tree comprises the root of the lazy tree together with the component lazy block tree.

INVARIANT 16. *Let H be the maximum creation-height for the nodes, other than the root, present in the tree initially. Then, in the corresponding normal tree:*

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

(ii) Every node on the right path is active (this does not include the root).

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

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

(v) Let inactive node v have creation-height k. Then its parent has creation-height greater than k.

When a lazy tree is created the active spans for the nodes in the corresponding component lazy block tree are initialized as follows. Let u be a node on the right path, of creation-height h; suppose it has a right child v of creation-height g (if there is no such node v let $g = 0$). Then u is given active span $[g + 1, h]$. Clearly, the new tree obeys Invariant 16.

The layers of a node are further categorized as *black* or *white*; an active node, with active span $[g, h]$, can be black with respect to each of the layers $[1, g - 1]$. In general, an active node v, with active span $[g, h]$, is black with respect to all the layers in some range $[f, g - 1]$, $f \geq 1$, called its *black span*; we say v is k-*black*, for $f \leq k \leq g - 1$. 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 17.

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

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

We define the following distances for each node v, active at layer k. Its *right path k-distance*, $d_k(v)$, is the number of proper left ancestors of v on the right path, excluding all k-black nodes. Its *interior right path k-distance*, $id_k(v)$, is the number of proper left ancestors of v below the first k-black node and below the right path.

For each k-active node we maintain a k-potential, which satisfies the following invariant (d_1 is a constant to be specified).

INVARIANT 18. *The k-potential at k-active node v is at least*

$$a_1 \cdot min\{d_k(v)/d_1, k\} + a_1 id_k(v)/d_1$$

We note that initially a node has $a_1 \cdot k$ units of potential for each layer k at which it could become active, so when a node first becomes k-active Invariant 18 holds.

For each k-black node we maintain a k-black potential of $\frac{a_1}{2d_1}$ units.

We conclude by stating the sufficient conditions that are imposed by the analysis of [CMSS98] (in the previous analysis a and d replace the present a_1 and d_1, respectively).

$$d_1 s'/2 \geq \max\{s + 1 + 2hd + 2ed, a_1/d_1\} \tag{3.23}$$

$$\frac{1}{2} a_1/d_1 \geq s + 1 + 2hd + 2ed \tag{3.24}$$

There is one matter related to this analysis which needs detailing. Immediately prior to a traversal of the right path, each node on that path is given $s'/2$ spare rotations (whether or not the node is part of a couple comprising two heavy nodes of the same lazy tree component SR). Each node's spares are subsequently provided by the rotation which involves that node. The $s'/2$ spares associated with each node are redistributed in order to pay for those rotations which are not self paying. The details can be found in [CMSS98].

Case 5. u and v are both in a component RP (or are both in P_1 or P_2).

We begin by describing the potential associated with the nodes of component RP.

The nodes originally forming the path RP are partitioned into three sets:

(i) Those nodes on a shortened path, RP', called *pure path nodes*.

(ii) *Rare black nodes*.

(iii) *Rightist nodes*.

The following invariants apply to the path RP.

Invariant 19. *The nodes in RP' form a contiguous path in the splay tree.*

Invariant 20. *After t traversals, a node on RP' has potential $a_2 \sum_{j=1}^{t+1} j + b_2$, $a_2 \geq a_1$ and $b_2 \geq b_1$ being constants to be specified.*

Invariant 21. *Let N be the normal form of the lazy tree L. Let v be a node in RP, a part of L.*

(i) Suppose that v is a rare black node. Each of v's left ancestors in N which is also in RP is also a rare black node.

(ii) Suppose that v is in RP'. Then each of v's left ancestors in N which is also in RP is either in RP' or is a rare black node.

Invariant 22. $|RP'| \neq 1$.

Each rightist node has a creation-height and corresponding potentials satisfying Invariants 16–18. Each rare black node also has a creation-height; these black nodes have the *rareness property*, namely there are no other nodes in this component RP with the same creation-height. The rare black nodes carry k-black potentials and k-potentials; they too satisfy Invariants 16–18. Owing to the rareness property we are able to reduce their initial values somewhat. However, it is easier to defer the precise specification of these potentials till Cases 5.2 and 5.3, where we explain how the potentials change as the rotations analyzed in these cases are performed. The nodes in RP' acquire creation-heights and appropriate rightist lazy tree potentials as and when they leave RP', either to become rare black nodes or rightist nodes.

Initially, $RP = RP'$. So an initial potential of $a_2 + b_2$ for each node in RP suffices, as claimed in Equation 3.10.

CASE 5.1. Neither u nor v is in RP'. This is treated as a rotation in a rightist lazy tree. It is handled exactly as in Case 4.

CASE 5.2. u and v are pure path nodes (i.e., both are on RP'). We require that the cost of the rotation be at most $s + 1 + ld$. This follows by ensuring that the pure path nodes carry no border or lazy debits. So we add the following invariant.

INVARIANT 23. *Let u be a pure path node in a lazy tree. Then u does not carry a border or lazy debit.* We note that Invariant 23 can be maintained by trimming the ends from RP' whenever an end is part of a couple including a node outside RP'. So if, in couple (u, v), with u the parent of v, v is the top node on RP', v is converted to a rare black node (by the method of Case 5.3, below); while if u is the bottom node on RP', u is converted to a rightist node (by the method of Case 5.4, below).

Suppose that this is the $(t + 1)$th traversal of u and v. Then their potentials are adjusted as follows. u, which leaves RP', receives potential $a_1 \sum_{j=1}^{t+1} j + b_1$ and becomes a rightist node with creation-height $t + 1$ (at this point we further reduce u's potential if it is light — see Section 3.3.1). v remains on RP' and so its potential increases to $a_2 \sum_{j=1}^{t+2} j + b_2$ (see Invariant 20); the rotation itself costs at most $s+1+ld$ units. So it suffices to have:

$$a_1 \sum_{j=1}^{t+1} j + b_1 + a_2 \sum_{j=1}^{t+2} j + b_2 + (s + 1 + ld) \leq 2a_2 \sum_{j=1}^{t+1} j + 2b_2 \tag{3.25}$$

This equation is satisfied by setting

$$a_2 \geq 2a_1 \tag{3.26}$$

$$b_2 \geq b_1 + (s + 1 + ld) + \frac{3}{2}a_2 \tag{3.27}$$

If v is now the only node on RP' it is converted to a rightist node, with creation-height $t + 2$, by reducing its potential to $a_1 \sum_{j=1}^{t+2} j + b_1$ (note Invariant 22).

