Master theorem

In the analysis of algorithms, the "master theorem" provides a cookbook solution in asymptotic terms (using Big O notation) for recurrence relations of types that occur in the analysis of many divide & conquer algorithms.

Consider algorithm :

```
Begin T(n) {
    if (n>1) {
        Statements
           :
           :
        T(n/b)
        T(n/b)
          .
        T(n/b)
    }
End
```

→ f(n)

for above algorithm we will write general recurrence relation

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \qquad \text{where } a \geq 1,\ b > 1$$

$n \rightarrow$ is the size of the problem

$a \rightarrow$ is the n.of subproblems in recursion

$n/b \rightarrow$ size of each subproblem

$f(n) \rightarrow$ cost of the work done other than recursive calls.

Master theorem:

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

for $a \geq 1$, $b > 1$, $k \geq 0$ and p is real numbers

Case 1: $a > b^k$

then $T(n) = \theta\left(n^{\log_b^a}\right)$

Case 2: $a = b^k$

a) if $p > -1$ then $T(n) = \theta\left(n^{\log_b^a} \log^{p+1} n\right)$

b) if $p = -1$ then $T(n) = \theta\left(n^{\log_b^a} \log\log n\right)$

c) if $p < -1$ then $T(n) = \theta\left(n^{\log_b^a}\right)$

Case 3 $a < b^k$

a) if $p \geq 0$ then $T(n) = \theta\left(n^k \log^p n\right)$

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

Note: Not all recurrence relations can be solved by using master theorem. It is applicable when $a \geq 1$, $b > 1$ and $f(n)$ is asymptotically positive function.

Eg1: $9T\left(\frac{n}{3}\right)+n$

Here $a=9$ $b=3$ $k=1$

a b^k check whether $a\geq 1, b>1, k\geq 0$
$p=0$

$9>3^1$

case $a>b^k$ then $\theta(n^{\log_b^a})$

$=\theta(n^{\log_3^9})$

$=\theta(n^{\log_3^{3^2}})$

$=\theta(n^2)$

$$\boxed{\therefore T(n) = O(n^2)}$$

Eg2: $T(n) = 3T\left(\frac{n}{4}\right)+cn^2$

$a=3$ $b=4$ $k=2$ $p=0$

$a\geq 1, b>1, k\geq 0, p=0$ so apply master theorem

$3<4^2 \Rightarrow 3<16$

case $a<b^k$ and $p\geq 0$ then $\theta(n^k \log^p n)$

$\Rightarrow \theta(n^2 \log^0 n)$

$=\theta(n^2)$

$$\boxed{\therefore T(n) = O(n^2)}$$

3. $T(n) = 3T(n/4) + n\log n$

$a=3 \quad b=4 \quad k=1 \quad p=1$

$a \geq 1$
$b > 1$
$k \geq 0$
p real numbel

$3 < 4^1$

case $a < b^k$

$p \geq 0$

$T(n) = \Theta(n^k \log^p n)$

$= \Theta(n^1 \log^1 n) = O(n\log n)$

$$\boxed{\therefore \ T(n) = O(n\log n)}$$

4. $T(n) = T\left(\frac{2n}{3}\right) + 1$

$\Rightarrow T(n) = T\left(\frac{n}{3/2}\right) + 1$

$a=1 \quad b=3/2 \quad k=0 \quad p$ is 0

Apply master theorem

$a \cdot b^k$

$1 \quad \left(\frac{3}{2}\right)^0$

$1 = 1$

case $a = b^k \quad p > -1$

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

$= \Theta(n^{\log_{3/2}^1} \log n)$

Master theorem Pit-Falls :

We can't use the master theorem if

* $T(n)$ is notmonotine, Ex: $T(n) = \sin n$
* $f(n)$ is not a polynomial. Ex: $T(n) = 2T(\frac{n}{2}) + 2^n$
* b cannot be expressed as a constant Ex; $T(n) = T(\sqrt{n})$

* $T(n) = 2^n T(\frac{n}{2}) + n^n$

  a is not a constant

* $T(n) = 0.5T(\frac{n}{2}) + n$

  $a<1$ can't have less than one sub problem

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

  f(n) which is the combination time is n't positive

* $T(\frac{n}{2}) + n(2-\cos n)$

* $T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n}$

  Here f(n) and $n^{\log_b^a}$ can be expressed with the ratio

  $$\frac{f(n)}{n^{\log_b^a}} = \frac{n/\log n}{n^{\log_2^2}} = \frac{n}{n \log n} = \frac{1}{\log n}$$ which is less than n^c, $c>0$

  $\therefore$ nonpolynomial, so does n't apply

Assignments: Solve the Recurrence relations

1. $T(n) = 3T(n/2) + n^2$

2. $T(n) = 7T(n/2) + n^2$

3. $T(n) = 4T(n/2) + n^2$

4. $T(n) = 3T(n/4) + n\log n$

5. $T(n) = 4T(n/2) + \log n$

6. $T(n) = T(n-1) + n$

7. $T(n) = 4T(n/2) + n^2 \log n$

8. $T(n) = 5T(n/2) + n^2 \log n$

9. $T(n) = 3T(n/3) + n/\log n$

10. $T(n) = 2T(n/4) + c$

11. $T(n) = T(n/4) + \log n$

12. $T(n) = T(n/2) + T(n/4) + n^2$

13. $T(n) = 2T(n/4) + \log n$

14. $T(n) = 3T(n/3) + n\log n$

15. $T(n) = 8T((n-\sqrt{n})/4) + n^2$

16. $T(n) = 2T(n/4) + \sqrt{n}$

17. $T(n) = 2T(n/4) + n^{0.51}$

18. $T(n) = 16T(n/4) + n!$