Tutorial 2 Exercise Solutions

COMP526: Efficient Algorithms

14–15 October, 2024

Exercise 1. Consider the sequence of numbers $T(n)$ defined recursively by

$$T(n) = \begin{cases} 3 & \text{if } n = 0; \\ T(n-1) + 4 & \text{if } n \geq 1. \end{cases}$$

(a) Compute the first 6 elements of $T(n)$, i.e., $T(0)$, $T(1)$, $T(2)$, $T(3)$, $T(4)$, and $T(5)$.

(b) Make an educated guess about the general pattern that this sequence follows. Write this guess as a *closed form* for $T(n)$, i.e., a formula for $T(n)$ without recursive reference to T.

(c) Now formally prove the correctness of your guess using mathematical induction.

Solution. (a) $T(0) = 3$, $T(1) = 7$, $T(2) = 11$, $T(3) = 15$, $T(4) = 19$, $T(5) = 23$.

(b) Here is a general approach to solve this type of problem. The idea is to insert the recursive definition while keeping n as a variable. Assume that n is large enough so that we can apply the second part of the definition of $T(n)$, namely $T(n) = T(n-1) + 4$. Iterating this process, we obtain

$$\begin{aligned} T(n) &= T(n-1) + 4 \\ &= (T(n-2) + 4) + 4 \\ &= T(n-2) + 2 \cdot 4 \\ &= (T(n-3) + 4) + 2 \cdot 4 \\ &= T(n-3) + 3 \cdot 4. \end{aligned}$$

After $i \leq n$ iterations, we thus obtain $T(n) = T(n-i) + i \cdot 4$. For $i = n$, this is $T(0) + 4n = 3 + 4n$ (from the first part of the definition).

So, our educated guess is $\forall n \in \mathbf{N}_0 : T(n) = 4n + 3$.

(c) Now, we formally prove the correctness of this "guess" by induction. In the notation of the lecture notes, we have $P(n) \equiv T(n) = 4n + 3$.

Base case: We have to check $P(0)$, i.e., $T(0) = 3$. By the first part of the definition of T, this is indeed the case.

Inductive step: Now, we have to prove $\forall n \in \mathbf{N} : P(n) \implies P(n+1)$.

Let $n \in \mathbf{N}$ be arbitrary, but fixed and assume $P(n)$ is true. ($P(n)$ is called the *inductive hypothesis.*) We have to prove $P(n+1)$, i.e., $T(n+1) = 4(n+1)+3$.

By the second part of the definition of T, $T(n+1) = T(n)+4$, and using the inductive hypothesis, this is $T(n+1) = (4n+3)+4 = 4(n+1)+3$, which is what we had to prove.

Now the claim follows for all n by the induction principle.

□

Exercise 2. Recall that given positive integers n and k, the ***modulo operation*** $n \bmod k$ computes the remainder when n is divided by k. That is, $r = n \bmod k$ if and only if $n = q \cdot k + r$ for some integer q and $0 \le r < k$. Consider the following MOD procedure that computes $n \bmod k$.

procedure MOD(n, k)
    $t \leftarrow n$
    while $t \ge k$ **do**
        $t \leftarrow t - k$
    end while
    return t
end procedure

(a) Argue that MOD(n, k) correctly computes $n \bmod k$. (Hint: what is a loop invariant maintained after each iteration of the loop?)

(b) Express the running time of this procedure as a function of n and k using big-O notation.

Solution. (a) Consider the following loop invariant: after each iteration of the loop, $t \bmod k = n \bmod k$. We argue that this invariant holds by induction:

Base case: Before the first iteration, we have $t = n$, so the invariant is trivially true.

Inductive step: Assume that the invariant holds at the beginning of the ith iteration, i.e., $t_i \bmod k = n \bmod k$. After the ith iteration, we have $t_{i+1} = t_i - k$. By the inductive hypothesis, we thus have $t_{i+1} \bmod k = (t_i - k) \bmod k = (n \bmod k - k) \bmod k = n \bmod k$. Thus the invariant is established for the $(i+1)$st iteration.

When the loop terminates, we have $t < k$, so $t \bmod k = t$. Therefore, by the invariant, we have $n \bmod k = t$. Thus the procedure correctly computes $n \bmod k$.

(b) To analyze the running time of MOD We will count the number of executed (pseudocode) instructions. First of all, outside of the loop, there are only 2 operations; they are executed exactly once. Each iteration of the loop adds 2 more instruction (the body and checking the condition).

Consider the value of t during the execution of the loop. T is originally equal to n, and the loop terminates when t stores a value between 0 and $k-1$. Each iteration reduces t by k, so we will have exactly $\left\lfloor \frac{n}{k} \right\rfloor$ iterations.

The overall number of instructions for $\textsc{Mod}(n,k)$ is therefore $2+2\left\lfloor \frac{n}{k} \right\rfloor = \Theta\left(\frac{n}{k}\right)$.

$\square$