max planck institut informatik

Algorithms and Data Structures **Summer 2008**
K. Mehlhorn and R. Seidel **Monday, Sept. 29th**
Exercises 36 to 39 **Tuesday, Sept. 30th**
morning and afternoon

Motivation

We learn more about algorithmic strategies in computational geometry. I indicates the relative difficult level of the exercises by numbers in the range from 10 to 30.

1. Ask questions about the material presented in class.
2. (convex hulls by circular sweep, 10) Let S be a set of points in the plane. Choose the lexicographically smallest point in S and make it the origin O. Then all points of S lie in the first and fourth quadrant. Sort the other points by their angle (for a point p different from the origin, its angle is the angle between the positive x-axis and the vector from O to p). Process the points in order.
3. (Pareto-optima). For two points p and q in the plane, we say that q dominates p if $p_x \leq q_x$ and $p_y \leq q_y$. For a point set S, let $P(S)$ be the points in S that are not dominated by any other point in S. The points in $P(S)$ are also called the Pareto-optima of S.

- (5) Draw a set S of 10 points in the plane and indicate the Pareto-optima. Where can points lie that are not dominated by any point in S?
- (10) Show that the set of Pareto-optima has a staircase-like shape, i.e., if p_1 to p_k are the Pareto-optima in order of increasing x-coordinate, then $y(p_1) > y(p_2) > \ldots > y(p_k)$.
- (10) Design a sweep algorithm for computing the Pareto-optima. What is its running time?
- (15) Design a divide-and-conquer algorithm for computing the Pareto-optima. What is its running time?

4. (hierarchical representation of upper convex hulls) The upper convex hull is the part of the convex hull that is visible from $y = +\infty$. Let S be a point set in upper convex position, i.e., S is the vertex set of $UCH(S)$. Let $S = (v_1, \ldots, v_n)$, where the points are sorted according to x-coordinate. A stratification of S is a sequence S_1, S_2, …, S_k such that

   - $S = S_1$ and $S_k = (v_1, v_n)$
   - for $i \geq 2$, S_i is a proper subsequence of S_{i-1} with the property that the first and the last element of S_{i-1} are in S_i and that in any subsequence of four consecutive elements of S_{i-1} at least one and at most three occur in S_i.
   - Each S_i is represented as a linked list of its elements. We also maintain cross links between the S_i's. More precisely, each list item also has an up- and a down-pointer. Let v be an item in S_i. The up-pointer points to the occurence of v in S_{i-1} and the down-pointer points to the occurence of v in S_{i+1}, if any.

   (a) (10) Give an example.

   (b) (10) Draw the pointer structure for your example.

   (c) (15) Prove that $k = O(\log n)$.

   (d) (15) Let p be an arbitrary point in the plane. Show how to determine whether $p \in UCH(S)$? What is the running time of your method?

   Hint: Explore the hierarchy. Solve the problem for $UCH(S_k)$ and work your way backwards.

   (e) (15) Let p be a point with $p \notin UCH(S)$. Show how to construct the tangents from p on $UCH(S)$.

   (f) (20) (incremental hulls) We want to maintain upper hulls under insertions. Assume $p \notin UCH(S)$. Show how to construct a hierarchical representation for $UCH(S \cup \{p\})$. Observe that $S \cup \{p\}$ is not necessarily in convex position. A solution consists of several steps. (1) Determine the tangents. (2) Delete the vertices between the tangents. (3) Add p. (4) Determine the new S_i's.

   What is the complexity of your method? Aim for

   $$O(\log n + \text{number of points that are no longer vertex of the upper hull}) .$$

   What is the amortized cost of an insertion?

5. (weight-balanced search trees) Recall that in search trees every node has either two children or no children. Nodes with no children are called leaves. For a node v, let $w(v)$ be the number of leaves in the subtree rooted at v (= the size of the subtree rooted at v). For an internal node, let $\ell(v)$ and $r(v)$ be the left and right child, respectively.

   Call a tree weight-balanced, if for every internal node v, we have

   $$w(\ell(v)) \geq w(v)/4 \quad \text{and} \quad w(r(v)) \geq w(v)/4 .$$

We call a tree well-balanced if for every node the sizes of the subtrees differ by at most one.

(a) (10) Consider a weight-balanced tree with 10 nodes. What is the range of sizes for the left subtree?

(b) (15) What is the maximal depth of a tree of size n?

(c) Consider the following insertion/deletion procedures. Every node stores its size.

i. In order to add a leaf, replace a leaf by a subtree with two leaves.

ii. In order to remove a leaf, make the other child of the leaf's parent a child of the leaf's grandparent.

iii. Update the sizes of all nodes.

iv. Walk back to the root and determine the highest node (= closest to the root) v, if any, that is out of balance, i.e., for which one of the children of v has less than one-fourth the weight of v. Replace the subtree rooted at v by a well-balanced tree.

(20 – 25) How many nodes are affected in step (iii)? How much does it cost to update the weights? Assume that step (iv) takes time $O(w(v))$. What is the amortized cost of n insertions and deletions starting with an empty tree.

Hint: recall the analysis of dynamic arrays.

6. (dynamic Pareto-optima) We want to maintain the Pareto-optima of a point set S under insertions and deletions. We store the set S in the leaves of a weight-balanced tree and mimick the divide and conquer algorithm. Consider a node v and suppose we have computed the staircases for the two children v_ℓ and v_r. The staircase at v consists of an initial part of the staircase at v_ℓ plus the staircase at v_r. *We store the final segment of the staircase of v_ℓ that does not make it to v at v_ℓ. We also store a pointer into the merged staircase to indicate the position where the two staircases were glued together.*

Globally, the effect is as follows: We mimick the divide-and-conquer algorithm, but we do not throw information away, when parts of a staircase are removed. Rather, we keep the part at the highest node of the tree where it was still relevant.

(a) (20) How long does it take to build a tree for n points? You may assume that the points are sorted by x-coordinate.

(b) (20) Consider a path from the root to a leaf. Show how you can undo the merge steps along this path. Can you do in time $O(1)$ per node?

(c) (25) Insert a new point. This corresponds to the insertion of a leaf. Undo the merges (as in the preceding item) along the path to the new leaf, add the leaf, and then work your way back up to the root. How much time does the remerging take?

(d) (25) Delete a point. Proceed as in the preceeding item.

(e) (30) Can you reduce the time required for items (c) and (d) to $O(\log n)$ per node on the path.?

(f) (3)) Maintain the underlying tree as a weight-balanced tree. Use item (a) to estimate the time required for rebuilding a subtree.

7. (convex hulls by divide-and-conquer, 15) Design a divide-and-conquer algorithm for the upper convex hull problem (= the part of the hull that is visible from $y = +\infty$) for points sets in the plane. Assume for simplicity that no two points have the same x-coordinate.

   Hint: Sort the points lexicographically, compute the hull of the first and the second half of the points recursively, and then merge the hulls by constructing their common tangents.

   How much time can you allow yourself for constructing the tangents if you aim for an $O(n \log n)$ algorithm? Show how to find the tangents.

   Hint: For the merge step, we have two upper convex hulls P_L and P_R. We also know a vertical line V that separates P_L from P_R. Let u be a vertex of P_L and v be a vertex of P_R and let $\ell(u,v)$ be the directed line from u to v. $\ell(u,v)$ either is tangent to P_L or intersects the boundary of P_L twice. Similarly, for P_R. Thus there are 4 possible cases. Discuss them and show that in each case, some parts of at least one of the polygons can be discarded from further consideration.

Have fun with the solutions.