# Analysis of Algorithms: Solutions 5

```
                  X
                  X
                  X
                  X
number of         X        X
homeworks      X  X        X
               X  X     X  X
               X  X  X  X  X
               X  X  X  X  X
            X  X  X  X  X  X     X
         X  X  X  X  X  X  X  X  X
         -------------------------
         1  2  3  4  5  6  7  8  9
                  grades
```

**Problem 1**
Determine asymptotic bounds for each of the following recurrences.

**(a)** $T(n) = 3 \cdot T(n/27) + n$.

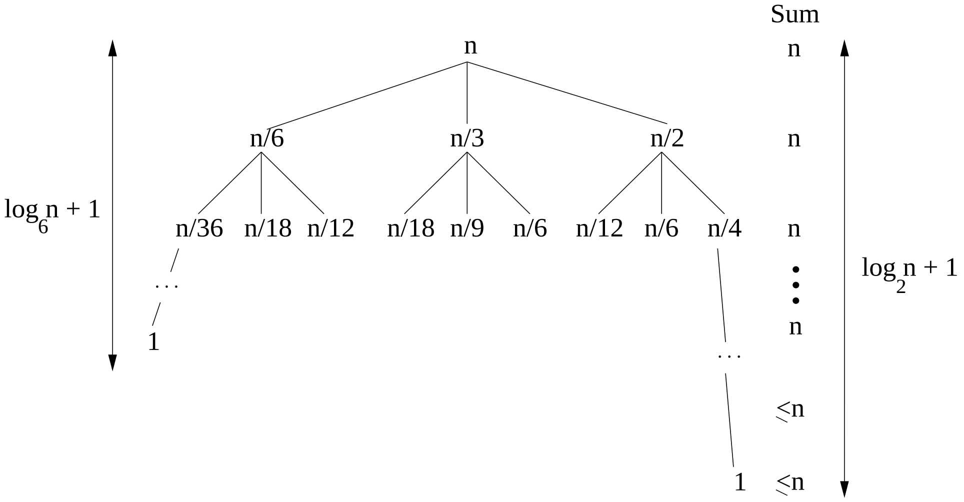
We use the master method; since $3 < 27$, we conclude that $T(n) = \Theta(n)$.

**(b)** $T(n) = 3 \cdot T(n/27) + \sqrt[3]{n}$.

$$\begin{aligned}
T(n) &= \sqrt[3]{n} + 3 \cdot T(\frac{n}{27}) \\
&= \sqrt[3]{n} + 3 \cdot \left(\sqrt[3]{\frac{n}{27}} + 3 \cdot T(\frac{n}{27^2})\right) \\
&= \sqrt[3]{n} + 3 \cdot \sqrt[3]{\frac{n}{27}} + 3^2 \cdot T(\frac{n}{27^2}) \\
&= \sqrt[3]{n} + 3 \cdot \sqrt[3]{\frac{n}{27}} + 3^2 \cdot \left(\sqrt[3]{\frac{n}{27^2}} + 3 \cdot T(\frac{n}{27^3})\right) \\
&= \sqrt[3]{n} + 3 \cdot \sqrt[3]{\frac{n}{27}} + 3^2 \cdot \sqrt[3]{\frac{n}{27^2}} + 3^3 \cdot T(\frac{n}{27^3}) \\
&\quad \cdots \\
&= \sqrt[3]{n} + 3 \cdot \sqrt[3]{\frac{n}{27}} + 3^2 \cdot \sqrt[3]{\frac{n}{27^2}} + 3^3 \cdot \sqrt[3]{\frac{n}{27^3}} + 3^4 \cdot \sqrt[3]{\frac{n}{27^4}} + ... + 3^{\log_{27} n} \cdot \sqrt[3]{\frac{n}{27^{\log_{27} n}}} \\
&= \underbrace{\sqrt[3]{n} + \sqrt[3]{n} + \sqrt[3]{n} + \sqrt[3]{n} + ... + \sqrt[3]{n}}_{\log_{27} n + 1} \\
&= \sqrt[3]{n} \cdot (\log_{27} n + 1) \\
&= \Theta(\sqrt[3]{n} \cdot \lg n).
\end{aligned}$$

**(c)** $T(n) = T(n/6) + T(n/3) + T(n/2) + n$.

We use the iteration method, which leads to the following tree:




The summation gives a lower and upper bound for $T(n)$:

$$n \cdot (\log_6 n + 1) \leq T(n) \leq n \cdot (\log_2 n + 1),$$

which implies that $T(n) = \Theta(n \cdot \lg n)$.

