1 Problem 2.3-3 on page 39

Base Case Proof

\(T(2) = 2 = 2 \log 2\). Therefore, when \(n = 2\), \(T(n) = n \log n\).

Inductive Hypothesis

If \(a = 2^b\) for \(b > 1\), we assume \(T(a) = T(2^b) = a \log a = b2^b\).

Inductive Step

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

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

Proof

\[
T(2^{b+1}) = 2T(2^b) + 2^{b+1} \quad \text{recurrence relation}
\]
\[
= 2 \cdot b2^b + 2^{b+1} \quad \text{inductive hypothesis}
\]
\[
= (b + 1) \cdot 2^{b+1} \quad \text{simple manipulation} \quad \blacksquare
\]

Conclusion

Combining the base case, hypothesis and the inductive step, we are able to conclude \(T(n) = n \log n\) (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 \(A[1 \cdots n]\) 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 \((A[1 \cdots n - 1])\) using insertion sort recursively (which accounts for \(T(n - 1)\)), and then insert \(A[n]\) into the sorted array \(A[1 \cdots n - 1]\) (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 $(n + 1)(a + c + d)$.

$\Theta$ notation

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

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

The functions sorted according to their growth rates:

$2^{2^n}, \left(\frac{3}{2}\right)^n, n^3, \lg(n!), \lg^2 n, n^{1/\lg n}$

- $2^{2^n} = \Omega(2^n)$, $(3/2)^n = O(2^n)$.
- $(3/2)^n$ grows exponentially and $n^3$ grows polynomially.
- $\lg(n!) = O(\lg(n^n)) = O(n \lg n) = O(n^3)$.
- $\lg(n!) = \Omega(\lg(2^n)) = \Omega(n \lg 2) = \Omega(n)$, $\lg^2 n = O(n)$.
- $n^{1/\lg n} = n^{\log_2 2} = 2$.

** $2^{2^n} \neq 4^n$
5 Problem 4-3(c) on page 108; solve this problem two ways: first with the master theorem on page 94, and then using secondary recurrences (pages 13-14 in the January 11 notes)

Using Master Theorem

\[
a = 4, b = 2, f(n) = n^2 \sqrt{n} \Rightarrow \frac{af(n/b)}{f(n)} = \frac{4(n/2)^2 \sqrt{n/2}}{n^2 \sqrt{n}} = \sqrt{\frac{1}{2}} < 1
\]

Therefore, \( T(n) = \Theta(f(n)) = \Theta(n^2 \sqrt{n}) \).

Using secondary recurrence

Let \( n_i = n \) and \( n_{i-1} = n/2 \). 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,

\[
n_i = 2n_{i-1} \Rightarrow n_i = \alpha 2^i \quad \text{(corresponds to (E - 2))}
\]

Since \( n_0 = 1, \alpha = 1 \) and \( n_i = 2^i \). Further, define \( F(i) = T(n_i) \). Then, the original recurrence:

\[
T(n) = T(n_i) = 4T(n/2) + n^2 \sqrt{n} = 4T(n_{i-1}) + n^2 \sqrt{n}
\]

becomes

\[
F(i) = 4F(i - 1) + n^2 \sqrt{n}
\]

We have supposed \( n = n_i \), and we derived that \( n_i = 2^i \). Therefore, the final recurrence to solve is:

\[
F(i) = 4F(i - 1) + (2^i)^2 \sqrt{2^i}
\]

which is annihilated by \((E - 4)(E - 2^{2.5})\). The corresponding closed formula is \( \alpha_1 4^i + \alpha_2 (2^{2.5})^i \), which is \( \Theta(2^{2.5i}) \). Recall that \( n = 2^i \). We can achieve the final \( \Theta \) notation by undoing the substitution as follows:

\[
T(n) = F(i) = \Theta(2^{2.5i}) = \Theta((2^i)^{2.5}) = \Theta(n^{2.5}) = \Theta(n^2 \sqrt{n})
\]