Note that the creation-heights for rightist nodes obey Invariant 16(v).

CASE 5.3. v is on RP', but u is not on RP'. We provide v with appropriate j-potential and j-black potential; v becomes a node of the complete rightist lazy tree to be formed from the initial path RP. In particular, v becomes a rare black node (which behaves in the same way as a black node in a rightist lazy tree). The rotation between u and v is paid for as in a rightist lazy tree rotation (see Case 4, above).

It remains to explain how to provide v with sufficient potential. Suppose the path, RP', immediately before the traversal has length $2^l \leq |RP'| < 2^{l+1}$; further suppose that the nodes on RP' have been traversed t times so far. Then v receives creation-height $h(v) = 2l + t + 1$. v requires $\frac{a_1}{2d_1}(h(v) - 1) = \frac{a_1}{2d_1}(2l + t)$ units of potential for its k-black potentials; it requires a constant potential of b_1 units; finally, it requires $a_1 \sum_{j=1}^{2l+t} \min\{t, j\}$ units of j-potential. (At first sight, one would expect a quantity $\frac{1}{2}a_1 h(v)(h(v) + 1)$. But, in fact, we can reduce this total for two reasons.

(i) The $h(v)$-potential can be set to zero for node v, since $ld_{h(v)}(v) = 0$.)

(ii) Only $\min\{t, j\}$ units of j-potential are required for the other values of j, since v can have at most t left ancestors in its component in the normal form of the lazy tree when and if it becomes j-active.

Note that t increases by 1 each traversal, while l decreases by at least 1 each traversal. So the creation-heights of the rare black nodes ordered by time of creation are strictly decreasing; also, they are larger than the creation-heights of all the non-rare rightist nodes that are eventually created — the non-rare rightist node with largest creation-height is the node which is eventually alone on the path RP'; it has creation-height at most $l+t+1$.) As the rare black nodes are also ordered left to right by their creation time, as well as being to the left of all non-rare nodes, Invariant 16(v) is satisfied by the rare black nodes too.

Assuming that

$$d_1 \geq 1 \tag{3.28}$$

the total potential needed for v is at most $a_1(l+t/2)+\frac{1}{2}a_1t(t+1)+2a_1l\cdot t+b_1$, which is $a_1[l(2t+1)+\frac{1}{2}(t^2+2t)]+b_1$. This potential is provided by the potential already at hand for node v plus a charge of $a_1(2t+1)$ to each node that leaves the path RP'. v already has a potential of $\frac{1}{2}a_2(t+1)(t+2)+b_2$ at hand; this covers $\frac{1}{2}a_1(t^2+2t)+b_1+a_1(2t+1)$ units of v's future potential (since $a_2 \geq 2a_1$ and $b_2 \geq b_1$); For $l \geq 1$, $2^{l-1} \geq l$; as there are at least $2^{l-1}-1$ nodes leaving RP', the charge to the leaving nodes covers the remaining $a_1(l-1)(2t+1)$ units of v's future potential. To cover this charge to the leaving nodes it suffices to modify Equation 3.25 as follows:

$$a_1\sum_{j=1}^{t+1} j+b_1+a_2\sum_{j=1}^{t+2} j+a_1(2t+1)+b_2+(s+1+ld) \leq 2a_2\sum_{j=1}^{t+1} j+2b_2$$

This equation is satisfied by setting

$$a_2 \geq 2a_1 \tag{3.29}$$

$$b_2 \geq b_1+(s+1+ld)+7a_1 \tag{3.30}$$

If v is a light node its k-potentials and k-black potentials are replaced by a potential of c_1; Equation 3.8 guarantees that there is sufficient potential at hand.

CASE 5.4. u is on the path RP', but v is not. Then u is made into a rightist node with creation-height $t+1$; the potential presently associated with u is larger than the potential it subsequently requires. If u is a light node its potential is reduced to c_1; Equation 3.8 guarantees that there is sufficient potential at hand. The rotation is then treated as a rotation in a rightist lazy tree; this is handled by Case 4.

It order to justify the claim made in Remark 2 we need to report the amortized cost of the rotations occurring in the traversal of a lazy tree extreme path.

LEMMA 3.1. *The cost of each rotation of a couple comprising two nodes on the extreme path of a lazy tree is at most* $\max\{hd, ed\}$.

Proof. In Section 3.4 we have considered all couples comprising two nodes on an extreme path. Cases 1, 2, 4.1, 4.4, 5.2–5.4 are all self-paying (that is, they do not require the creation of debits). Case 5.1 is subsumed by Case 4. Case 3 has cost at most ed and Cases 4.2 and 4.3 have cost at most hd. ◻

3.4.1. Rotations with the Lazy Tree Root. In a rotation with the lazy tree root, the root and the other node in the couple interchange lazy tree potentials. Debits are removed and created as for normal rotations on the inner skeleton of a block.

3.5. Splitting the Lazy Tree. A split of the lazy tree occurs when the accessed item lies strictly between the subblocks of the left guard and the right guard of the lazy tree. Thus a split must access a new i-block. So a split is the first access of a new sequence S_l. For this section, we assume that the open intervals spanned by the guards of each lazy tree are disjoint. In Section 3.6, we analyze the general case. Without loss of generality, we suppose the lazy tree being split is a right lazy tree.

At this point it is helpful to mention a few properties of the lazy and reserve potentials. Consider the global nodes in a new lazy tree. Let SL be the set of nodes contained strictly between the blocks of the guards of a lazy tree. For each heavy node, u, in the lazy tree, define its right neighbor $r\_n(u)$ to be the heavy 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. Finally, define the lazy rank of heavy node v to be $\frac{1}{gp}$ times its lazy potential.

Invariant 24. *Let v be a heavy node in the normal form of the lazy block tree for lazy tree L. Then:*

(i) $lazy\_rank(v) \geq \lfloor \log(wt(SL(v))) \rfloor$.

(ii) If, in the normal form of the large lazy block tree, v has no heavy node or guard in its right subtree, $lazy\_rank(v) + 1/gp \cdot reserve(v) \geq g\_rank(v)$. While if v does have a heavy node or guard in its right subtree, then $lazy\_rank(v) + 1/gp \cdot reserve(v) \geq lazy\_rank(r\_n(v))$.

