Design and Analysis of Algorithms

L12b: Recurrence Relation
MaxMin using Master Theorem

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Divide and Conquer: Recurrence Relation

$$T(n) = \begin{cases} g(n) & n\ small \\ T(n_1) + T(n_2) + \ldots + T(n_k) + f(n) & otherwise \end{cases}$$

- `T(n)`: time complexity for a problem of input size n
- `g(n)`: time complexity for solving directly for small inputs
- `f(n)`: Time complexity for dividing the problem into `k` subproblems and combining again from the solutions of `k` sub problems.
- `k` would vary depending upon the problem
  - Generally, $n_1 = n_2 = \ldots = n_k$
  - Assuming `a` instances, each of size `n/b`

$$T(n) = \begin{cases} T(1) & n = 1 \\ aT(n/b) + f(n) & n > 1 \end{cases}$$

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

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

$$\begin{aligned} T(b^k) &= aT(b^{k-1}) + f(b^k) \\ &= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \ldots + f(b^k)/a^k] \\ &= a^k[T(1) + \sum_{j=1}^{k} \frac{f(b^j)}{a^j}] \end{aligned}$$

- 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

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

Recurrence Relation: Master Theorem

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

$T(1) = c$ where $a \geq 1$, $b \geq 2$, $c>0$.

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

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

Recurrence :MaxMin Algo

- Recurrence relation for MaxMin

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

$T(1) = 0$

$T(2) = 1$

Using the Master theorem

$a=2 \ (a\geq1)$, $b=2 (b\geq2)$, $c=T(1)=0$, and

$f(n)=2 \in\Theta(n^d) \Rightarrow f(n)\in\Theta(1) \Rightarrow d=0$

Thus, $b^d=b^0=1 \Rightarrow a>b^d$ #3rd case in Master Theorem

Further, $\log_b a=\log_2 2=1$, thus from Master Theorem

$T(n)=\Theta(n^{\log_b a}) = \Theta(n^1) = \Theta(n)$

which corresponds to $3n/2 - 2$

Recurrence :MaxMin Algo

- Recurrence relation for MaxMin

```
T(n)=2T(n/2)+2
T(1) = 0
T(2) = 1
```

Note: we can't apply general recurrence relation to T(2) i.e. we can't write

`T(2)=2T(2/2)+2=2T(1)+2=0+2=2,`

Hence we need to stop at `T(2)` and can't go to `T(1)`.

Thus, recurrence relation becomes

$$T(n) = n^{log_b a}[\frac{T(2)}{a} + \sum_{j=2}^{log_b n} \frac{f(b^j)}{a^j}]$$

$$= n^{log_2 2}[\frac{1}{2} + \sum_{j=2}^{log_b n} \frac{2}{2^j}] \quad = n[\frac{1}{2} + \sum_{j=2}^{log_b n} \frac{1}{2^{j-1}}]$$

Recurrence :MaxMin Algo

$$T(n) = n[\frac{1}{2} + \sum_{j=2}^{log_b n} \frac{1}{2^{j-1}}]$$

$$= n[\frac{1}{2} + \frac{\frac{1}{2} - (\frac{1}{2})^{log_2 n}}{1 - \frac{1}{2}}] \qquad = n[\frac{1}{2} + \frac{\frac{1}{2} - (\frac{1}{n})}{\frac{1}{2}}]$$

$$= n[\frac{1}{2} + 1 - \frac{2}{n}] \qquad = n[\frac{3}{2} - \frac{2}{n}]$$

$$= \frac{3}{2}n - 2$$