SIAM J. COMPUT.
Vol. 32, No. 4, pp. 1056–1066

TREE PATTERN MATCHING TO SUBSET MATCHING IN LINEAR TIME*

RICHARD COLE† AND RAMESH HARIHARAN‡

Abstract. In this paper, we show an $O(n+m)$ time Turing reduction from the tree pattern matching problem to another problem called the *subset matching* problem. Subsequent works have given efficient deterministic and randomized algorithms for the subset matching problem. Together, these works yield an $O\left(n\log^2 m+m\right)$ time deterministic algorithm and an $O(n\log n+m)$ time Monte Carlo algorithm for the tree pattern matching problem.

Key words. tree pattern matching, subset matching

AMS subject classifications. 68W05, 68W40

DOI. 10.1137/S0097539700382704

1. Introduction.

In the tree pattern matching problem, the text and the pattern are ordered, binary trees, and all occurrences of the pattern in the text are sought. Here, the pattern occurs at a particular text position if placing the pattern with root at that text position leads to a situation in which each pattern node overlaps some text node. This problem has a number of applications (see [6]). Actually, in these applications, the tree need not be binary and the edges may be labelled; however, as shown in [4], this general problem can be converted to a problem on binary trees with unlabelled edges but with a blow-up in size proportional to the logarithm of the size of the pattern. In fact, this blow-up can also be avoided in our approach, as we will indicate in our description.

The naive algorithm for tree pattern matching takes time $O(nm)$, where n is the text size and m is the pattern size. Hoffman and O'Donell [6] gave another algorithm with the same worst case bound. This algorithm decomposes the pattern into strings, each string representing a root-to-leaf path. It then finds all occurrences of each of these strings in the text tree. The first $o(nm)$ algorithm was obtained by Kosaraju [9], who first noticed the connection of the tree pattern matching problem to the problem of string matching with don't-cares and the problem of convolving two strings. Kosaraju's algorithm takes $O(nm^{.75}\log m)$ time. Dubiner, Galil, and Magen [4] improved Kosaraju's algorithm by discovering and exploiting periodicities in paths in the pattern. They obtained a bound of $O(nm^{.5}\log m)$. This was the best bound known to date. Dubiner, Galil, and Magen also made the observation that the naive algorithm actually takes $O(nh)$ time, where h is the height of the pattern.

In this paper, we show how to reduce the tree pattern matching problem to the *subset matching* problem in linear time. The subset matching problem is to find all

*Received by the editors March 29, 2000; accepted for publication (in revised form) April 25, 2003; published electronically July 8, 2003. This work was supported in part by NSF grants CCR9202900, CCR9503309, CCR9800085, and CCR0105678. This work is based on an earlier work: "Tree pattern matching and subset matching in randomized $0(n\log^3 m)$ time," in Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC'97), © ACM, 1997. http://doi.acm.org/10.1145/258533.258553.

http://www.siam.org/journals/sicomp/32-4/38270.html

†Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012-1185 (cole@cs.nyu.edu).

‡Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India (ramesh@csa.iisc.ernet.in). This work was done in part while this author was visiting NYU.

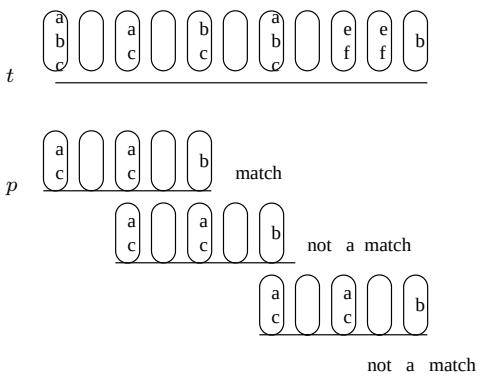

FIG. 1. *Example of subset matching.*

occurrences of a pattern string p of length m in a text string t of length n, where each pattern and text location is a set of characters drawn from some alphabet. The pattern is said to occur at text position i if the set $p[j]$ is a subset of the set $t[i+j-1]$ for all j, $1 \leq j \leq m$. It is required to find all text locations at which the pattern matches; i.e., each pattern set is a subset of the aligned text set (see Figure 1).

The reduction from tree pattern matching to subset matching proceeds in two steps.