Invariant 24 can be seen to hold when the lazy tree is created by considering node v at the point at which it becomes lazy. Also, is is readily seen that this invariant continues to hold following traversals of the extreme paths (for they leave the lazy rank of nodes in the normal form of the lazy tree unchanged).

Corollary 3.2. *Let v be a heavy node other than the root in lazy tree L. Let w be v's right child and u be v's left child.*

(i) *Suppose that v is on the left path of L; then $lazy\_rank(v) \geq g\_rank(w)$.*

(ii) *Suppose that v is not on the left path of L. Let v' be a right descendant of v, not necessarily proper. If v' is a heavy node, $lazy\_rank(v') \geq g\_rank(u)$.*

.

It is convenient to define v's *lazy weight* to be $wt(SL(v))$.

At this point, we appeal to the construction in [CMSS98] to show that the split can be performed. To do this we review Invariant 11, Properties 3 and 4, and Lemma 4.7 of that paper.

Invariant 25. *(Invariant 11 of [CMSS98].) Consider a right lazy tree containing node v. Suppose v is a light node and carries an additional debit (this is referring implicitly to a border debit). Then*

(i) v is not on the left path of the (right) lazy tree.

(ii) If v is on the right path of the (right) lazy tree then so are its parent and right child in the splay tree.

An analogous invariant applies to left lazy trees.

Property 1. *(Property 3 of [CMSS98].) The restoration of any further invariants concerning debits (this refers implicitly to Invariants 12 and 14) requires the removal of at most α additional debits from each lazy tree created by the split, α a constant.*

Lemma 3.3. *(Lemma 4.7 of [CMSS98].) If a lazy tree comprises light and heavy nodes where the heavy nodes carry lazy and reserve potentials satisfying Invariant 24, then each lazy tree resulting from the split described in [CMSS98] satisfies:*

(i) If the nodes on its left path are new to a left path, then they are all light nodes in the same block.

(ii) Assume the debits obey Invariants 3–6 and 13, and the invariants implicit in Property 1, namely Invariants 12 and 14. If the following equations hold,

then these Invariants can be restored following a split.

$$c' \geq \max\{4md, ld + 3md\} = 4md \tag{3.31}$$

$$b \geq 3md + 2ld + 2sd + gp + \alpha \cdot md \tag{3.32}$$

$$\textit{where } md = \max\{ed, hd\} \tag{3.33}$$

Note that $b = c' = \min\{c_3, c_4\}$; they are the additional potential associated with heavy and light nodes, respectively, to be used to cover the costs of removing these nodes from the lazy tree.

PROPERTY 2. *(Property 4 of [CMSS98].) The potential must be partitionable following a split; i.e., each (component of each) lazy tree created by the split must be provided with an appropriate potential (eg. a lazy complete tree potential).*

Clearly Invariant 25 is true of the border debits (see Invariant 14). We use the splitting method from [CMSS98]. To justify its use we need to bound α and prove Property 2. Both results rely on on Lemma 3.3(i). Indeed, Property 2 is already known for the lazy complete tree potential. It remains to consider the other types of potential and to determine a bound on α.

The Path **RP**. The nodes not on path RP' are treated as ordinary rightist lazy tree nodes. The nodes on RP', are separated into two paths, RP'_1, RP'_2, by the split (one of these paths may be empty). If either path RP'_i is of length 1 then it is converted to a rightist node (that is, its potential is reduced from $a_2 \sum_{j=1}^{i+1} j + b_2$ to $a_1 \sum_{j=1}^{i+1} j + b_1$ for some i). We note that this restores Invariant 22. There is no other change to the potentials of the nodes on the paths RP'_i. It is clear Invariants 19–21 and 23 continue to hold.

The paths P_1 and P_2 are treated in the same way.

The Path $\mathbf{P_3}$. Following the partitioning we need to provide c_2 potential to each node which is newly on a right extreme path (unless it is part of a contiguous sequence of nodes of length at least $4\alpha(c_l)$). We note that there are no nodes newly on a left extreme path and in a P_3 component (for such nodes become part of a BL component). Each node, v, which had been in component P_3, and which has been promoted or is now in an extreme left subblock of a new lazy tree or is now in a normal subblock (a subblock that is no longer part of a lazy tree), may spend up to $c_2 \cdot (4\alpha(c_l) + 1)$ potential is order to provide c_2 potential for each of the up to $4\alpha(c_l) + 1$ following nodes: those nodes in the contiguous portion of the new right extreme path, if any, immediately below and to the left of v in the splay tree, selected so as to restore the proper distribution of c_2 potentials (see Section 3.3.4); this is a portion of a right extreme path in component P_3 for a neighboring new lazy tree (the new component P_3 is a portion of the old component P_3). As node v is no longer in a P_3 component it can afford to spend potential $4c_2\alpha(c_l) + c_2$.

Also, when the tree formed from P_3 is split, the potential associated with P_3 is partitioned, in proportion to how many $2\alpha(c_l)$-right cascades can be performed on each tree formed from P_3 (by Sundar's bound [Su89], there is sufficient potential at hand). To see this, consider the tree induced by the original component P_3 plus the root of the splay tree in its normal form. We simulate the actual rotations as follows. Only traversals of extreme paths of current P_3 components are simulated, and then only on couples comprising two nodes of P_3. The effect is to maintain each current P_3 component in its normal form, so all extreme path traversals are simulated as right cascade sequences. So the effect of a split is merely to reduce the possible sequences of right cascades; hence Sundar's bound applies, or in other words, the potential associated with P_3 suffices to provide each of its partitions with their associated potential.

Bounding α. The additional constraints on the debits are given in Invariants 12 and 14, in the requirement that neighbors of a node with a lazy or border debit be in the same component and not simply in the lazy tree.

For each lazy tree created by the split, for each component, we remove the lazy or border debits, if any, from the leading and tail nodes of the components on the extreme paths. Clearly, there are at most twelve such lazy debits for each new lazy tree. So we can choose $\alpha = 12$.

By Lemma 4.4 of [CMSS98] the cost of the promotions is bounded as follows.

