Design and Analysis of Algorithm

M. Sakthi Balan

Department of Computer Science & Engineering
LNMIIT, Jaipur

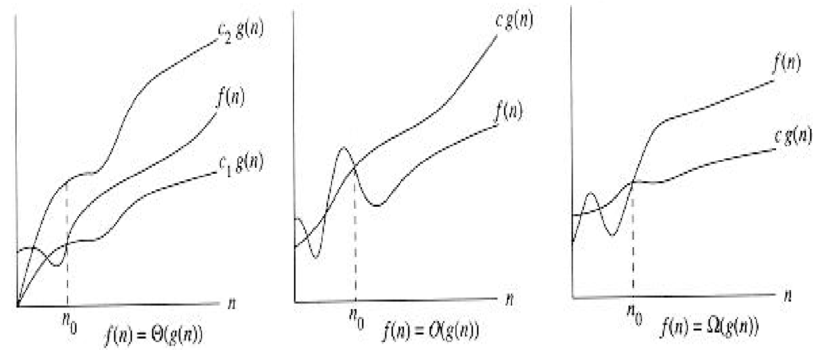
Contents

Asymptotic Analysis

Running time is a function on domain $N = \{0, 1, 2, \cdots\}$

- $\mathcal{O}$-notation – upper bound
- Θ-notation – asymptotically tight bound
- Ω-notation – lower bound

Asymptotic Analysis



LNMIIT

Asymptotic Analysis

Given two function $T(n)$ and $f(n)$:

- $T(n) = \mathcal{O}(f(n))$ if $\exists$ constants $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0, T(n) \leq cf(n)$
- $T(n) = \Omega(f(n))$ if $\exists$ constants $c > 0$ and $n_0 \geq 0$ such that $\forall n \geq n_0, T(n) \geq cf(n)$
- $T(n) = \Theta(f(n)$ if $T(n) = \mathcal{O}(f(n))$ and $T(n) = \Omega(f(n))$

LNMIIT

Asymptotic Analysis

Let f and g be two functions such that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$$

where $c(> 0)$ is a constant. Then

$$f(n) = \Theta(g(n))$$

Properties of Asymptotic Growth

1. If $f = \mathcal{O}(g)$ and $g = \mathcal{O}(h)$ then $f = \mathcal{O}(h)$
2. If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$
3. If $f = \Theta(g)$ and $g = \Theta(h)$ then $f = \Theta(h)$
4. If $f = \mathcal{O}(h)$ and $g = \mathcal{O}(h)$ then $f + g = \mathcal{O}(h)$
   This sum can be extended to k number of functions also
5. Suppose f and g are non-negative functions such that $g = \mathcal{O}(f)$. Then $f + g = \Theta(f)$
6. For a polynomial f of degree d in which the coefficients are positive. Then $f = \mathcal{O}(n^d)$
7. For every $b > 1$ and every $x > 0$ we have $log_b n = \mathcal{O}(n^x)$
8. For every $r > 1$ and every $d > 0$ we have $n^d = \mathcal{O}(r^n)$

Recurrence Equation

A recurrence is an equation or an inequality that describes a function in terms of its value of smaller inputs.

Recurrence Equation

```
MERGE-SORT(A,p,r)
  if p < r
    then q = ⌊(p+r)/2⌋
      MERGE-SORT(A,p,q)
      MERGE-SORT(A,q+1,r)
      MERGE(A,p,q,r)
```

Recurrence Equation

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c, \\ aT(n/b) + D(n) + C(n) & \text{otherwise.} \end{cases}$$

