Week 2 Preparation

(Solutions)

Useful advice: The following solutions pertain to the preparation problems. You are strongly advised to attempt the problems thoroughly before looking at these solutions. Simply reading the solutions without thinking about the problems will rob you of the practice required to be able to solve complicated problems on your own. You will perform poorly on the exam if you simply attempt to memorise solutions to these problems. Thinking about a problem, even if you do not solve it will greatly increase your understanding of the underlying concepts.

Problem 1. Given the following pseudocode, derive the recurrence relation that represents its time complexity. Define the base case and recurrence step.

```
1: function FACTORIAL(n)
2:     if n = 0 then return 1
3:     else return n × FACTORIAL(n − 1)
```

Solution

The following recurrence relation represents the function's time complexity

$$T(n) = \begin{cases} T(n-1)+c, & \text{if } n > 0, \\ b, & \text{if } n = 0. \end{cases}$$

where b and c are constants.

Problem 2. Given the following pseudocode, derive the recurrence relation that represents its time complexity. Define the base case and recurrence step.

```
1: function IS_POWER_OF_TWO(n)
2:     if n = 1 then return true
3:     else if n < 1 then return false
4:     else return IS_POWER_OF_TWO(n/2)
```

Solution

The following recurrence relation represents the function's time complexity

$$T(n) = \begin{cases} T(n/2)+c, & \text{if } n > 1, \\ b, & \text{if } n \leq 1. \end{cases}$$

where b and c are constants.

Problem 3. Given the following pseudocode, derive the recurrence relation that represents its time complexity. Define the base case and recurrence step.

```
1: function FIND_MIN(arr[1..n])
2:     if n = 1 then return arr[1]
3:     else
4:         potentialMin = FIND_MIN(arr[2..n])
5:         if arr[1] > potentialMin then
6:             return potentialMin
7:         else return arr[1]
```

Solution

The following recurrence relation represents the function's time complexity

$$T(n) = \begin{cases} T(n-1)+c, & \text{if } n > 1, \\ b, & \text{if } n = 1, \end{cases}$$

where n is the number of elements in *arr*, and b and c are constants.

Problem 4. Find a closed form solution for the following recurrence relation:

$$T(n) = \begin{cases} T(n-1)+c, & \text{if } n > 0, \\ b, & \text{if } n = 0. \end{cases}$$

[Hint: Use telescoping in order to express $T(n)$ in terms of $T(n-2)$ instead of $T(n-1)$, then in terms of $T(n-3)$ instead of $T(n-2)$, and so on until you can figure out the general pattern. Then use the base case to obtain a formula for $T(n)$ that only depends on n, b and c, but not on $T(\cdot)$.]

Solution

To find a closed form for T, we will use the method of telescoping, i.e. we look for a pattern as we substitute the recurrence relation into itself. For $n > 0$ we have

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

Continuing the pattern, we see that the general form for $T(n)$ seems to be

$$T(n) = \begin{cases} T(n-k)+kc, & \text{if } n > 0, \text{for all } 0 \le k \le n, \\ b, & \text{if } n = 0. \end{cases}$$

We want a closed form solution, so we need to eliminate the term $T(n-k)$. To do so, we make use of the fact that we have $T(0) = b$. By setting $k = n$, $T(n-k)$ becomes $T(0)$ and hence we obtain

$$\begin{aligned} T(n) &= T(n-n)+nc, \\ &= T(0)+nc, \\ &= b+nc. \end{aligned}$$

To prove this is correct, we check to see if our equation satisfies the recurrence. Manipulating $T(n)$, we see that

$$T(n) = nc+b = (n-1)c+b+c = T(n-1)+c,$$

as required. We also have $T(0) = b+0 = b$ as required. Hence this is a valid solution.

Problem 5. Find a closed form for the following recurrence relation:

$$T(n) = \begin{cases} 3T(n-1), & \text{if } n > 0, \\ c, & \text{if } n = 0. \end{cases}$$

Solution

To find a closed form for T, we will use the method of telescoping. For $n > 0$ we have

$$\begin{aligned} T(n) &= 3T(n-1), \\ T(n) &= 3(3T(n-2)) = 3^2 T(n-2), \\ T(n) &= 3^2(3T(n-3)) = 3^3 T(n-3). \end{aligned}$$

Continuing the pattern, we see that the general form for $T(n)$ seems to be

$$T(n) = \begin{cases} 3^k T(n-k), & \text{if } n > 0, \text{for all } 0 \leq k \leq n, \\ c, & \text{if } n = 0. \end{cases}$$

We want a closed form solution, so we need to eliminate the term $T(n-k)$. To do so, we make use of the fact that we have $T(0) = c$. By setting $k = n$, $T(n-k)$ becomes $T(0)$ and hence we obtain

$$\begin{aligned} T(n) &= 3^n T(n-n), \\ &= 3^n T(0), \\ &= c3^n. \end{aligned}$$

We should now check that this solution is correct. Substituting our proposed solution into the recurrence, we find

$$3T(n-1) = 3(c3^{n-1}) = c(3 \times 3^{n-1}) = c3^n = T(n),$$

as required. We also have $T(0) = c \times 3^0 = c$ as required.