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.

Using iteration on the recurrence relation we get

$$\begin{aligned} T(1) &= 1 \\ T(4/3) &= T(1) + 1 = 2 \\ T((4/3)^2) &= T(4/3) + 1 = 3 \\ &\vdots \\ T((4/3)^k) &= k + 1 \end{aligned}$$

By substituting $x = (4/3)^k$ which is also equivalent to $k = \log_{4/3} x$ we get

$$T(x) = \log_{4/3} x + 1$$

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 $F_n = \Omega(2^{cn})$

Lets prove that $F_n \geq 2^{0.5n}$ for $n \geq 6$ using induction.

1) Base case $n = 6$:
$$F_6 = 8 \geq 2^{0.5 \cdot 6}$$
2) Step case n:
$$\begin{aligned} F_n &= F_{n-1} + F_{n-2} \\ &\geq 2^{0.5 \cdot (n-1)} + 2^{0.5 \cdot (n-2)} \\ &= 2^{0.5n} \cdot (2^{0.5 \cdot (-1)} + 2^{0.5 \cdot (-2)}) \\ &\geq 2^{0.5n} \end{aligned}$$
3) Step case $n + 1$:
$$\begin{aligned} F_{n+1} &= F_n + F_{n-1} \\ &\geq 2^{0.5n} + 2^{0.5 \cdot (n-1)} \\ &= 2^{0.5n} \cdot (1 + 2^{0.5 \cdot (-1)}) \\ &\geq 2^{0.5 \cdot n} \cdot 2^{0.5} \\ &= 2^{0.5 \cdot (n+1)} \end{aligned}$$

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.

The pseudocode implementation (in Julia style) of the algorithm.

```
function sum(A)
  if lenght(A) == 1
    # If the array A has only one element return the element.
    return A[0]
  else
    # Otherwise divide the array into equal sized subarrays and
    # then add the sums of the subarrays together.
    B, C = divide(A)
    return sum(B) + sum(C)
  end
end
```

In the running time analysis we assume that operations `length`, `[]` (indexing) and `divide` on arrays are constant time. We also assume that addition `+` of two numbers is constant time. The recurrence relation therefore is of form

$$T(n) = 2T(\lceil n/2 \rceil) + O(1).$$

By using the *master theorem*, the runningtime of the algorithm is

$$T(n) = O(n).$$

As we can see, the divide and conquer algorithm for the summation is no better than direct summation term by term.

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.

This algorithms assumes that $n \geq 3$. In the case $n \in \{1, 2\}$ it is impossible to determine the fake coin using a balance. The recursion starts with $n_1 = n$ coins.

1) Split the stack of n_i coins into 3 *stacks* of equal size. The stacks will be of size $\lfloor n_i/3 \rfloor$. If the number of coins is not divisible by 3, there will be a *leftover stack* of 1 or 2 coins ($\mod (n, 3)$).
2) Lets denote the weights of the three stacks by w_1, w_2 and w_3. In order to figure out which stack has the fake coin, we need to compare each of the stack with each other using the balance. There are four different scenarios:
   I) If $w_1 \neq w_2 \wedge w_1 \neq w_3 \wedge w_2 = w_3$ then the fake coin is in the *stack 1*.
   II) If $w_1 \neq w_2 \wedge w_1 = w_3 \wedge w_2 \neq w_3$ then the fake coin is in the *stack 2*.
   III) If $w_1 = w_2 \wedge w_1 \neq w_3 \wedge w_2 \neq w_3$ then the fake coin is in the *stack 3*.
   IV) If $w_1 = w_2 \wedge w_1 = w_3 \wedge w_2 = w_3$ then the fake coin is in the *leftover stack*.
3) Now we have scenarios:
   I) If fhe fake coin is in the *stack 1, 2 or 3*.
      a) If the size of the stack is 1 then this is the fake coin.
      b) Otherwise repeat the procedure from step 1 using the stack that has the fake coin.
   II) Otherwise the fake coin is in the *leftover stack*.
      a) If there is 1 coin in the leftover stack, this is the fake coin.
      b) If there is 2 coins in the leftover stack, take one coin from any of the stacks 1, 2 or 3, and compare the leftover coins against it. The leftover coin that has unequal weight compared to this coin is the 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.