

Homework 2

Assigned: September 23

Due: October 5

Please respect the following guidelines for writing pseudocode:

1. C instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed $0 \dots n - 1$ or $1 \dots n$ (Problem 1 forces you to use $1 \dots n$).

Problem 1 Let T be a heap storing n keys. Assume for this problem that the heap is stored in an array A starting from 1 - that is, $A[1]$ is the biggest key in the heap. Give an efficient algorithm for reporting all the keys in T that are greater than or equal to a given query key x (which is not necessarily in T). Note that the keys do not need to be reported (printed) in sorted order. Analyse the running time. An $O(k)$ algorithm is needed for full grade, where k is the number of elements printed. That is, the algorithm should do a constant number of elementary operations per printed element.

Problem 2 Professor Amongus claims that a (2,3) tree storing a set of items will always have the same structure, regardless of the order in which the items are inserted. Show that Professor Amongus is wrong, by giving two orders of the same set of items giving distinct trees.

Problem 3 Consider the following sequence of keys:

(5,16,22,45,2,10,18,30,50,12,1)

Consider the insertion of items with this set of keys, in the order given, into an initially empty (2,3) tree T . Draw T after each insertion.

Problem 4 Suppose we are given two sorted arrays S and T , each with n items. Describe an $O(\log n)$ -time algorithm for finding the k th smallest key in the union of the keys from S and T (assuming no duplicates). Present pseudocode, argue correctness and analyze the running time.

Problem 5 Suppose we are given two n -element sorted sequences A and B that should not be viewed as sets (that is, A and B may contain duplicate entries). Describe an $O(n)$ -time method for computing a sequence representing the set $A \cup B$ (with no duplicates). Present pseudocode and analyze the running time.