Recurrence Equation

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

Recurrence Equation – Substitution Method

1. Involves guessing of formula
2. Then use the mathematical induction to show it works

Recurrence Equation – Substitution Method

Let $\mathcal{O}(nlogn)$ be the guess for the equation $T(n) = 2T(n/2) + n$
To prove: $T(n) \leq cnlogn$ for a constant $c > 0$

$$\begin{aligned} T(n) &\leq 2(cn/2log(n/2)) + n \\ &\leq cnlog(n/2) + n \\ &= cnlogn - cnlog2 + n \\ &= cnlogn - cn + n \\ &\leq cnlogn \end{aligned}$$

where $c \geq 1$

Recurrence Equation – Substitution Method

$$T(n) = 2.T(n/2) + 1$$

Let us assume the bound to be $\mathcal{O}(n)$. So we need to show that $T(n) \leq cn$

$$\begin{aligned} T(n) &\leq 2.cn/2 + 1 \\ &= cn + 1 \end{aligned}$$

$$\begin{aligned} T(n) &\leq (cn/2 - b) + (cn/2 - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b \end{aligned}$$

Recurrence Equation – Substitution Method

$$T(n) = 2.T(n/2) + 1$$

Let us assume the bound to be $\mathcal{O}(n)$. So we need to show that $T(n) \leq cn$

$$\begin{aligned} T(n) &\leq 2.cn/2 + 1 \\ &= cn + 1 \end{aligned}$$

$$\begin{aligned} T(n) &\leq (cn/2 - b) + (cn/2 - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b \end{aligned}$$

Recurrence Equation – Substitution Method

Guess $T(n) \leq cn$

$$\begin{aligned} T(n) &\leq 2(cn/2) + n \\ &\leq cn + n \\ &= \mathcal{O}(n) \end{aligned}$$

Is that correct?!

LNMIIT

Recurrence Equation – Substitution Method

Suppose the recurrence equation is

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + logn$$

Take $m = logn$

$$T(2^m) = 2T(2^{m/2}) + m$$

Take $S(m) = T(2^m)$

$$S(m) = 2S(m/2) + m$$

$S(m)$ is $\mathcal{O}(mlogm)$

$$T(n) = \mathcal{O}(lognloglogn)$$

LNMIIT

Recurrence Equation – Iteration Method

Consider the iteration:

$$T(n) = 3T(n/4) + n$$

We can iterate as follows:

$$\begin{aligned}
T(n) &= n + 3T(n/4) \\
&= n + 3(n/4 + 3T(n/16)) \\
&= n + 3(n/4 + 3(n/16 + 3T(n/64))) \\
&= n + 3(n/4) + 9(n/16) + 27T(n/64),
\end{aligned}$$

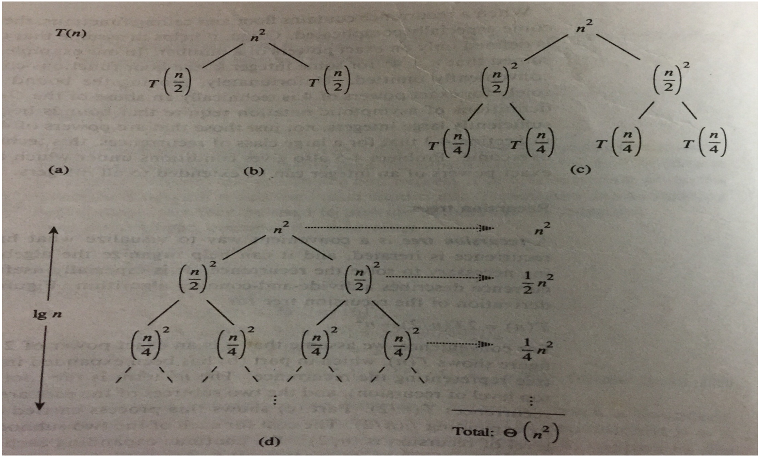
Recurrence Equation – Iteration Method

$$\begin{aligned} T(n) &\le n + 3n/4 + 9n/16 + 27n/64 + \cdots + 3^{log_4 n}\Theta(1) \\ &\le n\sum_{i=0}^{\infty}\left(\frac{3}{4}\right)^i + \Theta(n^{log_4 3}) \\ &= 4n + o(n) \\ &= \mathcal{O}(n). \end{aligned}$$

