CS 473: Fundamental Algorithms, Fall 2014

Discussion 3

September 16 / 17, 2014

3.1 RECURRENCES

Solve the following recurrences.

(A) $T(n) = 5T(n/4) + n$ and $T(n) = 1$ for $1 \leq n < 4$.
(B) $T(n) = 2T(n/2) + n \log n$
(C) $T(n) = 2T(n/2) + 3T(n/3) + n^2$

3.2 TREE TRAVERSAL.

Let T be a rooted binary tree on n nodes. The nodes have unique labels from 1 to n.

(A) Given the preorder and postorder node sequences for T, give a recursive algorithm to reconstruct a tree that satisfies the preorder and postorder sequences. Is this reconstruction unique?
(B) Given the preorder and inorder node sequences for T, give a recursive algorithm to reconstruct a tree that satisfies the preorder and inorder sequences. Is this reconstruction unique?

3.3 DIVIDE AND CONQUER.

Let $p = (x, y)$ and $p' = (x', y')$ be two points in the Euclidean plane given by their coordinates. We say that p dominates p' if and only if $x > x'$ *and* $y > y'$. Given a set of n points $P = \{p_1, \ldots, p_n\}$, a point $p_i \in P$ is undominated in P if there is no other point $p_j \in P$ such that p_j dominates p_i. Describe an algorithm that given P outputs all the undominated points in P; see figure. Your algorithm should run in time asymptotically faster than $O(n^2)$

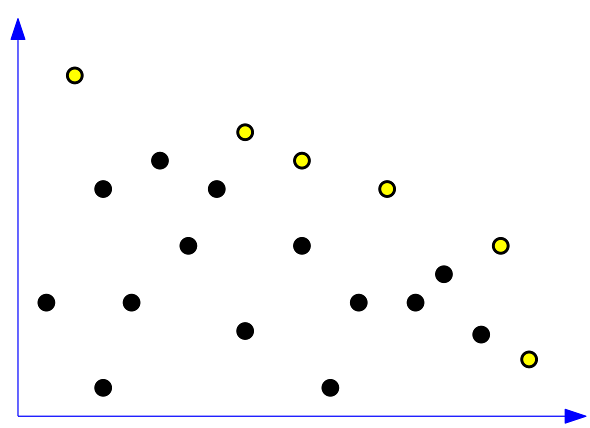

Figure 1: The undominated points are shown as unfilled circles.

3.4 Merging arrays.

Suppose you are given k sorted arrays $A_1, A_2, \dots, A_k$ where each array contains n elements. The goal is to merge all the arrays into a single sorted array A of kn elements. Given two sorted arrays of size x and y respectively, you know that they can be merged into a single sorted array in $O(x + y)$ time.

(a) Suppose you use the following algorithm for merging the k arrays. Merge A_1 and A_2. Merge the resulting array with A_3 and the result with A_4 and so on. What is the running time of this algorithm as a function of k and n?

(b) Give a more efficient algorithm.

(c) Consider the following modification to the merge sort algorithm. Instead of splitting the input array into 2 subarrays, recursively sorting each and merging the 2 sorted subarrays, we will split the input array into k subarrays, recursively sort each (using the modified algorithm), and merge the k sorted subarrays. How does the running time of the modified algorithm compare to that of the original algorithm?