LEMMA 3.4. *(Lemma 4.4 of [CMSS98].) The promotions in a split have the following cost: $2gp$ times the increase in global rank along the right split path plus $2gp$ times the increase in global rank along the left split path.*

COROLLARY 3.5. *Consider a single access A and the consequential promotions of nodes in lazy trees on the skeleton of superblock B. Let b be the number of blocks in superblock B. The promotions of access A cost at most $4gp(\log 2b + 2)$ units, there being a charge of $2gp(\log 2b + 1)$ to the right split path and of $2gp(\log 2b + 1)$ to the left split path.*

3.6. Multiple Level Lazy Trees. Because the access paths for a sequence, S_i, of accesses to an i-block 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. The generalization is exactly as in [CMSS98]; the analysis proceeds exactly as described there. Lemma 3.4 continues to hold. The only detail is that Equation 3.31 needs to be replaced by (see Equation 25 of [CMSS98]):

$$c' \geq 4md + \alpha \cdot md = 16md \tag{3.34}$$

Recall that $c' = \min\{c_3, c_4\}$ and $\alpha = 12$.

3.7. The Full Result. We pull together the various equations from the previous subsections in order to bound the cost of an access. First, we define an access to be of type i if the previous access is to a distinct i-block but to the same $(i+1)$-block. We show a bound of the form $f_1 \log b_{i+1} + f_2(\log\log n)^2 + f_3 \log\log n\alpha(n) + f_4$ on the amortized cost of a type i access, where f_i, $i = 1, 2, 3, 4$ are constants. (Recall that we are choosing $b_i = 2^{2^i}$.) Then we show how to replace the $\log b_{i+1}$ term by a $4 \log d$ term, where d is the distance between the currently accessed item and the previously accessed item.

To avoid tedious computations, we will assume that $n \geq 16$.

Recall the statement of Lemma 2.14; we chose $sd = s+1$, $ld, c = 17(s+1)$, $gp = 19(s+1)$, and $vp = 30(s+1)$. We need to satisfy the constraints present in Equations 3.8–3.34. By choosing equalities in some of these constraints we satisfy them all, as we demonstrate. Equalities in Equations 3.19, 3.21 and 3.33 yield $ed = 18(s+1)$, $md = hd = 37(s+1)$. Equality in Equation 3.20 yields $c_1 = 75(s+1)$, and equality in Equation 3.32 yields $c_3 = c_4 = 610(s+1)$ (this subsumes Equations 3.18, 3.31 and 3.34. Equality in Equation 3.17 yields $c_2 = 37(s+1)$. Now, we choose $s = 1$ and hence $s' = \frac{1}{3}$. With equalities in Equations 3.23 and 3.24 we obtain $d_1 = 1332$ (which subsumes Equation 3.28), and $a_1 = 1332 \times 222(s+1) = 295,704(s+1)$ (this subsumes Equation 3.8). Next, we choose $b_1 = 0$. With equalities in the following equations we obtain: Equation 3.29 (which is identical to Equation 3.26) $a_2 = 591,408(s+1)$; Equation

3.30, $b_2 = 2,069,946(s+1)$ (this subsumes Equations 3.25 and 3.27); Equation 3.15, $a_3 + b_3 = 2,662,574(s+1)$, and on choosing $a_3 = b_3$, this subsumes Equations 3.11–3.13 and 3.16; Equation 3.14, $q_1 = 2,662,574(s+1)$ (this subsumes Equations 3.9 and 3.10). Finally, from Equation 3.2, 3.6 and 3.7, we deduce that $e' = 24$ suffices.

Substituting these values into Equations 3.3, 3.4, and 3.5 we obtain that the amortized cost for the portion of the traversal in which the accessed item has visibility i for the first access to an i-block is bounded by $4,000\log 2b_{i+1} + 4,200$, and the amortized cost for a subsequent access is bounded by $[6,000,000+3,000,000\alpha(c_{i+1})]$. Summing over all levels, we obtain a bound for the first access in S_i of $8,000\log b_{i+1} + \lceil \log\log n\rceil[6,000,000+3,000,000\alpha(n)]$ (for recall that $b_i = 2^{2^i}$).

Now, we show how to avoid the dependence on block boundaries, which so far has meant that a search a short distance from the previous access may be treated as if it were far away (if a boundary between two high level blocks is crossed). The idea is to average over many possible positions for the block boundaries so that if two nodes, u, v, are distance d apart (counting the number of items spanned by $(u,v]$ or $(v,u]$, according as $v > u$ or $v < u$) then, on average, the two items are contained in a block of size less than d^4.

More precisely, imagine an ordered set of $2n-1$ items, with the n items of the tree occupying n contiguous positions in the set. We define block boundaries on the set of $2n-1$ items, and consider the n positions that can be occupied by the items in the tree. For each such position, the block boundaries partition the tree of n items; the corresponding potential is associated with the tree. The overall potential for the tree is simply the average of these associated potentials.

Now it is straightforward to bound the cost of a search. Suppose the current access is distance $d \geq 16$ from the previous access. Suppose $2^{2^{k+1}} \leq d < 2^{2^{k+2}}$. As we have seen, the first access to a j-block, which we call a search of type j, costs $f_1 2^{j+1} + f_5\lceil\log\log(2n-1)\rceil$, where $f_1 = 8,000$ and $f_5 = [6,000,000+3,000,000\alpha(2n-1)]$. For the n choices of block boundary at hand, let n_j denote the number of boundaries for which the current access is a search of type j. Then $n_{k+h} = d \cdot n/2^{2^{k+h+1}}$. The cost of the search is therefore

$$\leq \frac{1}{n}\sum_{h\geq 1} f_1 2^{k+h+1} n_{k+h} + f_1 2^{k+1} + f_5\lceil\log\log(2n-1)\rceil$$

$$\leq f_5\lceil\log\log(2n-1)\rceil + f_1 \sum_{h\geq 1} 2^{k+h+1}\cdot d/2^{2^{k+h+1}} + f_1\cdot\log d$$

$$\leq f_5\lceil\log\log(2n-1)\rceil + f_1 2^{k+1}\sum_{h\geq 1} 2^h \cdot 2^{2^{k+2}}/2^{2^{k+h+1}} + f_1\cdot\log d$$

$$\leq f_5\lceil\log\log(2n-1)\rceil + f_1\log d\cdot 4 = 4f_1\log d + f_5\lceil\log\log(2n-1)\rceil$$

For $d < 16$, the $\log d$ term can be absorbed into the $\log\log$ term.