**(d)** $T(n) = T(n-1) + 1/2^n$.

$$\begin{aligned} T(n) &= T(n-1) + \frac{1}{2^n} \\ &= T(n-2) + \frac{1}{2^{n-1}} + \frac{1}{2^n} \\ &\quad \cdots \\ &= \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + ... + \frac{1}{2^{n-1}} + \frac{1}{2^n} \\ &= 1 - \frac{1}{2^n} \\ &= \Theta(1). \end{aligned}$$

**(e)** $T(n) = 2 \cdot T(\sqrt{n}) + 1$.

We "unwind" the recurrence until reaching some constant value of $n$, say, until $n \leq 2$:

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \leq 2 \\ 2 \cdot T(\sqrt{n}) + 1, & \text{if } n > 2 \end{cases}$$

For convenience, assume that $n = 2^{2^k}$, for some natural value $k$.

$$
\begin{aligned}
T(2^{2^k}) &= 1 + 2 \cdot T(\sqrt{2^{2^k}}) \\
&= 1 + 2 \cdot T(2^{2^{k-1}}) \\
&= 1 + 2 \cdot \left(1 + 2 \cdot T(\sqrt{2^{2^{k-1}}})\right) \\
&= 1 + 2 + 4 \cdot T(2^{2^{k-2}}) \\
&= 1 + 2 + 4 \cdot \left(1 + 2 \cdot T(\sqrt{2^{2^{k-2}}})\right) \\
&= 1 + 2 + 4 + 8 \cdot T(2^{2^{k-3}}) \\
&\quad \cdots \\
&= 1 + 2 + 4 + \ldots + 2^{k-1} + T(2) \\
&= 2^k - 1 + \Theta(1) \\
&= \Theta(2^k).
\end{aligned}
$$

Finally, we note that $k = \lg \lg n$, which means that $T(n) = \Theta(2^{\lg \lg n}) = \Theta(\lg n)$.

**Problem 2**

The standard analysis of Merge-Sort$(A, p, q)$ is based on the assumption that we pass $A[1..n]$ by a pointer. If a language does not allow passing an array by a pointer, we may have two other options; for each option, determine the running time of Merge-Sort.

**(a)** Copy all elements of the array $A[1..n]$, which takes $\Theta(n)$ time.

Let $n$ be the size of the array $A[1..n]$, and $m$ be the size of the segment $A[p..q]$, sorted by the recursive call Merge-Sort$(A, p, q)$. The time of copying the array is $\Theta(n)$, and the time of the Merge operation is $\Theta(m)$, which leads to the following recurrence:

$$T(m) = 2 \cdot T(m/2) + \Theta(n) + \Theta(m).$$

Since $m \leq n$, we conclude that

$$T(m) = 2 \cdot T(m/2) + \Theta(n) = 2 \cdot T(m/2) + c \cdot n,$$

and unwind this recurrence as follows:

$$
\begin{aligned}
T(m) &= c \cdot n + 2 \cdot T(m/2) \\
&= c \cdot n + 2 \cdot c \cdot n + 2^2 \cdot T(m/4) \\
&= c \cdot n + 2 \cdot c \cdot n + 2^2 \cdot c \cdot n + 2^3 \cdot T(m/8) \\
&\quad \cdots \\
&= c \cdot n + 2 \cdot c \cdot n + 2^2 \cdot c \cdot n + \ldots + 2^{\lg m - 1} \cdot c \cdot n + 2^{\lg m} \cdot c \cdot n \\
&= (2^{\lg m + 1} - 1) \cdot c \cdot n \\
&= (2 \cdot m - 1) \cdot c \cdot n \\
&= \Theta(m \cdot n).
\end{aligned}
$$

Thus, the running time of MERGE-SORT$(A, p, q)$ is $\Theta(m \cdot n)$, where $m$ is the size of the segment $A[p..q]$. The top-level call to the sorting algorithm is MERGE-SORT$(A, 1, n)$; for this call, we have $m = n$, which means that the time complexity is

$$T(n) = \Theta(n^2).$$

**(b)** Copy the elements of the segment $A[p..q]$, which takes $\Theta(q - p + 1)$ time.

The complexity of copying the segment is $\Theta(m)$, which is the same as the time of the MERGE procedure; hence, copying does not affect the complexity of the algorithm. The recurrence is the same as the standard recurrence for MERGE-SORT, and the overall time is $\Theta(n \cdot \lg n)$.