filler

Master Theorem

Overview

1. Theorem
2. Proof
3. Examples

Theorem

Suppose a complexity function $T(n)$ is eventually non-decreasing and satisfies

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d) \qquad \text{for } n > 1,\ n \text{ a power of } b$$
$$T(1) = c$$

Where $b \geq 2$ and $d \geq 0$ are constant *integers*, and a and c are constant such that $a > 0$ and $c > 0$. Then

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \lg n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Also, using the same theorem, if in the recurrence relation

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

the equality is replaced by an inequality, such as

$$T(n) \leq aT\left(\frac{n}{b}\right) + O(n^d) \qquad \text{or} \qquad T(n) \geq aT\left(\frac{n}{b}\right) + O(n^d)$$

Then the results above hold true with "big O" or Ω, respectively, replacing Θ

Proof

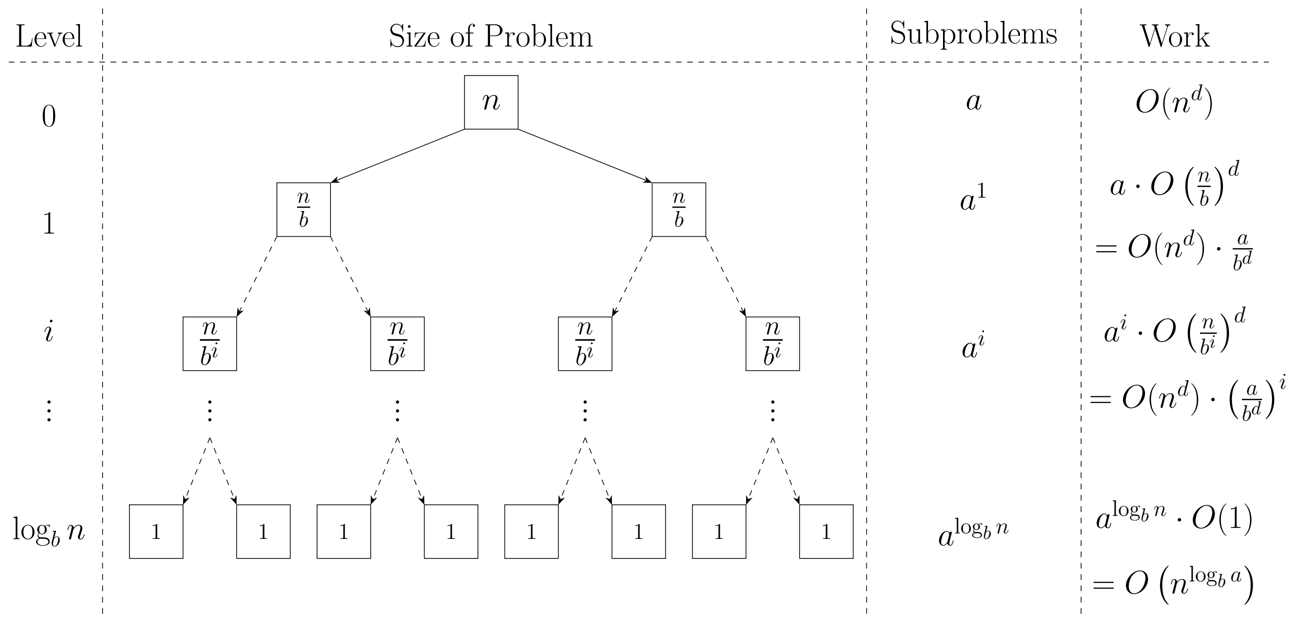
It is important to understand the different parts of the recurrence relation used in the theorem.

$$\underbrace{T(n)}_{\text{Recurrence relation}} = \overbrace{a}^{\text{Subproblems}} T \underbrace{\left(\frac{n}{b}\right)}_{\text{Divisor}} + O(n^{\overbrace{d}^{\text{Degree of work time complexity}}})$$

The general solution to the recurrence relation above can be described as

$$T(n) = \sum_{i=0}^{\log_b n} O\left(n^d\right)\left(\frac{a}{b^d}\right)^i$$

This can be visually seen by drawing the recursion tree generated by the relation shown above. The tree has a *depth* of $\log_b n$ and a *branching factor* of a. At level i, there are a^i nodes each with a work of $O\left(\frac{n}{b}\right)^d$. The value of the relation is the sum of all the nodes on the tree. The sum is found by summing the work from all $\log_b n$ levels.



Total Work $= \sum_{i=0}^{\log_b n} O\left(n^d\right)\left(\frac{a}{b^d}\right)^i$

Now that the solution is found, let's address the three cases shown in the theorem.

Case One

$$T(n) \in \Theta\left(n^d\right) \iff a < b^d, \text{ or } d > \log_b a.$$

For case one we will use substitution.

$$\sum_{i=0}^{\log_b n} \underbrace{O(n^d)}_{\text{a}} \underbrace{\left(\frac{a}{b^d}\right)^i}_{\text{r}} = O(n^d) \iff \left(\frac{a}{b^d}\right)^i < 1$$

