Design and Analysis of Algorithms

L06b: Master Theorem

Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Resources

- T1: Levitin
- T2: Horowitz

Solving Recurrence Relation

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

- Let `n=b`^{k}, then

$$T(b^k) = aT(b^{k-1}) + f(b^k)$$

$$= a[aT(b^{k-2}) + f(b^{k-1})] + f(b^k)$$

$$= a^2T(b^{k-2}) + af(b^{k-1}) + f(b^k)$$

$$= a^3T(b^{k-3}) + a^2f(b^{k-2}) + af(b^{k-1}) + a^0f(b^k)$$

$$\vdots$$

$$= a^kT(b^{k-k}) + a^{k-1}f(b^{k-(k-1)} + a^2f(b^{k-2}) + af(b^{k-1}) + a^0f(b^k)$$

$$= a^kT(1) + a^{k-1}f(b^1) + a^{k-2}f(b^2) + \ldots + a^0f(b^k)$$

$$= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \ldots + f(b^k)/a^k]$$

Solving Recurrence Relation

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

$$T(b^k) = aT(b^{k-1}) + f(b^k)$$

$$= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \ldots + f(b^k)/a^k]$$

$$= a^k[T(1) + \sum_{j=1}^{k} \frac{f(b^j)}{a^j}]$$

- Thus, `T(n)` depends upon `a`, `b`, and `f()`

As $n=b^k$, then $k=\log_b n$, thus

$a^k=a^{\log_b n} = n^{\log_b a}$, the recursion equation becomes

$$T(n) = n^{log_b a}[T(1) + \sum_{j=1}^{log_b n} \frac{f(b^j)}{a^j}] \qquad (1)$$

Master Theorem

$T(n) = aT(n/b) + f(n)$ for n=b^k, k=1,2, ...

T(1) = c

where, a≥1, b≥2, c>0

$$T(n) = n^{log_b a}[T(1) + \sum_{j=1}^{log_b n} \frac{f(b^j)}{a^j}] \qquad (1)$$

If $f(n)=\Theta(n^d)$, where $d\geq 0$, then

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

Summary

- Analysis of Non Recursive algorithms
- Analysis of recursive algorithms
- Recurrence relation examples