

CSCI 303 Homework 3

Problem 1 (9-1):

Given a set A of n numbers, we wish to find the k largest in sorted order using a comparison-based algorithm. Find the algorithm that implements each of the following methods with the best asymptotic worst-case running time, and analyze the running time of the algorithms in terms of n and k .

- a. Sort the numbers, and list the k largest.
- b. Build a max-priority queue from the numbers, and call EXTRACT-MAX k times.
- c. Use an order-statistic algorithm to find the i th largest number, partition around that number, and sort the k largest numbers.

Problem 2 (9.3-5):

Suppose that you have a “black-box” worst-case linear-time median subroutine. Give a simple, linear-time algorithm that solves the selection problem for an arbitrary order statistic.

Problem 3 (9.3-8):

Let $X[1, \dots, n]$ and $Y[1, \dots, n]$ be two arrays, each containing n numbers already in sorted order. Give an $O(\lg n)$ -time algorithm to find the median of all $2n$ elements in arrays X and Y .

Problem 4 (8.1-1):

What is the smallest possible depth of a leaf in a decision tree for a comparison sort?

Problem 5 (Derived from 8.1-4):

You are given a sequence of n elements to sort and a number k such that k divides n . The input sequence consists of n/k subsequences, each containing k elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence and larger than the elements in the preceding subsequence. Thus, all that is needed to sort the whole sequence of length n is to sort the k elements in each of the n/k subsequences. Show an $\Omega(n \lg k)$ lower bound on the number of comparisons needed to solve this variant of the sorting problem using a comparison-type sorting algorithm. (*Hint:* It is not rigorous to simply combine the lower bounds for the individual subsequences.)