• We show that the general tree pattern matching problem can be reduced to the following special case, called *spine pattern matching*, by a linear time Turing reduction. In spine pattern matching, there is a special path in each of the pattern and text called their spines. The spine begins at the root of its tree, and in addition each node on the spine has at most one nonspine child. Spines have additional properties as well, which will be described later. All matches of the pattern in the text are sought with the additional restriction that the spine of the pattern must match a portion of the spine of the text; i.e., nodes on the pattern spine must be aligned with nodes on the text spine. For intuition, one can think of the spine as being the path of left children starting at the root (and in fact one can reduce the general problem to this case in linear time, although we will not do so).

The above reduction may create several instances of the spine pattern matching problem, but the sum of the sizes of these instances will be linear. This reduction is completely deterministic. It proceeds by using the periodicity structure of paths and by decomposing the text tree into periodic paths in a nontrivial manner. Each path then gives a spine for the spine pattern matching problem.

• Next, we reduce the spine pattern matching problem to the subset matching problem in linear time. This is, in fact, readily done. The spine of the text tree gives the text string for the subset matching problem; the subtrees hanging from this spine determine the various text sets. Analogous facts hold for the pattern.

The two reductions above imply that the tree pattern matching problem can be reduced to several instances of the subset matching problem, the sum of the sizes of these instances being linear. Therefore, an algorithm for the subset matching problem yields an algorithm for the tree pattern matching problem with the same time complexity.

Cole and Hariharan [1] gave a randomized algorithm for the subset matching problem running in time $O((n+s)\log^3 m)$, where s is the sum of the sizes of all

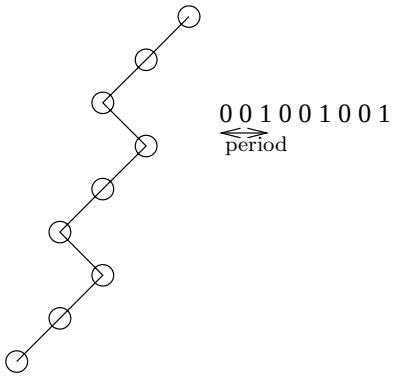

FIG. 2. *A path and its associated string.*

the pattern and text sets. Subsequently, Indyk [7] gave a deterministic algorithm for the subset matching problem running in time $O((n+s)m^{\sqrt{\frac{\log\log m}{\log m}}(1+o(1))})$. Later, Cole, Hariharan, and Indyk [3] gave a deterministic algorithm running in time $O((n+s)\log^3 m)$ and a randomized algorithm running in time $O((n+s)\frac{\log^3 m}{\log\log m})$. Indyk [8] also gave a randomized Monte Carlo algorithm with running time $O(cn\log n)$ and failure probability $O(1/n^c)$ for any fixed constant $c \geq 1$. Finally, Cole and Hariharan [2] gave a deterministic algorithm running in time $O(n\log^2 m)$. It follows that there is a deterministic algorithm running in time $O(n\log^2 m)$ and a Monte Carlo randomized algorithm running in time $O(n\log n)$ for the tree pattern matching problem.

This paper is organized as follows. Section 2 gives some required definitions. Section 3 describes the reduction of the spine pattern matching problem to the subset matching problem. Section 4 describes the reduction from the tree pattern matching problem to the spine pattern matching problem.

2. Definitions.

DEFINITION 2.1 (tree pattern matching). *We consider ordered binary trees; i.e., each internal node has a left and/or a right child. The text tree t has n nodes and the pattern tree p has m nodes. The problem entails finding all nodes v in t where p matches; i.e., when the root of p is aligned with v, each node in p is aligned with a node in t.*

DEFINITION 2.2 (paths, strings, and periods). *Note that paths in trees p and t can be expressed as strings over a two character alphabet, one character signifying a left edge and the other a right edge (see Figure* 2: 0 *represents a left edge and* 1 *a right edge). The* period *of a string $s[1\ldots|s|]$ is the smallest number $j > 0$ such that $s[i] = s[i+j]$ for all i, $1 \leq i \leq |s| - j$. If no such j exists, then the period of s is defined to be $|s|$. The* period *of a path is defined to be the period of its associated string.*

It is well known that the period can be computed in linear time [5]. The following lemma is classical [10].

