Algorithms I

Tutorial 3

August 26, 2016

Problem 1

CLRS 7.5-3

Problem 2

You are given a min heap containing n integers. You are asked to find the number of elements in the heap which are less than m. Your algorithm should run in $O(k)$ time, where k is the number of elements in the heap which are less than m.

Problem 3

You are given an integer n. You are also given at most n (x_i, v_i) pairs such that all x_i are distinct and $1 \leq x_i \leq n$. You need to design an ADT which supports the following operations:

- $getMax()$: Get the maximum value v among all the values present. Time : $O(1)$
- $decreaseKey(x, v)$: Decrease the value of key x to v. You can assume v is always less than the current value of x. Time : $O(\log n)$

You are allowed to pre-process the data before the operations.

Problem 4

You are given two arrays A and B of integers of sizes m and n. Your task is to check whether A and B are equal as *sets*. The arrays A and B need not be sorted, and may contain repeated occurrences of the same values. When we treat them as sets, all repetitions should be discarded (only one occurrence counts). For example, if $A = [5, 1, 2, 5, 1, 8, 1, 3]$ and $B = [2, 1, 8, 2, 5, 3]$, the answer is *Yes*, since both the arrays are equal to $\{1, 2, 3, 5, 8\}$ as sets. Design an algorithm to solve this problem in expected $O(m + n)$ time. Assume you have a hash table that supports *search* and *insert* in expected $O(1)$ time.

Problem 5

Here is an array which has just been partitioned by the first step of Quicksort: $[3, 0, 2, 4, 5, 8, 7, 6, 9]$. Which of these elements could have been the pivot?

Problem 6

How can you use the partitioning idea of quicksort to give an algorithm that finds the median element of an array of n integers?

Problem 7

You are given an array containing n elements. Your task is to find the *majority element*. A *majority element* is the element that occurs more than $\frac{n}{2}$ times in the array. Your algorithm should run in $O(n \log n)$ time. You are not allowed to use sorting. Can you do this in $O(n)$ time?

Problem 8

You are given an array containing n integers. You need to find the number of pairs (i, j) such that $i < j$ and $a[i] > a[j]$. Your algorithm should run in $O(n \log n)$ time.

Problem 9

Given two sets A_1 and A_2 (each of size n), and a number x, describe an $O(n \log n)$ time algorithm for finding whether there exists a pair of elements, one from A_1 and one from A_2, that add up to x.