LNMIIT

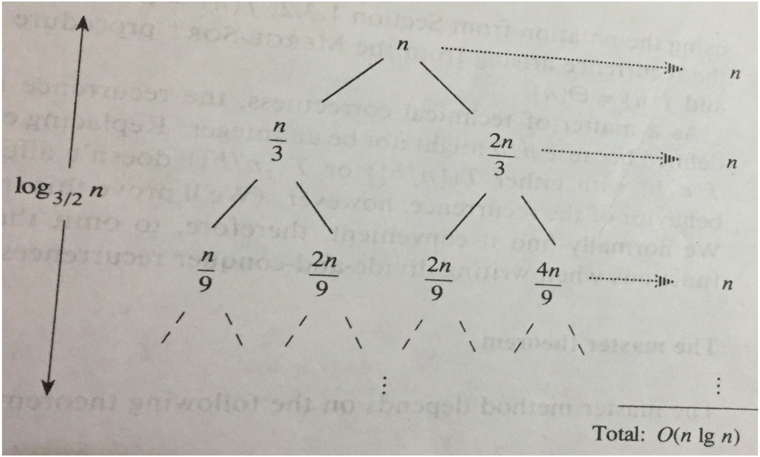
Recurrence Equation – Recursion Trees

$$T(n) = 2T(n/2) + n^2$$



Recurrence Equation – Recursion Trees

$$T(n) = T(n/3) + T(2n/3) + n$$



Recurrence Equation – Master Method

Master Theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n).$$

Then $T(n)$ can be bounded asymptotically as follows,

1. If $f(n) = \mathcal{O}(n^{log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{log_b a})$
2. If $f(n) = \Theta(n^{log_b a})$, then $T(n) = \Theta(n^{log_b a} logn)$
3. If $f(n) = \Omega(n^{log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

Recurrence Equation – Master Method

$$T(n) = 9T(n/3) + n$$

$$T(n) = T(2n/3) + 1$$

$$T(n) = 3T(n/4) + nlogn$$

LNMIIT

Recurrence Equation – Master Method

$$T(n) = 9T(n/3) + n$$

$a = 9, b = 3, f(n) = n$ and $n^{\log_b a} = \Theta(n^2)$.
Hence $T(n) = \Theta(n^2)$.

LNMIIT

Recurrence Equation – Master Method

$$T(n) = 9T(n/3) + n$$

$a = 9, b = 3, f(n) = n$ and $n^{\log_b a} = \Theta(n^2)$.
Hence $T(n) = \Theta(n^2)$.

Recurrence Equation – Master Method

$$T(n) = T(2n/3) + 1$$

$a = 1, b = 3/2, f(n) = 1$ and $n^{\log_b a} = 1$.
Hence $T(n) = \Theta(logn)$.

Recurrence Equation – Master Method

$$T(n) = T(2n/3) + 1$$

$a = 1, b = 3/2, f(n) = 1$ and $n^{\log_b a} = 1$.
Hence $T(n) = \Theta(logn)$.

LNMIIT

Recurrence Equation – Master Method

$$T(n) = 3T(n/4) + nlogn$$

$a = 3, b = 4, f(n) = nlogn$ and $n^{\log_b a} = \mathcal{O}(n^{0.793})$.
Also, For larger n,
$af(n/b) = 3(n/4)log(n/4) \leq (3/4)nlogn = cf(n)$ for $c = 3/4$.
Hence $T(n) = \Theta(nlogn)$.

LNMIIT

Recurrence Equation – Master Method

$$T(n) = 3T(n/4) + nlogn$$

$a = 3, b = 4, f(n) = nlogn$ and $n^{\log_b a} = \mathcal{O}(n^{0.793})$.
Also, For larger n,
$af(n/b) = 3(n/4)log(n/4) \leq (3/4)nlogn = cf(n)$ for $c = 3/4$.
Hence $T(n) = \Theta(nlogn)$.