LEMMA 2.3. *If $k \leq |s| - j$ is such that the period j of s does not divide k, then the string $s[k+1\ldots k+j]$ differs from the string $s[1\ldots j]$.*

DEFINITION 2.4 (spine pattern matching). *This is a restricted version of the tree pattern matching problem. In this problem, the text and the pattern each have one designated path, called their* spines. *The text and pattern spines originate at their*

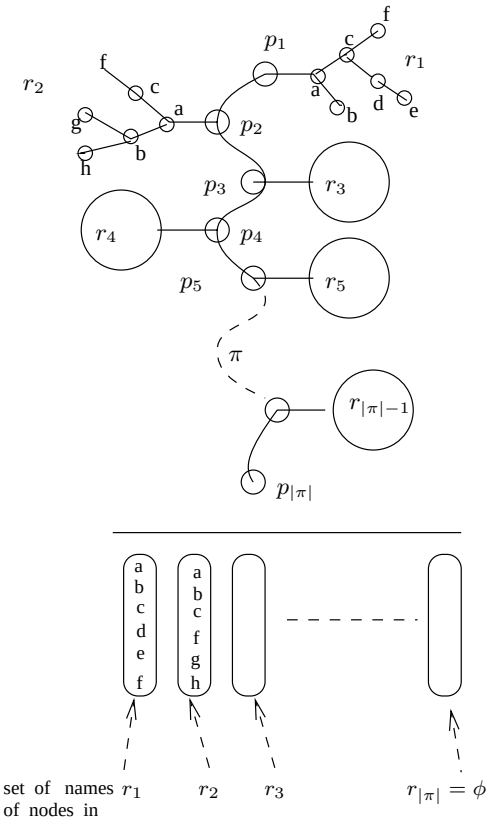

FIG. 3. *The spine and its associated set string.*

respective roots and are maximal paths having the same period, θ say (the θ needed for tree pattern matching will be determined later). If the input does not have this form, it is not a legitimate input for this problem. In fact, both spines when represented as strings will have the form $x^k x'$, where $|x| = \theta$ and x' is a prefix of x (here, the values of k and x' could differ for the pattern spine and the text spine, but x is identical for both spines). All matches of the pattern in which the pattern spine falls completely on the text spine are sought.

From maximality, it follows that both spines terminate at nodes with at most one child (a child which when added to the spine destroys its periodic structure). Since both spines have the same period θ, it follows that the pattern spine will fall completely on the text spine only if the root of the pattern is placed at certain nodes on the text spine. These nodes will occur at integer multiples of θ from the text root and will be designated "anchor nodes."

3. Reducing spine pattern matching to subset matching. The spines of the pattern p and the text t will define the strings for the subset matching problem. The subsets at each location in these strings will correspond to the off-spine subtrees of the spine nodes; an *off-spine subtree* is a subtree whose root is a nonspine node but the parent of whose root is on the spine. These subsets are obtained by labelling the nodes of the off-spine subtrees as follows (see Figure 3). The key fact about this labelling is that two nodes in two distinct off-spine subtrees (both of which could be

in the pattern or in the text, or, alternatively, one could be in the pattern and the other in the text) get the same label if and only if the paths from these nodes to the roots of their respective off-spine subtrees represent identical strings.

The off-spine subtrees of p are labelled first. The subtrees are overlaid to form a *combined pattern subtree*; the overlaying aligns the roots of the off-spine subtrees and recursively overlays their subtrees. Then the combined pattern subtree is traversed by any convenient method, e.g., a breadth first traversal, and the nodes are labelled by the associated numbering. For each spine node, we form a subset consisting of the collection of numbers labelling the nodes of its off-spine subtree. This collection of subsets defines the pattern for the subset matching problem instance. The off-spine subtrees of t are labelled using the same labelling. To do this, each off-spine text subtree and the combined pattern subtree are traversed in lock-step. Consider the text subtree laid over the combined pattern subtree. Clearly, any text node that lies beyond the combined pattern subtree will not be part of any match in which the pattern spine is aligned with a portion of the text spline. Consequently, these text nodes need not be and are not given labels, and indeed need not be and are not traversal. As a result, we have the following easy fact about the time complexity of the above computation.

FACT 1. *The labels to nodes in off-spine subtrees of the pattern can be given in* $O(m)$ *time. The labels to any one off-spine subtree* t' *in the text can be given in time* $O(\min\{|t'|, m\})$. *The total time taken for the labelling is thus* $O(n+m)$*; consequently, the size of the resulting subset matching problem is also* $O(n+m)$.

Recall our remark from the introduction that the case of larger degree and labelled trees can be handled without any extra overhead. Larger degree is simply handled by the usual binarization. Labelled trees are handled by pairing the given labels with the labels obtained here.

