Applied Algorithmics COMP526 – tutorial 2

L.A. Gąsieniec and D. Cartwright

1 Questions

1.1 Decreasing function and amortisation method

Apply the *decreasing function method*, see lecture notes from week 1, to prove that the function $Mod(n, k)$ (introduced last week, see also below) stops eventually. Try also to establish the time complexity of this procedure.

function $Mod(n, k$: *integer*): *integer*;
Input: positive integers n, k.
Output: value of $n \bmod k$.
  temp $\leftarrow n$;
  while $temp \geq k$ **do**
    temp $\leftarrow$ (*temp* $-k$);
  end-while;
  return *temp*;
end-function.

1.2 Telescoping recurrence and mathematical induction

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

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

Use telescoping method to find a *closed form* (without recursive reference) for the complexity function $T(n)$. Further prove by mathematical induction that the obtained closed form is correct.

1.3 Bottom-up heap construction

For this extra material consult COMP526 web pages.

2 Solutions

2.1 Decreasing function and amortisation method

Recall that a *decreasing function* $f()$ must satisfy two conditions. During each consecutive iteration of the loop the value of the function must be reduced, however it should never reach value 0. In our case we can take a (linear) function $f(temp) = temp + 1$, where we add 1 to ensure that the value of the function remains positive. Note that the initial *potential* of $f(temp)$ is $n + 1$, since at the start $temp$ is set to n. Further, during each consecutive iteration of the loop this potential is reduced exactly by k (this is due to the instruction *temp* $\leftarrow$ (*temp* $-k$);). Thus clearly $f(temp)$ is a decreasing function. Finally, $f(temp)$ must stay above 0 because the value $temp$ is always ≥ 0 due to the condition (**while** $temp \geq k$ **do**) in the loop. I.e., as soon as $temp < k$, further reduction of $temp$ is not possible. Thus the value (potential) of $f(temp)$ is always ≥ 1.

Note that the potential of $f(temp)$ is originally equal to n. During each iteration of the loop we perform a *constant* number of basic operations. The number of iterations can be easily bounded (from above) by $\frac{n}{k}$, since during each iteration the potential is always reduced by k (and it never reaches 0). Thus the time complexity of the procedure $Mod(n, k)$ is $O(\frac{n}{k})$.

2.2 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, i.e., we can apply the second part of the recursive definition of $T(n)$, namely $T(n) = T(n-1) + 4$. Note also that since n is large we can iterate this process a number (say i) times. We obtain:

$$T(n) = T(n-1) + 4 = (T(n-2) + 4) + 4 = T(n-2) + 2 \cdot 4 =$$

$$(T(n-3) + 4) + 2 \cdot 4 = T(n-3) + 3 \cdot 4 = ... = T(n-i) + i \cdot 4,$$

for all integer $i \leq n$. Also note that for $i = n$ we get $T(n) = T(0) + n \cdot 4$, where $T(0) = 3$ from the first part of the definition. Thus we conclude with the closed form $T(n) = 4n + 3$.

We can now check whether this form is correct using mathematical induction.

Basis Case Take $n = 0$. From the definition (1st part) we get $T(0) = 3$. Also from the closed form $T(0) = 4 \cdot 0 + 3 = 3$. Thus the basis case is done.

Inductive Step Assume now that the closed form gives the correct value of function $T(i)$, for all $0 \leq i \leq n-1$, i.e., in particular $T(n-1) = 4 \cdot (n-1) + 3$. Now since we know (from the definition) that $T(n) = T(n-1) + 4$ and (from the inductive assumption) that $T(n-1) = 4 \cdot (n-1) + 3$ we conclude that $T(n) = 4 \cdot (n-1) + 3 + 4 = 4n + 3$, which completes the proof.