PhD Comprehensive Exam: Math and Algorithms

Spring 1999

The exam includes eight problems; the length of the exam is three hours.
Problem 1
Consider the following sorting algorithm:

\[
\text{SLOW-SORT}(A, i, j) \\
\text{if } A[i] > A[j] \\
\quad \text{then exchange } A[i] \leftrightarrow A[j] \\
\text{if } i + 1 \geq j \\
\quad \text{then return} \\
\quad k \leftarrow \lfloor (j - i + 1)/3 \rfloor \\
\quad \text{SLOW-SORT}(A, i, j - k) \quad \triangleright \text{First two-thirds.} \\
\quad \text{SLOW-SORT}(A, i + k, j) \quad \triangleright \text{Last two-thirds.} \\
\quad \text{SLOW-SORT}(A, i, j - k) \quad \triangleright \text{First two-thirds again.}
\]

Argue that SLOW-SORT(A, 1, n) correctly sorts the input array A[1..n], and give a tight asymptotic bound for its running time.
Problem 2
Design a nonrecursive algorithm that finds the $k$th smallest element in an array $A[1..n]$, where $k \leq 20$. The running time of your algorithm must be $O(n)$. You may use an extra array of size 20.
Problem 3

A d-ary heap is like a binary heap, but instead of 2 children, nodes have d children.

(a) What is the exact height of a d-ary heap of n elements in terms of n and d?

(b) Suppose that you represent a d-ary heap by an array $A[1..n]$. Give an expression for finding the parent of an element with index $i$, where $1 \leq i \leq n$. 
Problem 4

Prove or disprove the following statement:

If \( \log(f(n)) = \Theta(\log(g(n))) \), then \( f(n) = \Theta(g(n)) \).
Problem 5

Suppose that you are using a programming language that allows only integer numbers and supports four operations on them: addition, subtraction, multiplication, and integer division; the running time of each operation is constant, that is, $\Theta(1)$. The result of the integer division of $n$ over $m$ is $\lfloor n/m \rfloor$; for example, $\lfloor 10/3 \rfloor = 3$. The language does not allow fractional numbers, and does not have operations for logarithms and exponentiation.

Write an efficient algorithm $\text{POWER}(n, m)$ that computes $n^m$, where $n$ and $m$ are positive integers, and give the asymptotic time complexity of your algorithm. Note that computing $n^m$ in $\Theta(m)$ time is too slow, and you need to design a faster algorithm.
Problem 6
Design at least three approximation algorithms for the bin-packing problem, and determine asymptotic upper bounds for the time complexity of your algorithms.
**Problem 7**

Suppose that $G$ is an undirected connected graph, with positive weights of edges, and $s$ is a vertex of $G$. Suppose further that $G'$ is obtained from $G$ by adding some constant to the weight of every edge (this constant is the same for all edges). Prove or disprove the following statements:

(a) Every minimum spanning tree of $G$ is also a shortest-paths tree of $G$ for the source $s$.

(b) Every minimum spanning tree of $G$ is also a minimum spanning tree of $G'$. 

Problem 8

Suppose that $S$ is a finite set of integer numbers. In the *Subset-Sum Problem*, we need to determine whether there is a subset $S'$ of $S$ whose elements sum to a given number $n$. That is, we have to construct an algorithm that inputs $S$ and $n$, and returns TRUE if there exists $S' \subseteq S$ such that

$$\sum_{s \in S'} s = n.$$  

In the *Set-Partition Problem*, we are determining whether $S$ can be split into two subsets that have the same sum of elements. That is, we need an algorithm that inputs $S$, and returns TRUE if there exists $S' \subseteq S$ such that

$$\sum_{s \in S'} s = \sum_{s \in (S - S')} s.$$  

The Subset-Sum Problem is known to be NP-complete. Use this fact to prove that the Set-Partition Problem is also NP-complete.