So we have shown:

THEOREM 3.6. *Consider a sequence of searches performed on an n-node splay tree, $n \geq 16$. Let item v be the current item being accessed and u the previous item accessed. Suppose that the distance from u to v is d. Then the amortized cost of* *the access, in rotations, is bounded by* $32,000 \log d + \lceil \log\log(2n-1) \rceil [6,000,000 + 3,000,000\alpha(2n-1)]$.

Finally, we need to consider the initialization costs. For each i-block, we need to provide its root with its global potential, unless it is the root of an i-superblock. Per i-block root, this costs at most $gp(2^{i+1}+1)$; there are at most $n/2^{2^{i+1}}+2$ such nodes, allowing for up to two undersize nodes. In addition, we need to provide the path and reserve path potentials, whose cost is bounded by $144c \cdot n$ as shown in Lemma 2.13 (the proof requires slight modification to account for up to two undersize blocks per level, but the result is unchanged), and the zig-zag potentials, with cost bounded by cn.

Summing over all levels i, we obtain that the initialization costs are bounded by:

$$144c \cdot n + gp \sum_{i=1}^{vis} (2 + n/2^{2^{i+1}}) \cdot (2^{i+1}+1) + cn$$

$$\leq 4930 + 3gp \cdot n \sum_{i \geq 1} (2^{i+1}+1)/2^{2^{i+1}} + gp(2^{vis+3} + 2vis)$$

$$\leq 4930 + 3gp \cdot n + 9gp \cdot n$$

$$\leq 6000n$$

We conclude:

Theorem 3.7. *Consider a sequence of searches performed on an n-node splay tree, $n \geq 16$. Let item v be the current item being accessed and u the previous item accessed. Suppose that the distance from u to v is d. (For the first search, $d = n$.) Then the amortized cost of the access, in rotations, is bounded by* $32,000 \log d + \lceil \log\log(2n-1) \rceil [6,000,000 + 3,000,000\alpha(2n-1)]$*, where the cost of initializing the potential is at most* $6000n$.

4. Paying for the Level Jumps. In this section we show how to remove the additive term of $O([\log\log n]\alpha(n))$ for each access. This charge is associated with the increases in visibility on the part of the accessed item. Recall that an item on the inner skeleton of an i-superblock has visibility i and an item on an external path of the vis-block has visibility vis. For each increase in the visibility of the accessed item there are three distinct charges. We detail these charges for the rotation that causes the accessed item to attain visibility $i+1$, for $i \geq 1$.

(i) The charge, vp (see Section 2).

(ii) The charge, $a_3\alpha(2^{2^{i+2}}) + b_3$, for making the topmost node of the just traversed right access path lazy (see Section 3.1). Recall that an i-superblock contains c' items; now we are setting $c' = \min\{n, 2^{2^{i+2}}\}$.

(iii) The charge of c for each h-block, $h \leq i+1$, which receives a new root; this is a charge of at most $2c$ for each unit increase in visibility (see Lemma 2.14).

These costs arise at most once at each increase in the visibility of the accessed item. They are charged to the new visibility of the accessed item. The total charge to visibility level $i+1$ is bounded by $cost_{i+1} = vp + a_3\alpha(2^{2^{i+2}}) + b_3 + 2c$.

The cost of the rotation in which the accessed item attains visibility 1 is charged to the access itself; this is a charge of at most $vp + 2c$ (for no lazy trees are created within a 1-block as accesses to the same 0-block are accesses to the same item, which therefore is to be found at the root of the splay tree).

The analysis focuses on an access to the right of the splay tree root; an access to the left of the root is treated entirely analogously.

A few definitions will be helpful. Let B be the i-superblock being accessed currently. Suppose the current access is to the same i-block as the preceding access. The right i-header, i_h_r, is defined to be the topmost node on the right access path for B (see Section 3.1 for the definition of the right access path). The *right i-leader* is defined to be the lowest ancestor of the right i-header which is on the right extreme path of the splay tree. We say the right i-header is *available* if it is on the left path (the *left i-path*) descending from the right i-leader. The number of proper right ancestors of i_h_r, called its *right depth*, is denoted i_d_r.

i_h_r has potential $c_5(cost_{i+1} \cdot \min\{i\_d_r, d_5 i \cdot cost_{i+1}\})$, where c_5 and d_5 are constants to be specified.

If the right i-header is not traversed there are no charges to visibility level $i+1$.

Consider an access which traverses the right i-header. The intuition is that the right i-header reduces its right depth and this either reduces its potential, or if it is too deep i-spares (to be defined) will be available; in either case, these pay for the charge $cost_{i+1}$. Specifically, if the right i-header is available, but is neither the topmost nor the second highest node on the left i-path, then on traversal its right depth decreases. If it is at right depth at most $d_5 i \cdot cost_{i+1}$ then the decrease in its potential provides the required $cost_{i+1}$ charge. If it is at greater depth, then $s'/2$ of the spares associated with the nodes on the left i-path at right depths $(d_5(i-1) \cdot cost_{i+1}, d_5 i \cdot cost_{i+1}]$ cover the $cost_{i+1}$ charge (note that these nodes are traversed). While the accesses are to the same i-block, whether or not the right i-header is traversed, its right depth does not increase.

Next, we handle the case that the right i-header is at right depth 0 or 1. But here charge (ii) is reduced: it becomes $a_2 + b_2$ (see Section 3.3.3). This is charged to the access operation itself, as is charge (i). This is a total charge of at most $a_2 + b_2 + 2vp$ (there may be two charges vp: one for a segment ending at the deepest node at right depth 1 and one for a segment ending at the deepest node at right depth 0). In this case we do not provide potential c to a node displaced from the root of its h-block, for $h \le i$. Instead, we eventually provide the unlabeled portions of the extreme paths of such blocks with new path potentials. Recall that the old root of the h-block was unlabeled because only a top portion of the chain adjacent to the root was traversed (see Section 2). As we will see, for each such block there are two possibilities.

To understand the possibilities, consider the right access path up to the right i-header. Consider an h-block B, $h \le i$, whose left extreme path is partially on the right access path. If a chain of block B has a node on the right access path, then the bottommost node of this chain is on the right access path (for its left child, if any, on the right access path is the root of another h-block). Thus at any point, for each $i \le vis$, for each $h \le i$, there is at most one chain belonging to an h-block, for which nodes remain on the right access path and for which paths of unlabeled nodes have been created. It remains to explain how to pay for these unlabeled paths. This happens in one of two ways.