Clearly, there is a match in the subset matching problem beginning at a location corresponding to an anchor node if and only if there is a match in the spine pattern matching problem with the pattern tree root aligned with the corresponding anchor node. This completes the reduction from spine pattern matching to subset matching.

4. Reducing tree pattern matching to one or more instances of spine pattern matching. Consider two matches of the pattern with the text in which the pattern instances overlap in $m/2$ or more locations. To avoid checking such matches independently, we seek to have the roots of both pattern instances lie on the same instance of a spine obtained from t.

DEFINITION 4.1. *The* size *of a node* v *in a tree is defined to be the number of nodes in the subtree rooted at* v. *Let* t_v *denote the subtree of* t *rooted at a node* v *in* t, *and let* p_v *denote the subtree of* p *rooted at a node* v *in* p.

4.1. Processing the pattern. We define the spine π of the pattern p to be the following path from the root to a node with at most one child. π consists of two segments, π_1 and π_2. π_1 is a centroid path; i.e., it is obtained by moving to the child with larger size at each step, with ties broken arbitrarily. π_1 ends when a node x such that $|p_x| \leq \frac{m}{2}$ is reached. Note that $|p_x| \geq \frac{m}{4}$. Let θ be the period of π_1. π_2 is the longest path starting at x such that the path π continues to have period θ. Note that π_2 has a vertex in common with π_1. π is readily computed in linear time. π_1 is terminated at x rather than at a node with at most one child to guarantee an overall linear-sized construction.

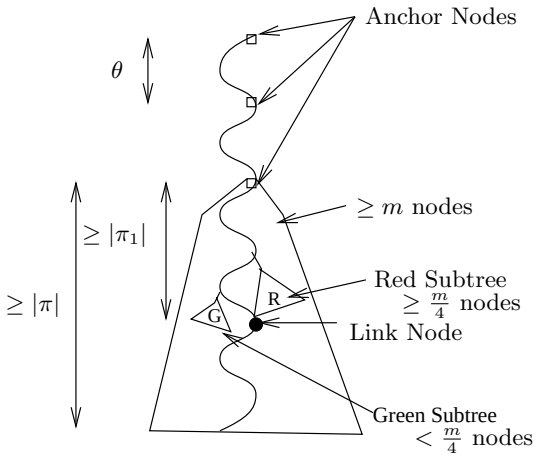

FIG. 4. *A θ-Path in C.*

4.2. Decomposing the text.

DEFINITION 4.2 (see Figure 4). *A path in t from a node u to a node v in t_u is a* θ-path *if it has period θ and is identical to the spine of the pattern in the first θ locations (when both paths are viewed as strings). This path is* maximal *if extending it to the distance θ ancestor of u or either child of v results in a path which is not a θ-path (in fact, v can have only one child). These paths are the candidate spline paths in the text, but we need to impose some further restrictions. Continuing with the definitions, the* link *node l in this path is the node closest to v such that $|t_l| \geq \frac{m}{4}$. An* anchor *node on this path is a node at distance an integer multiple of θ from the start node. The strong* anchor *nodes w on this path also satisfy the following properties. As we will see, matches occur only when the pattern root is aligned with a strong anchor node.*

1. *t_w has at least m nodes.*
2. *The distance from w to l is at least $|\pi_1|$, and thus has length at least θ.*
3. *The distance from w to v is at least $|\pi|$.*
4. *Consider the subtrees hanging from the maximal θ-*path *starting at w. Classify them as* red *subtrees if they have at least $\frac{m}{4}$ nodes and as* green *subtrees otherwise. If all these subtrees except exactly one are green, then the green subtrees plus the path together have at least $m/2$ nodes.*
5. *Either $u = w$ or the distance from u to w is an integer multiple of θ.*

We form a collection C of maximal θ-paths in t, whose start nodes are strong anchor nodes; i.e., they satisfy properties 1–5 above.

Clearly, if any of properties 1–3 or 5 do not hold, there cannot be a match with p's root aligned with w. To see the need for property (4) we argue as follows. Consider a match of p in which at most one of the off-spine subtrees R in t is red. As the spine of p does not match any nodes in R, the subtree of p matching R has size less than $m/2$. Consequently, the remainder of p, of size at least $m/2$, matches the aligned spine portion in t and its green subtrees, which therefore have combined size at least $m/2$.

