Math 320 Homework 1.9

Chris Rytting

September 19, 2015

1.53 (i)

As

$$
\begin{aligned}
&a = 3 \\
&b = 2 \\
&d = 2 \\
&\log_2(3) = 1.58 \\
&d > \log_2(3) \\
&\implies T(n) \in O(n_2)
\end{aligned}
$$

1.53 (ii)

As

$$
\begin{aligned}
&a = 4 \\
&b = 2 \\
&d = 1 \\
&\log_2(4) = 2 \\
&d < \log_2(3) \\
&\implies T(n) \in O(n^{\log_2(4)})
\end{aligned}
$$

1.53 (iii)

Master Theorem does not apply, as $n^n \notin O(n^d) \quad d \in \mathbb{R}$

1.53 (iv)

Master Theorem does not apply, as $2^n \notin O(n^d) \quad d \in \mathbb{R}$

1.53 (v)

As

$$
\begin{aligned}
&a = 16 \\
&b = 8 \\
&d = 1 \\
&\log_8(16) = \frac{4}{3} \\
&d < \log_8(16) \\
&\implies T(n) \in O(\log_8(16))
\end{aligned}
$$

1.53 (vi)

As

$$
\begin{aligned}
&a = 2 \\
&b = 2 \\
&d = 1 \\
&\log_2(2) = 1 \\
&d = \log_2(2) \\
&\implies T(n) \in O(n\log(n))
\end{aligned}
$$

1.54

$$
\begin{aligned}
a(T\lceil n-1\rceil) + g(n) &= a(a(T\lceil n-2\rceil) + g(n-1)) + g(n) \\
&= a^2(T\lceil n-2\rceil) + ag(n-1)) + g(n) \\
&= a^3(T\lceil n-3\rceil) + a^2g(n-2)) + ag(n-1)) + g(n) \\
&\cdots \\
&= a^n(T\lceil n-n\rceil) + a^ng(n-n) + a^{n-1}g(n-(n-1)) + \cdots + ag(n-1)) + g(n) \\
&= a^nT_0 + \sum_{i=1}^{n} a^{n-i}g(i)
\end{aligned}
$$

Which is the desired result.

1.55

By 1.27 and 1.36, we have that

$$T(n) = a^nT_0 + \sum_{i=1}^{n} a^{n-i}g(i) = a(T\lceil n-1\rceil) + c$$

We can see that for addition, $a = 1, g(i) = c$

$$T(n) = T_0 + \sum_{i=1}^{n} c = 1(T\lceil n-1 \rceil) + c$$

$$\implies T_0 + cn \in O(n)$$

Since $T_0 + cn < mn \quad m \in \mathbb{N}$

1.56

(Proof similar to 1.55) By 1.27 and 1.36, we have that

$$T(n) = a^n T_0 + \sum_{i=1}^{n} a^{n-i} g(i) = a(T\lceil n-1 \rceil) + c$$

We can see that for addition, $a = 1, g(i) = c$

$$T(n) = T_0 + \sum_{i=1}^{n} c = 1(T\lceil n-1 \rceil) + c$$

$$\implies T_0 + cn \in O(n)$$

Since $T_0 + cn < mn \quad m \in \mathbb{N}$

1.58

Take a unimodal sequence x_i with n elements. The way we will find the maximum is by cutting the sequence in half and comparing the first element of the right half and the last element of the left half. If the element from the right half is greater than the element from the left half, then we know the maximal element lies in the right half. Otherwise, the maximal element lies in the left half. Whichever half we know to contain the maximal element, we will do the same thing as we did to the whole sequence. This algorithm will terminate in $\log_2(n)$ steps or fewer, since the worse case scenario will require us splitting the series $\log_2(n)$ times before finding the maximum. Note that $\log_2(n) \in \log(n)$. See attached python script for actual algorithm.

```
def unimodal(tosort):
    """
    tosort: list of unimodal sequence to find maximum of.
    """
    half1 = tosort[:(len(tosort)/2)]
    half2 = tosort[(len(tosort)/2):]
    if len(tosort) == 1:
        return tosort
    else:
        if half1[-1] > half2[0]:
            return unimodal(half1)
        elif half1[-1] < half2[0]:
            return unimodal(half2)

unimodalsequence = [1,2,3,4,17,8,2,1]
print unimodal(unimodalsequence)
```

The results are as follows:

```
[17]
```