(i) The chain becomes a path (i.e., all its nodes are contiguous in the splay tree). The above process may have created up to two paths of unlabeled nodes without associated path potentials (note we intend paths and not just chains); one path was created while the root of the block had right depth 1, and the other was created while the root had right depth 0. These two paths are provided with path potentials by using the reserve path potential associated with the chain.

(ii) An access is made to a new i-block. Besides the h-block which was being accessed previously, there are at most $2(i-h+1)$ h-blocks, $h \leq i$, which have extreme paths with unlabeled nodes without associated path or chain potentials. The additional h-blocks are those whose roots are the j-leaders, $h \leq j \leq i$, if these j-leaders are the highest or second highest nodes on their j-paths. Again, each such block has at most two unlabeled paths. These paths are provided with path potentials and the cost is charged to the new access. This is a charge of at most $\sum_{h=1}^{i} 8c(i-h+1)\lfloor \log c_h \rfloor \leq 8c\sum_{h=1}^{i}(i-h+1)2^{h+1} \leq 64c \cdot 2^i$.

Now, suppose that the right i-header is not available. Then it need not be the case that its right height will be reduced when next traversed. However, the right i-header then becomes available. Note that once available, the right i-header can cease to be available only if it is traversed, at least during accesses to the same i-block. To cover the cost of the traversal of an unavailable right i-header, we provide it with an additional $cost_{i+1}$ potential. Outside of the initialization cost, this is obtained by doubling the charge for the traversals of the right i-header when available; i.e., in this case a charge of $2cost_{i+1}$ needs to be covered. Thus it suffices to have

$$c_5 \geq 2 \tag{4.1}$$

$$d_5 s'/2 \geq 2 \tag{4.2}$$

When an access to a new i-block is made, following the access, the potentials for the right and left h-headers are reinitialized, for $h \leq i$, at a cost of

$$2\sum_{h=1}^{i}[c_5 d_5 h cost_{h+1}^2 + cost_{h+1}] \leq 2i(i+1)c_5 d_5 cost_{i+1}^2$$

which is bounded by

$$2(i+1)^2 c_5 d_5 (a_3\alpha(2^{2^{i+2}}) + b_3 + vp + 2c)^2$$

In addition, in an access to a new i-block, the charge for the portion of the access in which the accessed item has not yet reached the inner $(i+1)$-skeleton totals

$$\sum_{1\leq h\leq i}[(4e'+9)gp \cdot (2^{h+1}+1) + vp + 2c \cdot h]$$

(see Equation 3.3). The costs of the remainder of the access are covered by the analysis of this section; they total:

$$a_2 + b_2 + 2vp + 64c \cdot 2^i$$

Finally, we note that each node on the access path spends at most 3 sets of $s'/2$ spares (one set is used for the pseudo-traversals of Section 3.1, one set for traversing an extreme path of a lazy tree, and one set is used for the analysis of this section). This is a total of $s/2$ spares per traversed node; these spares are at hand for s spares are provided per couple.

On taking equality in Equations 4.1, 4.2, we obtain $c_5 = 2$, $d_5 = 12$. On summing the charges for this section, we obtain a charge of $48[(i+1)\alpha(2^{2^{i+2}})]^2(6\cdot 10^6)^2 + 10{,}000\cdot 2^{i+1} + 4{,}300i + 34i^2 + 3\cdot 10^6$ for an access to a new i-block which is an access to the same i-superblock. In addition to the initialization costs present previously, we also have to initialize the right and left h-headers, for $h \ge 1$; this costs less than $10^{16}(\lceil \log\log n\rceil \alpha(n))^2$. Using the argument of Section 3.7, we obtain:

THEOREM 4.1. *Consider a sequence of searches performed on an n-node splay tree, $n \ge 16$. Let item v be the current item being accessed and u the previous item accessed. Suppose that the distance from u to v is d. (For the first search, $d = n$.) Then the amortized cost of the access, in rotations, is bounded by $42{,}000\log d + 10^{16}(\lceil \log\log d\rceil \alpha(d^2))^2$, where the cost of initializing the potential is at most $6000n + 10^{16}(\lceil \log\log n\rceil \alpha(n))^2$.*

5. Insertions and Deletions. In this section, the result is extended to incorporate insertions and deletions. The main idea is to allow blocks to have varying sizes and to combine or partition them as they become too small or too large. The previous bounds on block sizes will become lower bounds: An i-superblock will contain at least $b'_{i+1} = 2^{2^{i+1}}$ i-blocks and at most $b_{i+1} = 8b'_{i+1}$ i-blocks. Now, a 1-block will contain between 16 and $8\cdot 16$ items. There will be no undersize blocks, except possibly at the topmost level, where there is just one vis-block, which contains all the $(vis-1)$-blocks. In fact, with one exception dealt with later, an i-superblock will contain at most $4b'_{i+1}$ i-blocks.

We use the following rule: when an i-superblock reaches its usual upper boundary size ($4b'_{i+1}$ i-blocks), it is partitioned into two i-superblocks, one containing the leftmost $2b'_{i+1}$ i-blocks and the other the rightmost $2b'_{i+1}$ i-blocks. When an i-superblock reaches its lower bound size, it is combined with a neighboring i-superblock in the same $(i+1)$-superblock; this new i-superblock is partitioned in turn if it has size at least $4b'_{i+1}$ i-blocks. If the vis-block is partitioned, then we need to create a new higher level block, a $(vis+1)$-block, to contain the two vis-blocks at hand. Likewise, if there are only 2 $(vis-1)$-blocks and they are combined, then the vis-block is removed and the top level block is at level $vis-1$.

COMMENT. We use the terms combine and partition rather than the more natural join and split to avoid confusion with the split of a lazy tree.

