جامعة طرابلس ـ كلية تقنية المعلومات

Design and Analysis Algorithms

تصميم وتحليل خوارزميات

ITGS301

Lecture 6 : المحاضرة السادسة

Tripoli University - Faculty of Information Technology
Marwa Solla
m.solla@uot.edu.ly

Master Method

The Master Method is used for solving the following types of recurrence

$$T(n) = a\,T(n/b) + f(n)$$

Where a=> 1, b>1, and f is a function, f(n) > 0.

- n is the size of the problem.
- a is the number of subproblems in the recursion.
- n/b is the size of each subproblem. (Here it is assumed that all subproblems are essentially the same size.)
- f (n) is the sum of the work done outside the recursive calls, which includes the sum of dividing the problem and the sum of combining the solutions to the subproblems.

Master Theorem:

It is possible to complete an asymptotic tight bound in these three cases:

Idea: compare f(n) with $n^{\log_b a}$

Case 1: $T(n) = \Theta(n^{\log_b a})$ if $f(n) < n^{\log_b a}$

Case 2: $T(n) = \Theta(n^{\log_b a} \lg n)$ if $f(n) = n^{\log_b a}$

Case 3: $T(n) = \Theta(f(n))$ if $f(n) > n^{\log_b a}$

Example 1:

Solve **T(n) = 9T(n/3)+n** using Master theorem;

a=9, b=3, f(n) =n

and $n^{\log_b a} = n^{\log_3 9} = n^2$ now, $f(n) < n^{\log_3 9}$

Therefore by case 1, **T(n) =** Θ(n2)

Example 2:

Solve T(n) = T(2n/3)+1 using Master theorem;

a=1, b=3/2, f(n) =1

and $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

now, $f(n) = \Theta(n^{\log_b a})$,

Therefore by case 2,

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

The Simple Format of Master Theorem

Let $T(n)=aT(n/b)+cn^k$. with a, b, c, k are positive constants, and $a \geq 1$ and $b>1$,

Case 1: $T(n) = O(n^{\log_b a})$, if $a>b^k$.
Case 2: $T(n) = O(n^k \log n)$, if $a=b^k$.
Case 3 : $T(n) = O(n^k)$, if $a<b^k$.

$$f(n) = \Theta(n) \Rightarrow f(n) = O(n)$$

if f(n) = Theta(g(n)) you can say f(n) = O(g(n)) too!

Example 1:

Solve $T(n) = 4T(n/2) + n^3$. Using the Master method.
a= 4, b=2, k=3
$b^k = 2^3$

$a < b^k$ so the case 3 is applied
$T(n) = O(n^3)$.

Example 2:

Solve $T(n) = 2T(n/2) + 1$ Using the Master method.
a= 2, b=2, k=0
$b^k = 2^0$

$\therefore$ a > b^k so the case 2 is applied

T(n) = O(n).

Example 3:

Solve T(n) = 9T(n/3) + n. Using the Master method.
a= 9, b=3, k=1
$b^k = 3^1$

a > b^k so the case 1 is applied
$T(n) = O(n^{\log_b a}). = O(n^2)$.

Extended Version of Master Theorem

$$\boxed{T(n) = a\,T\left(\frac{n}{b}\right) + \theta\left(n^k \log^p n\right)}$$

Master's Theorem

F(n) = $n^p \log^p n$

- Here, a >= 1, b > 1, k >= 0 and p is a real number.

Compare : $\log_b a$ with K

Extended Version of Master Theorem

Case 1: if $\log_b a > K$

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

Case 2 : if $\log_b a = K$

If $p > -1$ then $T(n) = O(n^k \log^{p+1} n)$

If $p = -1$ then $T(n) = O(n^k \log \log n)$

If $p < -1$ then $T(n) = O(n^k)$

Case 3 : if $\log_b a < K$

If $p >= 0$ then $T(n) = O(n^k \log^p n)$

If $p < 0$ then $T(n) = O(n^k)$

Example3

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

We compare the given recurrence relation with $T(n) = aT(n/b) + \theta(n^k \log^p n)$.

Then, we have- $a = 2$ $b = 2$ $k = 1$ $p = 1$

Now, $a = 2$ and $b^k = 2^1 = 2$.

Clearly, $a = b^k$.

So, we follow case-02.

Since $p = 1$, so we have-

$T(n) = \theta(n^{\log_b a}.\log^{p+1} n)$

$T(n) = \theta(n^{\log_2 2}.\log^{1+1} n)$

Thus, $T(n) = 2T(n/2) + n \log n \Longrightarrow T(n) = n \log2\, n$ (Case 2)

$\boxed{\mathbf{T(n) = \theta(n\log^2 n)}}$

Inadmissible equations

The following equations cannot be solved using the master theorem:

- $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$

  a is not a constant; the number of subproblems should be fixed

- $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

  non-polynomial difference between $f(n)$ and $n^{\log_b a}$ (see below; extended version applies)

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  $a < 1$ cannot have less than one sub problem

- $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

  $f(n)$, which is the combination time, is not positive

- $T(n) = T\left(\frac{n}{2}\right) + n(2 - \cos n)$

  case 3 but regularity violation.

