CS207 Design and Analysis of Algorithms

Sajith Gopalan

Indian Institute of Technology Guwahati

sajith@iitg.ac.in

January 19, 2022

Divide and Conquer

Divide and Conquer

- solve a problem recursively
  - Divide the problem into a number of smaller instances of the same problem
  - For each of these subproblems, if its size is sufficiently large, then Conquer it recursively, else Conquer it directly
  - Combine the solutions to the subproblems into the solution for the original problem

Master Theorem

Theorem

For positive constants a and b and integer valued functions $f(n)$ *and* $T(n)$ *on* $\mathbb{N}^+$, *if* $T(n) = aT(n/b) + f(n)$ *for* $n > 1$, *but* $\Theta(1)$ *for* $n = 1$, *where* n/b *stands for either* $\lceil n/b \rceil$ *or* $\lfloor n/b \rfloor$, *then*
(i) $f(n) = O(n^{\log_b a - \epsilon})$, *for a constant* $\epsilon \in (0, 1)$, *implies*

$$T(n) = \Theta(n^{\log_b a})$$

(ii) $f(n) = \Theta(n^{\log_b a})$ *implies*

$$T(n) = \Theta(n^{\log_b a} \log_2 n)$$

(iii) $f(n) = \Omega(n^{\log_b a + \epsilon})$, *for a constant* $\epsilon \in (0, 1)$, *and there exists* $c < 1$ *and* $m > 0$ *such that for all* $n \geq m$, $af(n/b) \leq cf(n)$ *implies*

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

Proof

It is enough to upper bound $T(n) = aT(\lceil n/b \rceil) + f(n)$ for each of the cases.
Define $n_j = n$ if $j = 0$, $\lceil \frac{n_{j-1}}{b} \rceil$ if $j > 0$

Then
$n_0 \leq n, \quad n_1 \leq \frac{n}{b} + 1, \quad n_2 \leq \frac{n}{b^2} + \frac{1}{b} + 1, \quad n_3 \leq \frac{n}{b^3} + \frac{1}{b^2} + \frac{1}{b} + 1$

$n_j \leq \frac{n}{b^j} + \sum_{k=0}^{j-1} \frac{1}{b^j} < \frac{n}{b^j} + \frac{b}{(b-1)}$

Therefore,
$n_{\lfloor log_b n \rfloor} < \frac{n}{b^{\lfloor log_b n \rfloor}} + \frac{b}{(b-1)} < \frac{n}{b^{log_b n - 1}} + \frac{b}{(b-1)} = b + \frac{b}{(b-1)} = O(1)$

Proof

$T(n_0) = aT(n_1) + f(n_0)$

$= a^2 T(n_2) + af(n_1) + f(n_0)$

$= a^3 T(n_3) + a^2 f(n_2) + af(n_1) + f(n_0)$

$= a^j T(n_j) + a^{j-1} f(n_{j-1}) + \ldots + a^0 f(n_0)$

$= a^{\lfloor \log_b n \rfloor} T(O(1)) + \sum_{k=0}^{\lfloor \log_b n \rfloor - 1} a^k f(n_k)$ (When $j = \lfloor \log_b n \rfloor$)

$< a^{\log_b n} T(O(1)) + \sum_{k=0}^{\lfloor \log_b n \rfloor - 1} a^k f(n_k)$

$= \Theta(n^{\log_b a}) + \sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k f(n_k)$

Proof of case i

$$\sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k f(n_k) = O\left(\sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k (n_k)^{\log_b a - \epsilon}\right) =$$

$$O\left(n^{\log_b a - \epsilon} \sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k / b^{k \log_b a - k\epsilon}\right) = O\left(n^{\log_b a - \epsilon} \sum_{k=0}^{\lfloor log_b n \rfloor - 1} b^{k\epsilon}\right) =$$

$$O\left(n^{\log_b a - \epsilon} . \frac{b^{\epsilon \lfloor \log_b n \rfloor} - 1}{b^{\epsilon} - 1}\right) = O\left(n^{\log_b a - \epsilon} . \frac{n^{\epsilon} - 1}{b^{\epsilon} - 1}\right) = O\left(n^{\log_b a}\right)$$

Proof of case ii

$$\sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k f(n_k) = \Theta\left(\sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k (n_k)^{\log_b a}\right) =$$

$$\Theta\left(n^{\log_b a} \sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k / b^{k \log_b a}\right) = \Theta\left(n^{\log_b a} \sum_{k=0}^{\lfloor log_b n \rfloor - 1} 1\right) =$$

$$\Theta\left(n^{\log_b a} . \log_b n\right) = \Theta\left(n^{\log_b a} . \log_2 n\right)$$

Proof of case iii

$$\exists c < 1, \exists m > 0, \forall n \geq m, [af(n_1) \leq cf(n_0)]$$

$$\implies \exists c < 1, \exists m > 0, \forall n \geq m, [f(n_1) \leq (c/a)f(n_0)]$$

$$\implies \exists c < 1, \exists m > 0, \forall n \geq bm, [f(n_2) \leq (c/a)^2 f(n_0)]$$

$$\implies \exists c < 1, \exists m > 0, \forall n \geq b^{j-1}m, [f(n_j) \leq (c/a)^j f(n_0)]$$

That is, $f(.)$ at a sufficiently large n_j is upper bounded by $(c/a)^j f(n)$

For smaller values, $f(.)$ is upper bounded by a constant.

Proof of case iii

$$\sum_{k=0}^{\lfloor log_b n \rfloor - 1} a^k f(n_k) \leq \sum_{k=0}^{\lfloor log_b n \rfloor - 1} c^k f(n) + O(1) \leq$$

$$f(n) \sum_{k=0}^{\infty} c^k + O(1) = f(n) \frac{1}{1-c} + O(1) = \Theta(f(n))$$