We determine the cost of the combine and partition operations on an i-superblock. When performing a partition we have the following costs. Each global node on the two extreme paths between the new superblocks has its rank raised to $\lceil \log 2K\rceil$, where $K = 2^{\lceil \log b_{i+1}\rceil}$. This costs up to $4\cdot 2^{2^{i+1}}\cdot(2^{i+1}+3)\cdot gp$. In addition, a path potential has to be provided for up to four new chains: these are the chains starting and ending at the node that has newly become the root of an i-superblock, chains which are also parts of the extreme paths of i-superblocks. An i-superblock contains at most $4^{i+1}\cdot 2^{2^{i+2}}$ items. So a chain can contain at most one fewer items, and hence the cost of the new chains is at most $24c[2(i+1)+2^{i+2}]+4c$ (see Section 2). In addition, on the new chains, the i-global nodes may need to be provided with an extra zig-zag potential of c units each, if they are not adjacent to two chain nodes. In fact this only applies to the global nodes newly on an extreme path of an i-superblock, which is at most $4b'_{i+1}$ nodes. So this is a further cost of $4c\cdot 2^{2^{i+1}}$. Finally, the new root of an i-superblock receives its $(i+1)$-global potential, which costs at most $(2^{i+1}+3)gp$. This in turn may raise the global potential of some of its ancestors for a further cost of $(2^{i+1}+3)gp$. The total cost of the partition is

$$part = 2^{2^{i+1}}[4gp(2^{i+1}+3)+4c] + 2^{i+1}\cdot(48c+2gp) + i\cdot 48c + (52c+6gp)$$

The combine has zero cost.

In addition, when a combine or partition is performed, we treat it as ending the current sequence of accesses to the superblock being combined or partitioned. This entails additional costs of $48[(i+2)\alpha(2^{2^{i+3}})]^2(6\cdot 10^6)^2 + 10{,}000\cdot 2^{i+2} + 4{,}300(i+1) + 34(i+1)^2 + 3\cdot 10^6$ (see Section 4).

In order to cover these costs, each time an i-block is combined or partitioned, it provides the following potential to its parent (the i-superblock in which it is contained): $p_{i+1} = 4gp(2^{i+1}+3) + 4c + [2^{i+1}\cdot(48c+2gp) + i\cdot 48c + (52c+6gp) + 48[(i+2)\alpha(2^{2^{i+3}})]^2(6\cdot 10^6)^2 + 10{,}000\cdot 2^{i+2} + 4{,}300(i+1) + 34(i+1)^2 + 3\cdot 10^6 + p_{i+2}]/2^{2^{i+1}}$. It can be checked that $p_{i+1} = 152\cdot 2^{i+1} + 10^{16}$ suffices. The accumulation of this potential by an i-superblock will ensure that when and if it is removed by a combine or partition it will have enough potential at hand to pay for its removal and for the potential that has to be provided to its parent block; for between the creation and removal of an i-superblock there will have been at least b'_{i+1} combines and/or partitions of its i-blocks. At level 0, when an item is inserted or deleted, it needs to provide potential p_1 to its 1-block, but this is $O(1)$.

Next, we need to modify the analysis. For the moment, we ignore the question of accesses that straddle a block boundary. A search is treated as before. An insertion is analyzed as follows. The item is inserted as in a binary search tree. After adding the item, we perform any necessary block partitions. The item is splayed now. When the item is added, it is not provided with any potential, nor are the appropriate chain potentials increased. The reason is that the item does not need any associated potential in order to be splayed; the potential is needed at nodes which are traversed. The access is analyzed as before: as the access proceeds, the item will be provided with potential as described in the earlier analysis. So the cost of an insertion at distance d from the previous access is p_1 plus the cost of a search at distance d.

For the deletion, at least for our analysis, we need to modify the operation described by Sleator and Tarjan [ST85]. We offer reasons as to why such a modification may be needed in order to obtain the finger search result we seek. Sleator and Tarjan proceed as follows. First the item e is splayed, placing it at the root. Then the resulting two subtrees of e are joined (for example, by splaying the rightmost item in the left tree, and then making the right tree its child). Our method differs. First we swap the item, e, to be deleted with its predecessor, if e has a left child. Then the current predecessor of e (previously, the second predecessor, if e had a left child at the start of the operation) is splayed. Following the splay, e has no children, and it is removed. The reason for our choice of deletion method is that the change in potential owing to the removal of item e might be non-constant if it is the root of a block.

There is an unpleasing assymmetry to the operation described above. An alternate implementation is to perform two splays, in turn on the predecessor and successor of the item, e, to be deleted; then e is deleted — it is the right child of the left child of the root of the splay tree and has no children at this point. The accesses are analyzed as before and the deletion can only reduce the potentials associated with the various blocks. The unattractive feature here is that two searches and two splays are needed.

Now, we analyze the first implementation of the delete operation. The access is analyzed as before. The only novel feature is the elimination of an item, which may add nodes to the extreme path of the blocks to which the item belongs and thereby increase the potential of the splay tree; but this effect can arise only if the item in question is an extreme item in its block. In fact, at the time item e is deleted it is childless, so no nodes are added to the extreme paths of any blocks; indeed, the removal of e can only shorten these paths, thereby reducing the potential. So the cost of a deletion is at most c_1 plus the cost of an access at distance $d+2$.

It remains to deal with nearby accesses that straddle a block boundary. First we explain how to form sequences of accesses. Consider the i-blocks. The accesses are partitioned into maximal subsequences S_i of accesses to pairs of neighboring i-blocks. Whenever an i-block is partitioned or combined a new sequence is started.

It may be that a particular sequence S_i accesses only one i-block. But suppose that S_i accesses two neighboring i-blocks, B_1' and B_2'. If they are contained in distinct i-superblocks, B_1 and B_2, respectively, then at the start of the sequence S_i, B_1 and B_2 are combined into i-superblock B; at the end of the sequence B is split into B_1 and B_2 with the new boundary being between B_1' and B_2', as before. B_1' and B_2' still exist at this point in time for no partition or combine of these blocks has been caused by their becoming too large or small during sequence S_i, although they may have been combined and partitioned because of accesses straddling neighboring $(i-1)$-blocks. As before the cost of the combine is zero. The cost of the split is modest, because the most recent accesses to the combined B_1 and B_2 are to adjacent i-blocks B_1' and B_2'. We need only provide the following potentials: chain potential for up to three new chains, which costs at most $18c[2(i+1)+2^{i+2}]+3c$, and the $(i+1)$-global potential for the new root of an i-superblock, which costs at most $(2^{i+1}+3)\cdot gp$. There are no i-global nodes whose potential needs increasing.

