TEL AVIV UNIVERSITY
Department of Computer Science
0368.4281 – Advanced topics in data structures
Fall Semester, 2004/2005

Homework 5, Jan 15, 2005

Due on Monday Jan 31. Don't forget to keep a copy of the homework.

1. Let $x_1, \ldots, x_n$ be an ordered sequence of elements. At any time this sequence is partitioned into an ordered set of $1 \leq k \leq n$ intervals each containing a consecutive subset of the elements. The partition into intervals is dynamic and manipulated via the following operations:
1) Split(j): The interval I containing item j is split into two intervals. One containing all the elements in I preceding or equal to j. The second contains all elements in I that follow j if there are any. If there are no elements after j in I the operation leaves I intact and does nothing.
2) Concat(j): This operation combines the interval I containing item j with the interval immediatly to its right. If I is the last interval the operations does nothing.
3) SameInterval(i,j): This operation returns true if i and j belong to the same interval in the current partition.

Describe an efficient data structure for this problem. Assume we start such that there is only one interval containing the entire sequence. (Look for a data structure with time bound of $O(\log\log n)$ per operation)

2. Think of the setup in the previous question. Assume we only want to support the operations *Split* and *SameInterval*. Design a more efficient data structure for this problem.

Hint: Try to find first a data structure where *Split* takes $O(\log n)$ amortized time and a *SameInterval* query takes $O(1)$ time. Then think of how the reduce the time ot takes to split further. ($O(1)$ should be possible) Use the RAM model. (A solution to the first step would give a substantial fraction of the points)

3. You are given arbitrary intervals on the real line each has a priority assigned to it. Build a data structure (as simple as you can) that supports insertions of new intervals and queries of the form: "given a point find an interval of maximum priority containing it". (Try to find a data structure with logarithmic query time, logarithmic insertion time, and linear space for this problem.)

4. The *lower envelope* of the set of lines S, is the piecewise linear function $f(x) = \min\{ax + b \mid (y = ax + b) \in S\}$. A vertex of the lower envelope is a point $(x, f(x))$ which is an intersection of two lines in S.

We start with an empty set of lines S and perform a sequence $\sigma_1, \ldots, \sigma_n$, where each σ_i is either an insertion of a line to S or a deletion of a line from S. Let E_i be the set of vertices on the lower envelope of the lines in S after we performed $\sigma_1, \ldots, \sigma_i$. Let $E = \cup_{i=1}^{n} E_i$. Prove an upper bound as tight as you can on $|E|$.

5. An $n \times n$ grid is given. We refer to each point on this grid by (i, j), where $1 \leq i \leq n$, and $1 \leq j \leq n$. Fix i, *line* i of the grid is the set of points (i, j), where $1 \leq j \leq n$. For each line i you are given a pair (k, m) of indices where $1 < k \leq m < n$ and a triple of integers (x, y, z). The idea is that the points (i, j) for $1 \leq j < k$ are associated with x, the points (i, j) for $k \leq j \leq m$ are associated with y, and the points (i, j) for $m < j \leq n$ are associated with z.

Design an algorithm to preprocess the triples so that we can efficiently answer queries of the form "Given a grid-point (i, j) find the sum of the integers associated with all points (i', j) for $i' \leq i$. Analyze the query-time, space, and preprocessing time of your data structure. (I see a solution with $O(\log n)$ query time and $O(n \log n)$ space and preprocessing time, but it is possible that you can do even better.)

Bonus:

6. Assume that we use 2-4 trees and that each time a node v is split we have to pay a cost which is equal to the size of v's subtree. Under this assumption, for any m, and n, show a sequence of m operations on a 2-4 tree of size n that takes $\Omega(m \log^k n)$ time for any k.