Master theorem limitations

Can not be used :

T(n) is not monotone , ex: sin n.

T(n) is not polynomial , ex: 2^n

a is not constants ex: a = 2^n

a < 1

Logarithmic rules

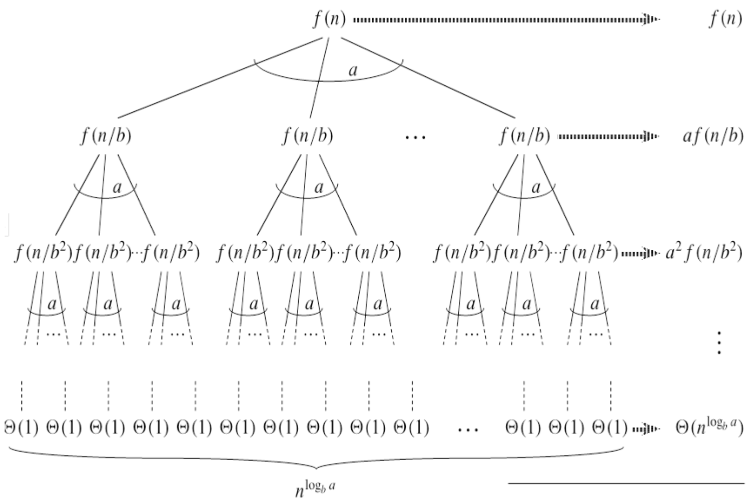
$$
\begin{aligned}
\log_a(bc) &= \log_a(b) + \log_a(c) \\
\log_a(b^c) &= c\log_a(b) \\
\log_a(1/b) &= -\log_a(b) \\
\log_a(1) &= 0 \\
\log_a(a) &= 1 \\
\log_a(a^r) &= r \\
\log_{1/a}(b) &= -\log_a(b) \\
\log_a(b)\log_b(c) &= \log_a(c) \\
\log_b(a) &= \frac{1}{\log_a(b)} \\
\log_{a^m}(a^n) &= \frac{n}{m}, \qquad m \neq 0
\end{aligned}
$$

Recursion Tree Method

Idea: Convert the recurrence into a tree, use this tree to rewrite the function as sum, and then use techniques to solve recurrence.

The recursion tree generated by *T(n) = a T(n/b) + f(n).*

Where
a is number of sub problems that are solved recursively
b is size of each sub problem relative to n
n/b is the size of the input to recursive call.
F(n) is the cost (time) of dividing and recombining the sub problem.



Each node represents the cost of a single sub problem.
Sum up the costs with each level to get level cost.

Costs with each level = a^i f(n/b_i)

for (i = 0,1,2,3,…,logb n-1)

where ai is the number of subtrees (or nodes at level i).
but at the last level T(1) = 1
f(1)= 1.

$n/b_i = 1 \rightarrow n = b_i \rightarrow i = \log_b n$
so at last level when T(1) =1
cost = $a^i f(n/b_i)$
= $a^i . f(1)$
= $a^i . (1)$

when $i = \log_b n \rightarrow a^i = a^{\log_b n}$
$a^{\log b\ n} = n^{\log b\ a}$

= $a^i . (1)$
= $(1). a^{\log_b n}$
= $(1). n^{\log_b a}$
= $T(n) = \theta (n^{\log_b a})$.

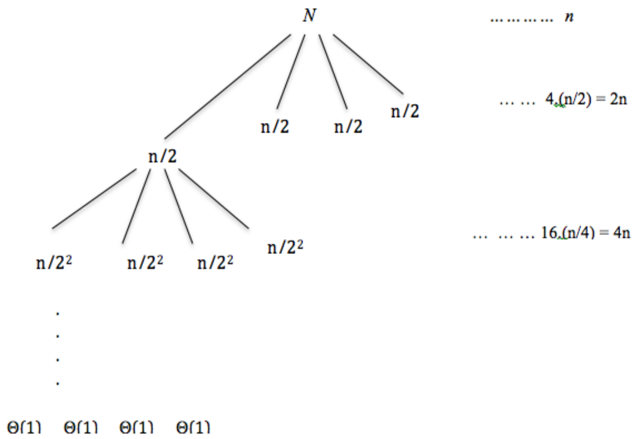
the sum up all the level cost to get total cost.

$$\text{Total: } \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$$

Example:

solve T(n) = 4 T(n/2) + n using recursion tree.

answer



$T(n) = [\ 2^{\log_b n-1+1} - 1\ /\ 2-1\ ].n + n^2$

$T(n) = [\ 2^{\log_b n} - 1\ /\ 2-1\ ].\ n + n^2$

$T(n) = [\ n^{\log_2 b} - 1\ /\ 2-1\ ].n + n^2$

$T(n) = [\ n - 1].n + n^2$

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

$T(n) = 2n^2 - n$

$\therefore$ Total cost $= \Theta(n^2)$.

The End .