LNMIIT

Recurrence Equation – Master Method

$$T(n) = 2T(n/2) + nlogn$$

$a = 2, b = 2, f(n) = nlogn$ and $n^{\log_b a} = n$.
But *nlogn* is asymptotically larger than *n* but not polynomially!
So Master theorem is not applicable to this recurrence relation.

LNMIIT

Recurrence Equation – Master Method

$$T(n) = aT(n/b) + n^c$$

where $a, b \geq 1$ and $c > 0$ then

$$T(n) = \begin{cases} \Theta(n^{log_b a}) & \text{if } a > b^c, \\ \Theta(n^c log_b n) & \text{if } a = b^c, \\ \Theta(n^c) & \text{if } a < b^c. \end{cases}$$

Recurrence Equation – Master Method

$$
\begin{aligned}
T(n) &= aT(\frac{n}{b}) + n^c \\
&= n^c + a((\frac{n}{b})^c + aT(\frac{n}{b^2})) \\
&= n^c + (\frac{a}{b^c})n^c + a^2T(\frac{n}{b^2}) \\
&= \ldots \\
&= n^c + (\frac{a}{b^c})n^c + (\frac{a}{b^c}))^2 n^c + (\frac{a}{b^c}))^3 n^c + \ldots \\
& (\frac{a}{b^c}))^{log_b n-1} n^c + a^{log_b n} T(1)
\end{aligned}
$$

Recurrence Equation – Master Method

Case 1: $a < b^c$

$$a < b^c \iff \frac{a}{b^c} < 1 \implies \sum_{k=0}^{\log_b n - 1} \left(\frac{a}{b^c}\right)^k$$

$$\leq \sum_{k=0}^{\infty} \left(\frac{a}{b^c}\right)^k = \frac{1}{1 - \left(\frac{a}{b^c}\right)} = \Theta(1)$$

$$a < b^c \iff \log_b a < \log_b b^c = c$$

$$\begin{aligned} T(n) &= n^c \sum_{k=0}^{\log_b n - 1} \left(\frac{a}{b^c}\right)^k + n^{\log_b a} \\ &= n^c \cdot \Theta(1) + n^{\log_b a} \\ &= \Theta(n^c) \end{aligned}$$

Recurrence Equation – Master Method

Case 2: $a = b^c$

$$
\begin{aligned}
a = b^c \quad & \Longleftrightarrow \quad \frac{a}{b^c} = 1 \Longrightarrow \sum_{k=0}^{log_b n - 1} (\frac{a}{b^c})^k \\
& \quad = \sum_{k=0}^{log_b n - 1} 1 = \Theta(log_b n) \\
a = b^c \quad & \Longleftrightarrow \quad log_b a = log_b b^c = c \\
T(n) \quad & = \quad n^c \sum_{k=0}^{log_b n - 1} (\frac{a}{b^c})^k + n^{log_b a} \\
& = \quad n^c \cdot \Theta(log_b n) + n^{log_b a} \\
& = \quad \Theta(n^c log_b n)
\end{aligned}
$$

Recurrence Equation – Master Method

Case 3: $a > b^c$

$$a > b^c \iff \frac{a}{b^c} > 1 \implies \sum_{k=0}^{log_b n-1} (\frac{a}{b^c})^k$$

$$= \Theta((\frac{a}{b^c})^{log_b n}) = \Theta(\frac{a^{log_b n}}{(b^c)^{log_b n}}) = \Theta(\frac{a^{log_b n}}{n^c})$$

$$\begin{aligned} T(n) &= n^c \cdot \Theta(\frac{a^{log_b n}}{n^c}) + n^{log_b a} \\ &= \Theta(n^{log_b a}) + n^{log_b a} \\ &= \Theta(n^{log_b a}) \end{aligned}$$

LNMIIT