Design And Analysis Of Algorithms

Condensed Notes

May 4, 2025

Chapter 1

Asymptotic Analysis of Algorithms

1.1 Master theorem

Given a recurrence $T(n) = aT(n/b) + f(n)$ where $a \geq 1$ and $b > 1$

1. if $f(n) = O(n^{log_b a - \epsilon})$ then $T(n) = \Theta(n^{log_b a})$

2. if $f(n) = O(n^{log_b a} log^k n)$ then $T(n) = \Theta(n^{log_b a} log^{k+1} n)$

3. if $f(n) = \Omega(n^{log_b a + \epsilon})$ then $T(n) = \Theta(f(n))$

which simply means

if $f(n)$ is polynomially smaller than $n^{log_b a}$, then $n^{log_b a}$ dominates, and the runtime is $\Theta(n^{log_b a})$

If $f(n)$ is instead polynomially larger than $n^{log_b a}$, then $f(n)$ dominates, and the runtime is $\Theta(f(n))$.

Finally, if $f(n)$ and $n^{\log_b a}$ are asymptotically the same, then $T(n) = \Theta(n^{log_b a} log n)$.

1.1.1 Simpler Form

Given a recurrence $T(n) = aT(n/b) + f(n)$ where $a \geq 1$,$b > 1$ and $f(n) = \Theta(n^k log^p n)$

1. if $log_b a > k$ then $T(n) = \Theta(n^{log_b a})$
2. if $log_b a = k$ then
   - if $p > -1$ then $T(n) = \Theta(n^k log^{p+1} n)$
   - if $p >= -1$ then $T(n) = \Theta(n^k loglogn)$
   - if $p < -1$ then $T(n) = \Theta(n^k)$
3. if $log_b a < k$ then
   - if $p \geq 0$ then $T(n) = \Theta(n^k log^p n)$
   - if $p < 0$ then $T(n) = \Theta(n^k)$

1.2 Master theorem for subtract recurrences

Given a recurrence $T(n) = cifn \leq 1elseaT(n-b) + f(n)$

and $f(n)isO(n^k)$ then

if$(a\rangle 1) T(n) = O(n^k)$

if$(a = 1) T(n) = O(n^{k+1})$

if$(a\langle 1 T(n) = O(n^k a^{n/b})$

1.3 Points

- f(n) has to be polynomial for applicatrion of master theorem
- log n = O(n)