Note that the paths in C need not be disjoint; however, their combined length will still be $O(n)$, as we shall show later in Lemma 5.18.

The algorithm for constructing these paths is given next.

The path decomposition algorithm. The decomposition is obtained using the following algorithm. For each node x in T, this algorithm first determines the longest θ-path which begins at x. This is done in $O(n)$ time using a Knuth–Morris–Pratt-type automaton in conjunction with a depth-first traversal of t as in the algorithm of Hoffman and O'Donell [6]. Next, the algorithm determines those maximal θ-paths found above which satisfy properties 1–4, discarding all other paths. To this end, it computes the size of each subtree, which allows property 1 to be tested. Properties 2–4 are readily tested by means of a subsequent traversal of each path. It will also be useful to determine, for each such maximal θ-path, whether the node at distance θ from the start of the path is also a strong anchor node. Since, as we will see, the sum of the length of paths in C is $O(n)$, the total time taken above is $O(n)$.

Thus determining matches of p at strong anchor nodes on paths in C suffices to determine all matches of p in t. Further, note that when p is placed with its root at a strong anchor node on some path in C, the spine of p lies completely on that path.

4.3. Processing paths in C. The purpose of processing a path $\rho \in C$ is to determine whether or not p matches at w for each anchor node w on ρ. Each path ρ in C will be processed as follows.

Let u be the node at which ρ starts. u itself is a strong anchor node. Whether or not the pattern matches at u is determined in a brute force manner. This requires $O(m)$ time. We will show in Lemma 5.17 that there are $O(n/m)$ paths, and hence the total time taken over all paths in this process is just $O(n)$.

Matches at other strong anchor nodes on ρ are determined differently, i.e., by reduction to an instance of the spine pattern matching problem.

Consider the portion of ρ starting from the second anchor node onwards, denoted $trunc(\rho)$. If $trunc(\rho)$ starts with a strong anchor node, it provides the spine of the text instance. Clearly, there is a match of p rooted at an anchor node on $trunc(\rho)$ if and only if there is a match at the same location in the corresponding spine pattern matching problem instance.

This concludes the reduction.

5. The analysis. Let $s_1, \ldots, s_{|\rho|-\theta}$ denote the off-spine subtrees, if any, for $trunc(\rho)$, in increasing order of distance from the start node of ρ. Some of the s_i's might not exist. By Fact 1, reducing this instance of the spine pattern matching problem to the subset matching problem takes time $O(\sum_{i=1}^{|\rho|-\theta} \min\{|s_i|, m\} + |\rho| - \theta)$ (plus, of course, $O(m)$ time for processing the pattern, which is common to all the instances of the spine pattern matching problem which result above); also, it yields a text of size $O(\sum_{i=1}^{|\rho|-\theta} \min\{|s_i|, m\} + |\rho| - \theta)$ and a pattern of size $O(m)$.

The total time taken to process ρ is thus $O(m + \sum_{i=1}^{|\rho|-\theta} \min\{|s_i|, m\} + |\rho| - \theta)$. This quantity can be split into four parts: $O(m)$ time for checking for an occurrence of p at the first anchor node, time proportional to its size for each green subtree hanging from $trunc(\rho)$, $O(m)$ time for each red subtree hanging from $trunc(\rho)$, and $O(|\rho| - \theta)$ time for the path itself. We need to show that this sums to $O(n)$ over all paths ρ. By Lemma 5.17, there are $O(n/m)$ paths; hence the first part sums to $O(n)$ time. By Corollary 5.3, the green subtrees in the truncated paths are disjoint; hence the second part sums to $O(n)$. By Lemma 5.8, there are $O(n/m)$ red subtrees; hence the third part sums to $O(n)$. Finally, by Corollary 5.2, the truncated path lengths sum to $O(n)$, and hence the fourth part sums to $O(n)$ also. This yields $O(n)$ time overall. The same argument shows the resulting subset matching problems have texts of total size $O(n)$; also, they each have the same pattern of size $O(m)$.

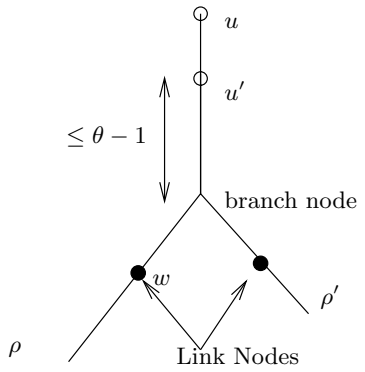

FIG. 5. *Overlap is at most* $\theta - 1$.

