UNIVERSITY OF LIVERPOOL

Department of Computer Science
Sebastian Wild

Date: 2020-02-19
Version: 2020-02-19 16:10

Tutorial 2 for COMP 526 – Applied Algorithmics, Winter 2020

—including solutions—

It is highly recommended that you first try to solve the problems on your own before consulting the sample solutions provided below.

Problem 1 (Decreasing function and amortization method)

Consider again the Mod function from last time:

```
1  procedure Mod(n, k)
2  // Input: positive integers n, k.
3  // Output: value of n mod k.
4  t := n
5  while t ≥ k
6      t := (t − k)
7  end while
8  return t
```

a) Apply the *decreasing potential method* to prove that the function $Mod(n, k)$ always terminates.

b) Try to establish the time complexity of this procedure.

Solutions for Problem 1 (Decreasing function and amortization method)

a) Recall that a *potential* Φ must satisfy two conditions: it must be a natural number, and during each iteration of the loop, the value of the function must be reduced.

In our case, we can use $\Phi = t$; we now have to check the two conditions.

- $\Phi \in \mathbb{N}_0$. t is initially n and $n \in \mathbb{N}$ by precondition of Mod. Since the only change to t is the assignment $t := t - k$, t will remain integral. Moreover, we only execute the update when t was at least k, so we will always have $t \geq 0$. (Recall also our loop invariant from last week.)
- Φ strictly decreases. In every iteration of the loop, we decrement t by k, and $k \geq 1$ by the precondition of Mod. Hence Φ strictly decreases in each iteration.

Together, this establishes that the loop can only be executed a finite number of times.

b) 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).

We can now use our potential to argue about the number of iterations of the loop. Φ is originally equal to n and ultimately between 0 and $k-1$ (recall again our loop invariant); moreover, each iteration reduces Φ by k, so we will have exactly $\lfloor \frac{n}{k} \rfloor$ iterations.

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

Although the example is very simple, it demonstrates that invariants and potentials are not only useful mathematical tools for formal verification, but also good guiding principles to help us understand algorithms and data structures.

Problem 2 (Telescoping recurrence and mathematical induction)

Given a complexity function $T(n)$ recursively defined as

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

Find a *closed form* (without recursive reference) for $T(n)$ by iterating (inserting the recursive definition) until you can make an educated guess.

Then prove the correctness of your guess by mathematical induction.

Solutions for Problem 2 (Telescoping recurrence and mathematical induction)

We first use the *telescoping mechanism* to establish a closed form of the complexity function $T(n)$. 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 $\forall n \in \mathbb{N}_0 : T(n) = 4n + 3$.

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

- Induction Basis
  We have to check $A(0) \equiv T(0) = 3$. By the first part of the definition of T, this is indeed the case.
- Inductive Step
  Now, we have to prove $\forall n \in \mathbb{N}_0 : A(n) \Rightarrow A(n+1)$.

  Let $n \in \mathbb{N}_0$ be arbitrary, but fixed and assume $A(n)$ is true. ($A(n)$ is called the *inductive hypothesis*.) We have to prove $A(n+1) \equiv T(n+1) = 4(n+1) + 3$.

  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.