TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in algorithms
Spring Semester, 2011/2012

Homework 2, April 7, 2012

Due on April 29 (Sunday, in my mailbox, second floor in Schrieber). Please keep a copy of the homework !
I apologize for the long questions, I hope you'll find them interesting.

1. In this question, you'll make precise some facts that we mentioned in class. Please answer rigorously. (Assume that all items in the trees below have distinct keys.)

a) Describe precisely an implementation of insert, concatenate, and split for 2-4 finger search trees with fingers at both ends. Recall that a concatenate takes two trees T_1 and T_2 such that all the elements in T_1 are smaller than all the elements in T_2 and returns one tree T containing all the elements that are either in T_1 or in T_2. Split takes a tree T and a key x and returns two trees T_1, containing all elements smaller than x in T, and T_2 containing all elements larger or equal to x in T.

b) Prove that your implementation has the following properties:
1) Concatenate and split take $O(\log n)$ worst-case time, where n is the total number of elements in the tree/trees participating in the operation.
2) Consider a sequence of inserts, concatenate, and split operations starting from an empty tree. Concatenate of a tree T_1 with n_1 items to a tree T_2 with n_2 items takes $O(\log(\min\{n_1, n_2\}))$ amortized time. Split at x_i or insert of x_i, takes $O(\log(\min\{d_i, |T_i| - d_i\}))$ amortized time, if x_i is the d_i^{th} smallest element in T_i.

c) Assume that we start with a single finger search tree T containing n elements with distinct keys (and parent pointers), and perform a sequence of split operations as follows. Each split operation obtains as an input a pointer to a leaf containing element x in one of the trees T' and splits T' into two trees one containing all elements larger than or equal to x and the other containing the elements smaller than x. Argue that the total cost of the splits is $O(n)$.

2. In this problem we will figure out an interesting connection between Monge matrices, the SMAWK algorithm, and search trees which are optimal for a particular access distribution.

Consider n elements, $k_1, \ldots, k_n$, such that the probability which we access k_i is p_i. We want to find the binary tree T, with $k_1, \ldots, k_n$ at its leaves, that minimizes $\sum_i p_i d_i$, where d_i is the depth of k_i. We will do that by the following steps. (Recall that in class we did not find the optimal tree but one which is close to optimal.)

a) Define $w(i,j) = \sum_{\ell=i}^{j} p_\ell$. Let $B[i,j] = 0$ if $i = j$, and $B[i,j] = w(i,j) + \min_{i<t\leq j}\{B_{i,t-1} + B_{t,j}\}$. Argue that $B[1,n]$ gives the cost of the optimal tree. Give an $O(n^3)$ time algorithm to compute the optimal tree based on this observation.

b) Show that $B[i,j]$ has the Monge property. That is for every $1 \leq i \leq i' \leq j \leq j' \leq n$,

$$B[i,j] + B[i',j'] \leq B[i',j] + B[i,j'].$$

c) For every $1 \leq d \leq n$, define a matrix B^d. The matrix B^d is partially defined: For $1 \leq i < t \leq i+d \leq n$ we have

$$B^d[i,t] = w(i,i+d) + B[i,t-1] + B[t,i+d] \ .$$

Other entries are undefined. Prove that B^d has a Monge property in the part where it is defined. (Hint: recall that adding a constant to each row preserves the Monge property, and transposing a matrix preserves the Monge property...) Show how to use the SMAWK algorithm to compute row minima of B^d.

d) Based on your answer to (c) show how to compute the optimal search tree in $O(n^2)$ time.

3. Consider the problem where we want to maintain points in R^2, such that given $x_1 \leq x_2$, and k we can find the k points with the largest y coordinate among all points (x,y) such that $x_1 \leq x \leq x_2$. (Assume that the points have distinct x and y coordinates.)

For that we maintain the points sorted by their x-coordinate as leaves in a $2-4$ tree T. In each internal node v we store a secondary finger search tree T_v with all the points in the subtree of v sorted by their y-coordinate.

a) Show how to answer a query in $O(\log n + k \log\log n)$ time using this data structure.

b) We add support for insertions and deletions of points as follows. To insert a point $p = (x,y)$ we simply insert it to T and to each secondary tree T_v of a node v along the path to p's leaf in T. If a node v along the path splits into v_1 and v_2 we construct the secondary trees T_{v_1} and T_{v_2} using T_v. To delete a point $p = (x,y)$ we delete it from T and from every secondary tree T_v of a node v along the path to p. When we merge nodes, we construct the tree for the new node resulting from the merge from the trees of the nodes which we merged. When node v_1 steals a child from a node v_2 we reconstruct T_{v_1} and T_{v_2}.
How much time can a sequence of m operations on an arbitrary initial tree with at most n elements take ? (For every m show a bad sequence of operations and prove a lower bound on its running time.)

c) To improve the amortized complexity of update operations we work with a tree defined as follows, instead of the 2-4 tree as the primary tree:
1) Each leaf is at the same distance from the root (as in a 2-4 tree).
2) If v is an internal node node at level h (leaves are at level 0) then the size of its subtree $s(v)$ is at most $2 \cdot 8^h$
3) For all internal nodes except the root $s(v) \geq \frac{8^h}{2}$.
4) The root has more than one child.
Prove that all internal nodes have at most 32 and at least 2 children.

d) Describe an algorithm for inserting and deleting nodes into the tree defined in part (c) above. Design the algorithm such that if we use this tree as the primary tree for maintaining a dynamic set of points in the plane subject to insertions, deletions, and queries as in the beginning of the question, then a sequence of m update operations (insert or delete) on a data structure containing at most n points takes $O(m \log^2 n)$ time.