5.1. Showing $O(n)$ time.

5.1.1. Some properties of paths in C.

LEMMA 5.1. *Consider two paths* ρ, ρ' *in* C *starting at nodes* u *and* u'*, respectively (see Figure* 5*). Suppose* u' *lies on* ρ*. Then at most the first* $\theta - 1$ *edges of* ρ' *are also present in* ρ*.*

Proof. From the construction of C, the length of the path between u and u' is not divisible by θ. The lemma then follows from Lemma 2.3. ∎

COROLLARY 5.2. *If the first* θ *edges are removed from each path in* C*, then the resulting collection of paths is node disjoint. Hence the truncated paths have total lengths* $O(n)$*. Also, as the link node is not among the first* θ *nodes by property* 2 *of paths in* C*, the link node of* ρ' *cannot lie on* ρ*.*

COROLLARY 5.3. *The green subtrees hanging from the truncated paths are all disjoint.*

Proof. It suffices to consider the green subtrees hanging from two paths $\rho, \rho' \in C$, starting at u, u', respectively, where u is a proper ancestor of u' (for if u and u' are unrelated, then clearly the green trees hanging from ρ and ρ' are disjoint). By Corollary 5.2, $trunc(\rho)$ and $trunc(\rho')$ are disjoint. For a contradiction, suppose that G is a green subtree hanging from $trunc(\rho)$ and containing v, a node in a green subtree hanging from $trunc(\rho')$. It follows from Lemma 5.1 that $trunc(\rho')$ lies within G. But $trunc(\rho')$ includes the first anchor node w' on ρ', and the subtree of t rooted at w' contains at least m nodes. Then G, which contains this subtree, would be red. ∎

LEMMA 5.4. *Let* ρ, ρ' *be as in Lemma* 5.1 *(see Figure* 5*). Then* ρ' *cannot overlap a node in* ρ *which is a proper descendant of* ρ*'s link node* w*. Therefore, if* ρ' *overlaps the link node* w *of* ρ*, then it branches away from* ρ *at* w*.*

Proof. By Corollary 5.2, the link node l' of ρ' is not on ρ. If ρ' overlaps a node w' in ρ which is a proper descendant of w, then $|t_{w'}| \geq |t_{l'}| \geq \frac{m}{4}$, and therefore w cannot be the link node of ρ, a contradiction. ∎

LEMMA 5.5. *Let* ρ*,* ρ'*, and* ρ'' *be three paths whose start nodes appear in this order on the path from the root of* t *to the start node of* ρ''*. Suppose that* ρ'' *overlaps* ρ'*'s link node, and suppose that* ρ' *shares at least one edge with* ρ*. Then* ρ *and* ρ'' *do not overlap.*

Proof. Suppose, for a contradiction, that ρ and ρ'' overlap. Consider the portion q of ρ' up to the start node of ρ''. By assumption, q lies entirely on ρ, and thus by Lemma 5.1, $|q| \leq \theta - 1$. But q is a period of the portion of ρ' from its start node

to its link node, as can be seen by considering the overlap of ρ' with ρ''. Further, this portion of ρ' has π_1 as a prefix, and as $|\pi_1| \geq \theta$, q would also be a period of π_1, contrary to the definition of θ. □

5.1.2. Bounding the number of red subtrees. We bound the number of red subtrees hanging from the truncated paths in C by associating them in part with a set of $O(n/m)$ nodes of t, called *marked* nodes. A path with k red subtrees will have $k-1$ marked nodes.

DEFINITION 5.6. *A node in t is marked if its left and right subtrees both contain at least $\frac{m}{4}$ nodes.*

LEMMA 5.7. *The number of marked nodes in t is $O(\frac{n}{m})$.*

Proof. Each subtree of $m/4$ or more unmarked nodes is contracted to a single unmarked leaf if its parent is marked. Each maximal subtree of unmarked nodes whose parent is unmarked is discarded. Finally, maximal paths of unmarked nodes are replaced by single edges. This reduces the original tree to a new tree in which all internal nodes are marked and correspond one to one to the original marked nodes, and all leaves are unmarked, each corresponding to $m/4$ or more distinct unmarked nodes in t. Further, each internal node has two children. Thus, as there are at most $4n/m$ leaves, there are fewer than $4n/m$ internal nodes. □

LEMMA 5.8. *There are $O(n/m)$ red subtrees hanging from truncated paths whose first node is a strong anchor.*

Proof. Suppose strongly anchored truncated path ρ has $k > 1$ red subtrees hanging from it. Such a path has $k-1$ marked nodes (namely the *lcas* of the red subtree roots). By Corollary 5.3, as these truncated paths are disjoint, and by Lemma 5.7, as there are $O(n/m)$ marked nodes, there can be only $O(n/m)$ anchored truncated paths with two or more red subtrees.

