Principles of Algorithmic Techniques - Home Assignment 1

Jaan Tollander de Balsch

2018-09-18

Question 1

Solve the recurrence

$$T(n) = T(3n/4) + 1, \quad T(1) = 1$$

by using iterative method. You may assume that $n = 4^k$ for some integer k.

Question 2

Let $T(n)$ be a functions such that

$$T(n) \leq T(n/5) + T(3n/10) + O(n)$$

and $T(j) = 1$ for all $j < 10$. Prove that $T(n) = O(n)$.

Question 3

The Fibonacci numbers $F_0, F_1, F_2, ...$ are defined by the rule

$$F_0 = 0, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}, n \geq 2.$$

In this problem we will confirm that this sequence grows exponentially fast and obtain some bounds on its growth.

1) Use induction to prove that $F_n \geq 2^{0.5n}$ for $n \geq 6$.
2) Find a constant $c < 1$ such that $F_n \leq 2^{cn}$ for all positive integer n. Show that your answer is correct.
3) What is the largest c you can find for which $F_n = \Omega(2^{cn})$

Question 4

Devise a divide-and-conquer (recursive) algorithm for adding up all numbers in the array $A[1...n]$. Your algorithm should divide the array A into two equal subarrays.

1) Describe your algorithm in a clear pseudocode.
2) Analyze the runnning time of your algorithm. Prove your answer.

Question 5

Given an array $A[1...n]$, we say that entry $A[i]$ is *locally minimum* if

$$A[i] = \min(A[i-1], A[i], A[i+1]);$$

for $i = 1$, it is locally minimum if $A[i] < A[i+1]$ and for $i = n$, when $A[i] < A[i-1]$.

Describe an algorithm that finds a local minimum in an array A in time $O(\log n)$. Explain your algorithm either in clear English or a clear pseudocode.

Question 6

There are n gold coins, one of which is a fake. All gold coins have the same weight, and the fake coin has different weight. You are allowed to use a balance whether two subsets of coins have equal weights. Describe a strategy that uses $O(\log n)$ measurements of balance and find a fake coin.

Question 7

Given two sorted arrays $A[1...n]$ and $B[1...n]$, describe an algorithm that find the k-th smallest element in $A \cup B$ in $O(\log n)$ time.

Question 8

Run the strongly connected components algorithm on the following directed graphs, starting with Node A. Whenever there is a choice of vertices to explore, always pick one that is alphabetically least.

Answer the following questions:

a) In which order are the strongly connected components found?
b) Which are the source and sink SCCs?
c) Draw the meta-graph.

Question 9

A **number maze** is an $n \times n$ grid of positive integers. A token starts in the upper left corner; your goal is to move the token to the lower-right corner. On each turn, you are allowed to move the token up, down, left, or right; the distance you may move the token is determined by the number on its current square . You are never allowed to move the token off the edge of the board.

Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution.

Question 10

Let us revisit the stable matching problem. Let U be the set of universities and S be the set of students where $|U| = |S|$.

Each student s has a preference relation $\prec_s$ where $u \prec_s u'$ is used to denote the fact that student s prefers to go to university u' to u. Similarly for the preference relation $\prec_u$ for each university u.

In class, we dicussed the face where $\prec_s$ and $\prec_u$ give complete ranking over all universities and students respectively, i.e. $\prec_s$ given a ranking over U and $\prec_u$ given a ranking over all S.

Now, we are interested in the following setting: Some students may not be interested in going to some university, i.e. Nidia (hypothetical person) may only be interested in a certain subset of universities $U_{Nidia} \subseteq U$ and for other universities not in her interest, she may prefer to stay home rather that studying there. The same does for universities: Each university u may be only interested in some subset $S_u \subseteq S$ of students, and prefer to not admit any other student.

1) First, explain what it means to be a stable assignment in this setting.
2) Show that in this setting, there always exists a stable assignment and it can be computed efficiently. Prove correctness and analyze the efficientcy of your algorithm.