Homework 5

Finding all tandem repeats in a string

Let X be an array of length n containing n distinct elements in sorted order. Given a *finger* i, i.e. an index into X, and an element y where $X[i] \leq y$, a *finger search* for y in X is to find the index j such that $X[j] \leq y < X[j+1]$. Let $d = j - i$ denote the (rank) distance between the finger and the position found.

a) Describe how to perform a finger search in X in time $O(\log d)$.

To represent a dynamically changing sorted list of elements the list can be stored as a (2,3)-tree. A (2,3)-tree is a search tree where all leaves have the same depth, and all internal nodes have two or three children. Elements are stored at the leaves in sorted order from left-to-right, and each internal node v stores a set of keys to guide searches. In the following it is assumed that the keys stored at an internal node v are the (two or three) smallest elements contained in each of the subtrees rooted at a child of v. For the sake of simplicity assume that a (2,3)-tree always contains the value $-\infty$.

The operations needed in the following are

- $\mathsf{Create}(e)$ creates a (2,3)-tree only containing e (and $-\infty$).
- $\mathsf{Insert}(p, e)$ inserts element e as a new leaf to the right of the element pointed to by p, assuming that the insertion preserves the left-to-right ordering.
- $\mathsf{Search}(p, y)$ performs a finger search for y starting at the element x pointed to by p, assuming that $x \leq y$.

The Insert operation proceeds as follows. First a new leaf containing e is created as the right sibling of the leaf pointed to by p. This requires the existence of parent pointers. Let v be the parent of the new node. If v now has degree four, v is split into two nodes of degree two and a new child pointer is inserted in the parent v' of v. If v' now has degree four, v' is split recursively. Note that the nodes being split by an Insert operation all have degree three prior to the Insert operation, and that updating the search keys at the internal nodes takes $O(1)$ time per node being split. In the worst case all nodes on the path from p to the root are split (including splitting the root causing a new root to be created), i.e. the worst case number of nodes being split is $\Theta(\log n)$, where n is the number of leaves of the (2,3)-tree. The amortized number of nodes being split in an Insert operation is much better.

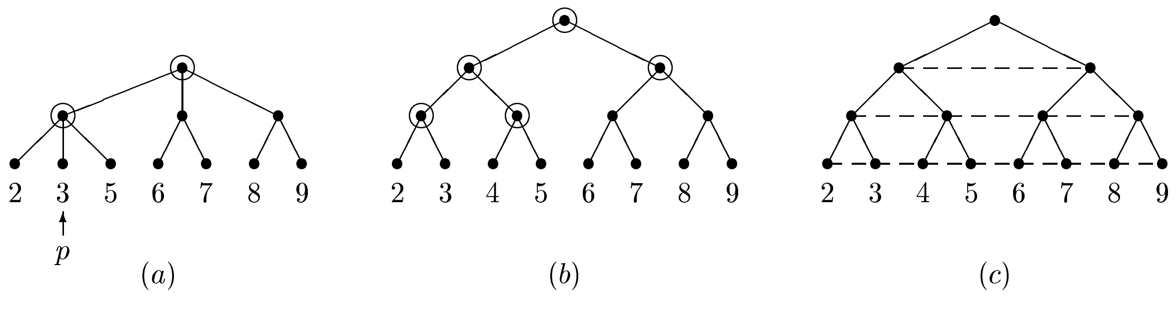

Figure 1: (2,3)-trees: (b) shows the result of performing $\mathsf{Insert}(p, 4)$ on the tree in (a). The marked nodes in (a) are the nodes going to be split, and the marked nodes in (b) are the new nodes. The result of adding level-links to the tree in (b) is shown in (c).

b) Show that $\mathsf{Insert}(p, e)$ on a (2,3)-tree makes amortized $O(1)$ node splittings, i.e. $\mathsf{Insert}(p, e)$ takes amortized constant time.

To support efficiently the $\mathsf{Search}(p, y)$ operation the (2,3)-trees are extended with *level-links*. For each level in a level-linked (2,3)-tree all nodes are kept in a linked list from left-to-right. The level-links are straightforward to maintain with at most a constant overhead for each node splitting.

c) Describe how to perform a finger search $\mathsf{Search}(p, y)$ in a level-linked (2,3)-tree T in time $O(\log d)$, where $d = |\, T \cap [x, y]\,|$ and x is the element pointed to by p.

d) Show how a search for k elements given as a sorted list $y_1 < \cdots < y_k$ can be done in time $O(k + \sum_{i=1}^{k} \log d_i)$ in a (2,3)-tree T with n leaves, where $d_i = |\, T \cap]y_{i-1}, y_i]\,|$ and $y_0 = -\infty$.