If a strongly anchored truncated path has only one red subtree then by property 4 of the paths, the path together with its green subtrees contains at least $m/2$ nodes. This is also true if it has no red subtrees. By Corollaries 5.2 and 5.3, the paths and their green subtrees are disjoint from each other. Consequently, there are only $O(n/m)$ such paths. □

5.1.3. Bounding the number of paths. It remains to bound the number of paths. To this end, we form suitable collections of paths.

Consider the following procedure for forming collections of paths. A collection starts with a path $\tilde{\rho}$ whose start node is not overlapped by any other path; $\tilde{\rho}$ is called the *root* path of the collection. The collection is built up by iterating the following step. For each path ρ in the collection, every path ρ', whose start node is on ρ but which does not overlap the link node of any other path, is added to the collection. Call these level 1 collections. Level $i+1$ collections are formed in the same way from paths not in any level h collection, $h \leq i$, and ignoring overlaps with paths in level h collections, $h \leq i$.

We will show that there are $O(n/m)$ paths in the collections containing two or more paths. We will then characterize the one path collections and show that they too contain $O(n/m)$ paths.

DEFINITION 5.9. *The* point of attachment *for a level $i+1$ collection C, $i \geq 1$, is defined in terms of C's root path ρ and the parent $\tilde{\rho}$ whose link node $\tilde{l}$ is overlapped by ρ; the point of attachment for C is link node $\tilde{l}$.*

DEFINITION 5.10. *The tree for collection C is the tree formed by the edges on its paths. The reduced tree for collection C is the subtree of its tree rooted at its point of attachment.*

LEMMA 5.11. *Each node in the tree for collection C on the path from the tree's root to its attachment point has a single child.*

Proof. Let $\tilde{C}$ be the collection containing $\tilde{\rho}$, the path whose link node is overlapped by ρ, the root path of C. Suppose, for a contradiction, that ρ' were a path in C, with start node on ρ and which branches away from ρ at or before C's point of attachment. We argue that ρ' would in fact have been added to $\tilde{C}$. Clearly, ρ''s start node lies on $\tilde{\rho}$. By assumption, ρ' does not branch away from $\tilde{\rho}$ at its link node $\tilde{l}$; hence by Lemma 5.4, it branches away before. If ρ' overlapped the link node of another path in $\tilde{C}$, then $\tilde{\rho}$ would overlap this link node too, which is not the case. Thus ρ' can be added to $\tilde{C}$, and so would not be in C, a contradiction. □

LEMMA 5.12. *Let C be a level $i+1$ collection, $i \geq 1$, ρ its root path, $\tilde{\rho}$ the path whose link node is overlapped by ρ, and $\tilde{C}$ the collection containing $\tilde{\rho}$. Then every path overlapping $\tilde{\rho}$'s link node is in C.*

Proof. By Lemma 5.5, if ρ and ρ' both overlap $\tilde{\rho}$'s link node, they do not overlap each other's link nodes. Thus they would both be added to C. □

LEMMA 5.13. *The reduced trees for the collections are disjoint.*

Proof. Without loss of generality, suppose that all the paths are connected (otherwise consider each connected component of paths separately). We will prove that not only are the reduced trees disjoint but that for each i, the reduced trees for level i collections are unrelated (for short the level i reduced trees); i.e., no tree is ancestral to another. The proof is by induction on i, the collection level. Clearly, the trees for level 1 collections are disjoint and unrelated. Now suppose that the reduced trees for each level h collection, $h \leq i$, are all disjoint and that the level i reduced trees are unrelated. Then the link nodes appearing in the level i reduced trees are also unrelated, for each link node appears on only one path in a collection. By construction, the root path of each level $i+1$ collection overlaps the link node of a path in a level i collection. By Lemma 5.12, each such root path overlaps a distinct link node, and thus the level $i+1$ reduced trees are disjoint from each other and are unrelated. □