Proof:

$$\sum_{i=0}^{n-1} ar^i = ar^0 + ar + ar^2 + ar^3 + \cdots + ar^{n-1}$$

$$= a\left(\frac{1-r^n}{1-r}\right) \qquad r \neq 1$$

$$r > 1 \text{ or } r < 1 \qquad = \begin{cases} O(a), & r < 1 \\ O(ar^{n-1}), & r > 1 \end{cases}$$

$$\sum_{i=0}^{n-1} ar^i \leq O(a) \iff r < 1$$

$$\sum_{i=0}^{n-1} ar^i \leq O(ar^{n-1}) \iff r > 1$$

Case Two

$$T(n) \in \Theta\left(n^d \lg n\right) \iff a = b^d, \text{ or } d = \log_b a$$

Proof:

$$\frac{a}{b^d} = 1 \iff d = \log_b a$$

$$\begin{aligned}
\sum_{i=0}^{\log_b n} O(n^d)\left(\frac{a}{b^d}\right)^i &= \sum_{i=0}^{\log_b n} O(n^d) \\
&= (\log_b n + 1) \cdot O(n^d) \\
&= (\log_b n) \cdot O(n^d) + \underbrace{O(n^d)}_{\text{lower order term}} \\
&= \log_b n \cdot O(n^d) \\
&= O\left(n^d \cdot \frac{\log n}{\log b}\right) \\
&= O\left(n^d \lg n\right)
\end{aligned}$$

Case Three

$$T(n) \in \Theta\left(n^{\log_b n}\right) \iff a > b^d, \text{ or } d < \log_b a.$$

Proof:

$$
\begin{aligned}
\frac{a}{b^d} > 1 &\iff d < \log_b a \\
\sum_{i=0}^{\log_b n} O(n^d)\left(\frac{a}{b^d}\right)^i &= O\left(O(n^d)\cdot\left(\frac{a}{b^d}\right)^{\log_b n}\right) \\
&= O\left(O(n^d)\cdot\frac{a^{\log_b n}}{(b^d)^{\log_b n}}\right) \\
&= O\left(O(n^d)\cdot\frac{n^{\log_b a}}{b^{\log_b n^d}}\right) \\
&= O\left(O(n^d)\cdot\frac{n^{\log_b a}}{n^d}\right) \\
&= O\left(\frac{Cn^d}{n^d}\cdot n^{\log_b a}\right) \\
&= O\left(n^{\log_b a}\right)
\end{aligned}
$$

Examples

Merge Sort

Merge Sort is a recursive algorithm, the time complexity can be expressed with the following recurrence relation.

$$T(n) = \underbrace{2}_{a} T\underbrace{\left(\frac{n}{2}\right)}_{b} + \underbrace{O(n)}_{d=1}$$

Using the Master Theorem we get

$$a = 2 \qquad\qquad b = 2 \qquad\qquad d = 1$$

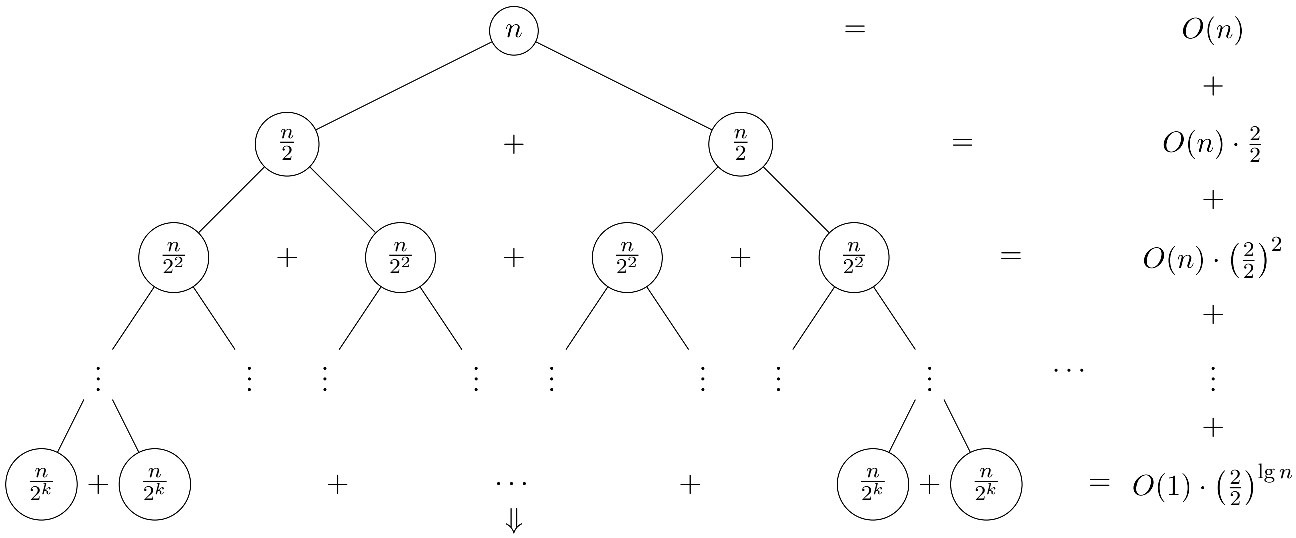
Which means the middle case holds true for Merge Sort

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \lg n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

$$a = b^d \qquad\qquad 2 = 2^1$$

$$T(n) \in \Theta(n \lg n).$$

The proof of this can be seen visually with the diagram below



$$O\left(\sum_{i=0}^{\lg n} 2^i \cdot \frac{n}{2^i}\right) = O\left(\sum_{i=0}^{\lg n} n\right) = O(\lg(n) \cdot n) \Leftrightarrow O(n \lg n)$$

Other Examples

Consider the following recurrence relation

$$T(n) = \underbrace{9}_{a} T(n/\underbrace{3}_{b}) + 5n^{\overbrace{1}^{d}}$$

By the Master Theorem, using case 3, because $9 > 3^1$

$$T(n) \in \Theta\left(n^{\log_3 9}\right) = \Theta(n^2).$$

Consider the following recurrence relation

$$T(n) = \underbrace{3}_{a} T(n/\underbrace{2}_{b}) + 3n^{\overbrace{3}^{d}} + 2n^2.$$

By the Master Theorem, using case 1, because $3 < 2^3$

$$T(n) \in \Theta(n^3).$$