e) Show that if $\sum_{i=1}^{k} d_i \leq n$ and each $d_i \geq 1$, then $\sum_{i=1}^{k} \log d_i \leq \log \binom{n+k}{k}$.

f) Show that the searching in d) for the case $k \leq n$ takes time at most $a \log \binom{n+k}{k}$ for an appropriate constant a.

g) Argue that a sorted list of $k \leq n$ elements can be inserted into a (2,3)-tree containing n elements in amortized time at most $a' \log \binom{n+k}{k}$ for some appropriate constant a'.

h) Show that two level-linked (2,3)-trees with respectively n_1 and n_2 leaves can be merged in time at most $a'' \log \binom{n_1+n_2}{n_1}$ for some appropriate constant a''

i) Show that *any order* of merging n level linked (2,3)-trees each of size one to one tree of size n using h) $n - 1$ times takes time $O(\log n!)$, i.e. $O(n \log n)$.

Let $\Sigma = \{a, b\}$ be a binary alphabet. A string $s \in \Sigma^*$ is denoted a *tandem repeat* if $s = ww$ for some $w \in \Sigma^*$, i.e. s is the concatenation of the same string twice. In the following we consider the problem of finding *all substrings* of a string that are tandem repeats. We identify a substring $u = s[i \,..\, i + |u| - 1]$ that is an occurrence of a tandem repeat in s at position i by the integer pair $(i, |u|/2)$. In the string below all occurrences of tandem repeats are underlined. The tandem repeat occurrences are $\{(1, 2), (4, 2), (4, 1), (5, 1), (6, 1), (7, 2), (8, 2), (11, 1)\}$.

$$b \quad a \quad b \quad a \quad a \quad a \quad a \quad b \quad a \quad b \quad a \quad a \quad b$$

The tandem repeats in a string occur as two types: *branching* and *non-branching*. A tandem repeat occurrence (i, d) in s is branching if and only if $s[i + d] \neq s[i + 2d]$, where it is assumed that s is followed by a symbol $\$ \notin \Sigma$. In the above example $(7, 2)$ is a non-branching tandem repeat occurrence whereas $(8, 2)$ is a branching tandem repeat occurrence.

j) Show that if (i, d) is a non-branching tandem repeat occurrence in s, then $(i + 1, d)$ is a tandem repeat occurrence in s.

k) Show that given a list $R_{\rm br}$ of all branching tandem repeat occurrences in s, a list $R \supseteq R_{\rm br}$ containing all tandem repeats can be computed in time $O(|R|)$.

Consider the suffix tree $T(s)$ for $s\$$. Each leaf corresponds to a suffix of s. If ℓ is a leaf with path-label $s[i \,..\, |s|]\$$, we assign ℓ the *leaf-list* $LL(\ell) = \{i\}$. For each internal node v we define the leaf-list $LL(v)$ of v to be the union of all the leaf-lists at the leaves of the subtree rooted at v.

l) Prove that $LL(v)$ is the set of all start positions in s where the path-label of v occurs, and argue that if v is an internal node with children v_1 and v_2, then $LL(v) = LL(v_1) \cup LL(v_2)$.

m) Show that if (i, d) is a branching tandem repeat occurrence in s, then there is an internal node v with children v_1 and v_2 in $T(s)$, such that v has path-label $s[i \mathinner{..} i+d-1]$. Furthermore show that either $i \in LL(v_1)$ and $i + d \in LL(v_2)$, or $i \in LL(v_2)$ and $i + d \in LL(v_1)$.

Assume in the following that leaf-lists are represented by (2,3)-trees.

n) Let v be an internal node in $T(s)$ with path label u and children v_1 and v_2. Assume $L_1 = LL(v_1)$ and $L_2 = LL(v_2)$ have been computed. Using f) show that all branching tandem repeat occurrences uu in s can be computed in time $O\left(\log\binom{|L_1|+|L_2|}{|L_1|}\right)$. Describe how $L = LL(v)$ can be computed from L_1 and L_2 (while destroying L_1 and L_2) in time $O\left(\log\binom{|L_1|+|L_2|}{|L_1|}\right)$.

o) Given $T(s)$ show that all branching tandem repeat occurrences in s can be computed in time $O(n \log n)$.

p) Argue that there are at most $O(n \log n)$ branching tandem repeat occurrences in s.

q) Conclude that all occurrences of tandem repeats R in a string can be computed in time $O(n \log n + |R|)$.