

Selection, Heaps, Hashing

October 20, 2009

Homework 3

Due Date: Tuesday, 27 October 2009 by end of lecture

Problem 3-1. 15pts

CLRS 5.2-1, page 122 (page 98 in 2nd edition).

Problem 3-2. 15pts

In the deterministic linear time selection algorithm we were dividing the array into groups of size 5. What is special about this number? In particular, what will happen if we choose to divide into groups of 3? groups of 7? Will the algorithm be correct in each one of these cases? Will it still run in linear time? Any advantages of using 5 as opposed to some other number?

Problem 3-3. 15pts

Give an $O(n \log k)$ -time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists.

Problem 3-4. 15pts

Suppose we have an arbitrary data type that supports constant-time comparisons and copies but no other operations. For example, we could compare x to y and swap them if $x < y$ in constant time. Can we implement a data structure over this data type such that INSERT and EXTRACT-MAX both run in $o(\log n)$ time? If so, provide main ideas behind the implementation. If not, prove it is impossible.

Problem 3-5. 20pts

CLRS 11-1, page 282 (page 249 in 2nd edition).

Problem 3-6. 20pts

CLRS 11-2, page 283 (page 250 in 2nd edition).

Problem 3-7. Extra credit: Direct approach to finding the “second smallest element” in an array is to first find the minimum, delete it, and then find the minimum in the remaining set. This results in $(2n - \text{constant})$ comparisons. Show that we can do this using only $n + O(\log n)$ comparisons in the worst case. More precisely, show that $n + \lceil \log_2 n \rceil - 2$ comparisons are sufficient.