1 Problem 2.3-3 on page 39

Base Case Proof

\[ T(2) = 2 = 2 \lg 2. \] Therefore, when \( n = 2, T(n) = n \lg n. \)

Inductive Hypothesis

If \( a = 2^b \) for \( b > 1 \), we assume \( T(a) = T(2^b) = a \lg a = b^b. \)

Inductive Step

The next inductive step to be proved is: if the above hypothesis holds, the following is true

\[ T(2^{b+1}) = T(2^b) + 2^{b+1} \]

Proof

\[
T(2^{b+1}) = 2T(2^b) + 2^{b+1} \quad \text{recurrence relation}
\]
\[
= 2 \cdot b^b + 2^{b+1} \quad \text{inductive hypothesis}
\]
\[
= (b + 1) \cdot 2^{b+1} \quad \text{simple manipulation} \quad \blacksquare
\]

Conclusion

Combining the base case, hypothesis and the inductive step, we are able to conclude \( T(n) = n \lg n \) (where \( T(n) \) is recursively defined as above) when \( n = 2^k \) for \( k > 1 \).

2 Problem 2.3-4 on page 39

Let \( T(n) \) be the running time needed to recursively sort \( A[1 \ldots n] \) using insertion sort. Then, we have:

\[
T(n) = \begin{cases} 
O(1) & n = 1 \\
T(n-1) + O(n) & n > 1 
\end{cases}
\]

because we sort the first \( n - 1 \) elements \( (A[1 \ldots n - 1]) \) using insertion sort recursively (which accounts for \( T(n-1) \)), and then insert \( A[n] \) into the sorted array \( A[1 \ldots n - 1] \) (which accounts for \( O(n) \)).
3 Problem 2-3(a) on page 41

Since we don’t know running time of each operation, let’s assume the following.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Running time per operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignment (‘=’)</td>
<td>a</td>
</tr>
<tr>
<td>subtraction</td>
<td>b</td>
</tr>
<tr>
<td>addition</td>
<td>c</td>
</tr>
<tr>
<td>multiplication</td>
<td>d</td>
</tr>
<tr>
<td>comparison</td>
<td>e</td>
</tr>
</tbody>
</table>

Line 1

The running time is \(a\).

Line 2

This is equivalent to ‘for(\(i = n; i >= 0; i --\))’. Therefore, the running time is \(a + (n + 2)b + (n + 2)e\). Note that the loop body will be executed \(n + 1\) times, and both subtraction and comparison will be conducted one more time to hit the termination condition.

Line 3

Each of this loop body contains one multiplication, one addition and one assignment, and this loop body is executed for \(n + 1\) times. Therefore, the running time is \((n + 1)(a + c + d)\).

Θ notation

In total, the running time is \(2a + (n + 2)b + (n + 1)c + (n + 1)d + (n + 2)e\). Since \(a, b, c, d,\) and \(e\) are all constants, the running time is \(Θ(n)\).

4 Problem 3-3(a), fourth row only, on pages 61-62; justify your answers!

The functions sorted in decreasing order in terms of growth rate:

\((n + 1)!\), \(e^n\), \((\lg n)^{\lg n}\), \(4^{\lg n}\), \(2^{\lg n}\), \(\sqrt{\lg n}\)

- Taking the log of \((n + 1)!\), we have \(\lg((n + 1)!) = \sum_{i=1}^{n+1} \lg i \geq \sum_{i=\frac{n}{2}}^{n+1} \lg i \geq \frac{n}{2} \lg \frac{n}{2} = \frac{n}{2} \lg n - \frac{n}{2}\). Taking the log of \(e^n\), we have \(\lg(e^n) = n \lg e\). Clearly \(\frac{n}{2} \log n - \frac{n}{2} = \Omega(n \log e)\), which indicates \((n + 1)! = \Omega(e^n)\)

- Taking the log of \(e^n\), we have \(\lg(e^n) = n \lg e\). Taking the log of \((\lg n)^{\lg n}\), we have \(\lg((\lg n)^{\lg n}) = \lg n \lg \lg n\). \(n \lg e = \sqrt{n} \times \sqrt{n} \lg e = \Omega(\lg n \lg \lg n)\), which indicates \(e^n = \Omega((\lg n)^{\lg n})\)

- Taking the log of \((\lg n)^{\lg n}\), we have \(\lg((\lg n)^{\lg n}) = \lg n \lg \lg n\). Taking the log of \(4^{\lg n}\), we have \(\lg(4^{\lg n}) = 2 \lg n\). \(\lg n \lg \lg n = \Omega(2 \lg n)\), which indicates \((\lg n)^{\lg n} = \Omega(4^{\lg n})\)

- \(4^{\lg n} = n^{\lg 4} = n^2\), \(2^{\lg n} = n\). Clearly \(4^{\lg n} = \Omega(2^{\lg n})\)

- As \(2^{\lg n} = n\), it is clear that \(n = \Omega(\sqrt{\lg n})\).
5 Problem 4-3(a) on page 108; solve this problem two ways: first with the master theorem on page 94, and then using secondary recurrences (pages 13 in the January 10 notes)

Using Master Theorem

\[ a = 4, b = 3, f(n) = n \log n \Rightarrow \frac{af(n/b)}{f(n)} = \frac{4(n/3) \log (n/3)}{n \log n} = \frac{4}{3} (1 - \log_3 3) > 1 \]

Therefore, \( T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_3 4}) \).

Using secondary recurrence

Let \( n_i = n \) and \( n_{i-1} = n/3 \). Further, we assume \( T(1) \) is the base case and \( n_0 = 1 \). This does not affect the final result since we are solving for the \( \Theta \) notation of the function. Then,

\[ n_i = 3n_{i-1} \Rightarrow n_i = \alpha 3^i \quad \text{(corresponds to \( (E - 3) \))} \]

Since \( n_0 = 1 \), \( \alpha = 1 \) and \( n_i = 3^i \). Further, define \( F(i) = T(n_i) \). Then, the original recurrence:

\[ T(n) = T(n_i) = 4T(n/3) + n \log n = 4T(n_{i-1}) + n \log n \]

becomes

\[ F(i) = 4F(i - 1) + n \log n \]

We have supposed \( n = n_i \), and we derived that \( n_i = 3^i \). Therefore, the final recurrence to solve is:

\[ F(i) = 4F(i - 1) + (3^i) \log 3^i = 4F(i - 1) + \log 3 \cdot i3^i \]

which is annihilated by \( (E - 4)(E - 3)^2 \). The corresponding closed formula is \( \alpha_1 \cdot 4^i + (\alpha_2 \cdot i + \alpha_3)3^i \), which is \( \Theta(4^i) \). Recall that \( n = 3^i \). We can achieve the final \( \Theta \) notation by undoing the substitution as follows:

\[ T(n) = F(i) = \Theta(4^i) = \Theta(4^{\log_3 n}) = \Theta(n^{\log_3 4}) \]