The subsequence S_i is further partitioned, so that all the accesses in each new subsequence are to the same block. Suppose the i-blocks B_1' and B_2' are not combined for some pair of accesses of S_i which straddle the boundary between i-superblocks B_1 and B_2; this implies these two accesses are to non-neighboring i-subblocks. So an access to a non-neighboring $(i-1)$-block, i.e., an access at distance at least 2^{2^i}, may start a new i-sequence of accesses to an i-block. Let d denote the distance spanned by this access; then $d+2 \geq 2^{2^i}$ (the $d+2$ takes account of the possible additional shift by 2 due to a delete operation). Following the analysis of Section 4, the cost of this access is bounded by:

$$\sum_{1\le h\le i} [(4e'+9)gp\cdot(2^{h+1}+4)+vp+2c\cdot h]$$
$$+\sum_{1\le h\le i} 2(i+1)^2 c_5 d_5 (a_3\alpha(2^{2^{i+2}})+b_3+vp+2c)^2+p_1+a_2+b_2+2vp+64c2^i$$

The term $(2^{h+1}+4)$ reflects the increased maximum rank on the boundary of an h-block. The cost is bounded by $new\_cost_i = 20,000\cdot 2^i + 10^{16}[(i+1)\alpha(2^{2^{i+2}})]^2+10^{16}$. In turn this is bounded by $20,000\log(d+2)+10^{16}[(\log\log(d+2)+1)\alpha(2^{4\log(d+2)})]^2+10^{16}$.

The last step in the analysis is to account for a series of accesses which each shift a small distance, but cummulatively shift a large distance and cause a high index block boundary to be crossed. Clearly a simple amortization can handle this; we specify this precisely next. Suppose the current access is at distance d from the previous access, where $2^{2^i} \le d+2 < 2^{2^{i+1}}$. Then potential $2\textit{new-cost}_i\cdot 5^{2-h}$ is provided to the $(i+h)$-block to which the previously accessed item belongs, for $h\ge 0$. Consider a maximal sequence S_j of accesses to a j-block, B_j, and the access immediately following it. The distance spanned by these accesses is at least $2^{2^j}-2$, for otherwise the accesses would all be in adjacent $(j-1)$-blocks and hence in the same j-block (because of our combining rule). We determine the contribution made by these accesses to B_j and show that it is at least $\mathit{new\text{-}cost}_j$. The contribution by an access of length d, where $2^{2^{j-h}} \le d+2 < 2^{2^{j-h+1}}$ is $2\mathit{new\text{-}cost}_{j-h} \cdot 5^{2-h} \ge 2\mathit{new\text{-}cost}_j \cdot 5^{2-2h}$. The total contribution is minimized when each d takes on the maximal value consistent with a given contribution; this yields a total contribution of the form $\sum_l 2\mathit{new\text{-}cost}_j \cdot 5^{2-2h_l}$, where $\sum_l [2^{2^{j-h_l+1}} - 3] \ge 2^{2^j} - 2$; by inspection, the contribution is minimized with $h_l = 2$, when it totals at least $\mathit{new\text{-}cost}_j$.

The additional charge for each operation is $\frac{25}{2}\mathit{new\text{-}cost}_i$, where $2^{2^i} \le d+2 < 2^{2^{i+1}}$.

We have shown:

THEOREM 5.1. *The extension of the Dynamic Finger Conjecture to incorporate the insert and delete operations is true, when the delete is modified as described above. The amortized cost of an access at distance d from the previous access is bounded by* $250,000\log(d+2) + 10^{18}[(\log\log(d+2)+1)\alpha(2^{4\log(d+2)})]^2 + 10^{18}$.

6. Improved Constants. Our conjecture is that the true bound has much smaller constants than those shown here. We briefly sktch one approach for obtaining such a bound.

The main difficulty in improving the bounds lies in the potentials needed for the lazy trees. If it were possible to have $\log n$ levels of visibility, where each superblock contained just a constant number of blocks, we would avoid the need for global potentials and hence for lazy trees. There would still be a need for potentials associated with extreme paths (small and large debits and chain potentials) but changes in these would all be paid for by increases in visibility. This would now be of size $\Theta(\log n)$, rather than $\Theta(\log\log n)$, so there would be a continued need to pay for level jumps (see Section 4). Unfortunately, following the analysis used there would lead to an $O(\log^2 d)$ cost for an access at distance d from the previous access. It appears possible to improve this, by not providing the chain potentials in full, and instead taking a *lazy* approach. In particular, we would provide only $O(1)$ potential for each individual jump in visibility on the access path, however large. This eliminates the need for potentials associated with right i-headers; offsetting this, we now have to handle "missing" chain potentials. We have not worked out all the details, however, for it appears that we still need an inverse Ackerman term, arising for similar reasons to the inverse Ackerman term used in analyzing paths P_3 (see Section 3.3.4). At present we do not see how to amortize this term, and it leads at the very least to an additive inverse Ackerman term on the amortized cost of each access. In fact, we suspect that this bound can be reduced to $O(1)$. Perhaps this would follow from improving Sundar's bound for the Dequeue Conjecture [Su89], or some variant of this conjecture.

7. Acknowledgements. We thank an anonymous referee for making a number of detailed suggestions that helped improve the paper's presentation. Any remaining errors or lack of clarity are solely due to the author.

REFERENCES

[CMSS98] R. COLE; B. MISHRA, J. SCHMIDT, A. SIEGEL, *On the Dynamic Finger Conjecture for splay trees. Part I: Splay Sorting* $\log n$*-block sequences*, SIAM J. Comput., to appear.

[Luc88a] J.M. LUCAS, *Arbitrary splitting in splay trees*, Technical Report No. DCS-TR-234, Computer Science Department, Rutgers University, 1988.

[Luc88b] ———, *Canonical forms for competitive binary search tree algorithms*, Technical Report No. DCS-TR-250, Computer Science Department, Rutgers University, 1988.

[ST85] D.D. Sleator, R.E. Tarjan, *Self-adjusting binary search trees*, JACM, 3(1985), pp. 652–686.

[STT86] D.D. Sleator, R.E. Tarjan, W.P. Thurston, *Rotation distance, triangulations, and hyperbolic geometry*, Proc. 18th Symposium on Theory of Computing, 1986, pp. 122–135.

[T85] R.E. Tarjan, *Sequential access in splay trees takes linear time*, Combinatorica, 5(4), 1985, pp. 367–378.

[Su89] R. Sundar, *Twists, turns, cascades, deque conjecture, and scanning theorem*, Proc. 13th Symposium on Foundations of Computer Science, 1989, pp. 555–559.

[W86] R. Wilber, *Lower bounds for accessing binary search trees with rotations*, Proc. 27th Symposium on Foundations of Computer Science, 1986, pp. 61–69.