LEMMA 5.14. *Each collection of $k > 1$ paths has its own $k-1$ distinct marked nodes.*

Proof. The marked nodes for a collection C of k paths are simply the least common ancestors (LCAs) of the link nodes for these paths. As t is binary, and these k link nodes are distinct, there are $k-1$ LCAs. Each LCA has a link node in each of its two subtrees, and consequently each LCA is marked. Finally, Lemma 5.11 implies that the LCAs all lie in the reduced tree for C, and as by Lemma 5.13 these reduced trees are disjoint, the marked nodes associated with each collection are distinct. □

COROLLARY 5.15. *There are $O(n/m)$ paths in collections of two or more paths.*

Now consider the paths in collections of size one. These paths form chains of paths in which each successive path overlaps the link node of its predecessor.

LEMMA 5.16. *There are $O(n/m)$ paths in the above chains.*

Proof. Two sets C_e and C_o of paths are formed. C_e comprises every even index path on the chains (the second, fourth, ... paths) and C_e the odd index paths.

By construction, none of the chains overlap. By Lemma 5.5, none of the paths in C_e overlap (apply the lemma to successive paths ρ_{2i}, ρ_{2i+1}, ρ_{2i+2} on a chain), and the same holds for paths in C_o.

Now we argue in a similar way to the proof of Lemma 5.11. There are $O(n/m)$ paths in C_e with one or more marked nodes (as marked nodes on distinct paths must be distinct). The remaining paths on C_e have at most one red subtree each; by property 4, each such path includes between its nodes and those of its green subtrees

at least $m/2$ distinct nodes (i.e. unshared with any other such path). This is a further $O(n/m)$ paths. Thus C_e includes $O(n/m)$ paths. The same is true for C_o. □

We have shown the following lemmas:

LEMMA 5.17. *There are $O(n/m)$ paths.*

LEMMA 5.18. *The paths have total length $O(n)$.*

This leads to the following theorem.

THEOREM 5.19. *There is a linear time reduction from the tree pattern matching problem to a collection of instances of the subset matching problem of overall linear size.*

Proof. This is immediate from Corollary 5.2 and Lemma 5.17. □

6. Further comments. It is not completely clear whether this construction maps unlabelled trees to the set strings as compactly as possible, for ancestral information is lost in the reduction. Indeed, an unlabelled n-node tree can be represented using $O(n)$ bits, whereas a size n set problem in general requires $\Theta(n \log n)$ bits and will do so after our reduction. In general, n labels would require $\Theta(n \log n)$ bits, so it appears the reduction is tight for labelled trees. Thus this raises the question of whether there are algorithms for unlabelled tree pattern matching that are faster by a $\Theta(\log n)$ factor.

Acknowledgment. We thank the referees for their suggestions which helped improve the presentation.

REFERENCES

[1] R. COLE AND R. HARIHARAN, *Tree pattern matching and subset matching in randomized $O(n \log^3 m)$ time*, in Proceedings of the 29th ACM Symposium on Theory of Computing, El Paso, TX, 1997, pp. 66–75.

[2] R. COLE AND R. HARIHARAN, *Verifying candidate matches in sparse and wildcard matching*, in Proceedings of the 34th ACM Symposium on Theory of Computing, Montreal, QC, Canada, 2002, pp. 592–601.

[3] R. COLE, R, HARIHARAN, AND P. INDYK, *Tree pattern matching and subset matching in deterministic $O(n \log^3 m)$ time*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 1999, pp. 245–254.

[4] M. DUBINER, Z. GALIL, AND E. MAGEN, *Faster tree pattern matching*, J. ACM, 41 (1994), pp. 205–213.

[5] D. GUSFIELD, *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, Cambridge, UK, 1997.

[6] C. M. HOFFMAN AND M. J. O'DONELL, *Pattern matching in trees*, J. ACM, 29 (1982), pp. 68–95.

[7] P. INDYK, *Deterministic superimposed coding with applications to pattern matching*, in Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, Miami Beach, FL, 1997, pp. 127–136.

[8] P. INDYK, *Faster algorithms for string matching problems: Matching the convolution bound*, in Proceedings of the 39th IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, 1998, pp. 166–173.

[9] S. R. KOSARAJU, *Efficient tree pattern matching*, in Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 178–183.

[10] M. CROCHEMORE AND W. RYTTER, *Text Algorithms*, Oxford University Press, New York, 1994, pp. 27–31.