Analysis of Algorithms

Solving Recurrences

Recursion

- Many algorithms are recursive in nature.
- When we analyze them, we get a recurrence relation for time complexity
- running time on an input of size n as a function of n and the running time on inputs of smaller sizes.
- Merge Sort, to sort a given array, we divide it in two halves and recursively repeat the process for the two halves.
- Finally, we merge the results. Time complexity of Merge Sort can be written as $T(n) = 2T(n/2) + cn$.
- There are three main methods of

Substitution method

- We make a guess for the solution and then we use mathematical induction to prove the guess is correct or incorrect.
- The substitution method for solving recurrences has two parts.
- 1. Guess the correct answer.
- 2. Prove by induction that your guess is correct.
- Substitution proofs must ensure that they use the same constant as in the inductive hypothesis

Example

For example consider the recurrence $T(n) = 2T\left(\frac{n}{2}\right) + n$
We guess the solution as T(n) = $O(nlogn)$.
Now we use induction to prove our guess.
We need to prove that T(n) <= $cnlogn$.
We can assume that it is true for values smaller than n.

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$
$$\leq \frac{2cn}{2log\left(\frac{n}{2}\right)} + n$$
$$= cnLogn - cnLog2 + n$$
$$= cnLogn - cn + n$$
$$\leq cnLogn$$

Recurrence Trees

- In this method, we draw a recurrence tree and calculate the time taken by every level of tree.
- Finally, we sum the work done at all levels.
- To draw the recurrence tree, we start from the given recurrence and keep drawing till we find a pattern among levels.
- The pattern is typically an arithmetic or geometric series.

Example

- Consider the recurrence relation
- $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + cn^2$

cn^2

/ \

$T\left(\frac{n}{4}\right)$ $T\left(\frac{n}{2}\right)$

RT

- If we further break down the expression T(n/4) and T(n/2),
- we get following recursion tree.

- cn^2
- / \
- $\frac{c(n^2)}{16}$ $\frac{c(n^2)}{4}$
- / \ / \
- $T\left(\frac{n}{16}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{8}\right)$ $T\left(\frac{n}{4}\right)$

RT

- Breaking down further gives us following
- cn^2
- / \
- $\frac{c(n^2)}{16}$ $\frac{c(n^2)}{4}$
- / \ / \
- $\frac{c(n^2)}{256}$ $\frac{c(n^2)}{64}$ $\frac{c(n^2)}{64}$ $\frac{c(n^2)}{16}$
- / \ / \ / \ / \

RT

- To know the value of T(n), we need to calculate sum of tree
- nodes level by level. If we sum the above tree level by level,
- we get the following series
- $T(n) \; = \; c\left(n^2 + \frac{5(n^2)}{16} + \frac{25(n^2)}{256}\right) + \; \ldots.$
- The above series is geometrical progression with ratio $\frac{5}{16}$.

- To get an upper bound, we can sum the infinite series.
- We get the sum as $\frac{n^2}{1-\frac{5}{16}}\, which\ is\ O(n^2)$

Master theorem

- Master theorem is used for solving recurrences where all the sub-problems are of the same size.
- We assume that the input to the master method is a recurrence of the form
- $T(n) = aT\left(\frac{n}{b}\right) + \theta(n^k\ log^p n)\ log^0 n$

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

$$a\ >=\ 1, b\ >\ 1, k\ >=\ 0\ and\ p\ is\ a\ real\ number.$$

The variables

- *a* is the number of subproblems that are solved recursively; i.e. the number of recursive calls.
- *b* is the size of each subproblem relative to n; n/b is the size of the input to the recursive call.
- *f(n)* is the cost of dividing and recombining the subproblems
- To solve recurrence relations using Master's theorem, we compare a with b^k.

Case 1 and Case 2

- **Case 1**
- $If\ a\ >\ bk, then\ T(n)\ =\ \theta\ (n^{log_b a})$
- **Case 2**
- $if\ a\ =\ bk\ and$
- $If\ p\ <\ -1, then\ T(n)\ =\ \theta\ (n^{log_b a})$
- $If\ p\ =\ -1, then\ T(n)\ =\ \theta\ (n^{log_b a}. \log^2 n)$
- $If\ p\ >\ -1, then\ T(n)\ =\ \theta\ (n^{log_b a}. logp^{\ +\ 1} n)$

Case 3

- Case 3
- $If\ a\ <\ bk\ and$
- $If\ p\ <\ 0, then\ T(n) =\ O\ (n^k)$
- $If\ p\ \geq\ 0, then\ T(n) =\ \theta\ (n^k log^p n)$

Example 1

- Solve the following recurrence relation using Master's theorem-

  $T(n) = 3T(n/2) + n^2$
- We compare the given recurrence relation with
- $T(n) = aT(n/b) + \theta(n^k log^p n)$.
- Then, we have-
- $a = 3$
- $b = 2$
- $k = 2$
- $p = 0$

Now, $a \ = \ 3 \ and \ bk \ = \ 22 \ = \ 4$.
Clearly, $a \ < \ b^k$.
So, we follow case-03.

Since $p \ = \ 0$, so we have

$$T(n) \ = \ \theta \ (n^k log^p n)$$
$$T(n) \ = \ \theta \ (n^2 \log^0 n)$$

Thus,

$$T(n) = \theta(n^2)$$

Problem-02:

Solve the following recurrence relation using Master's theorem-

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

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,

$T(n) = \theta\,(n\log^2 n)$

Problem-03:

Solve the following recurrence relation using Master's theorem-

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

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

Then, we have-

$a = 2$ $b = 4$ $k = 0.51$ $p = 0$

Now, $a = 2$ and $b^k = 4^{0.51} = 2.0279$.

Clearly, $a < b^k$.

So, we follow case-03.

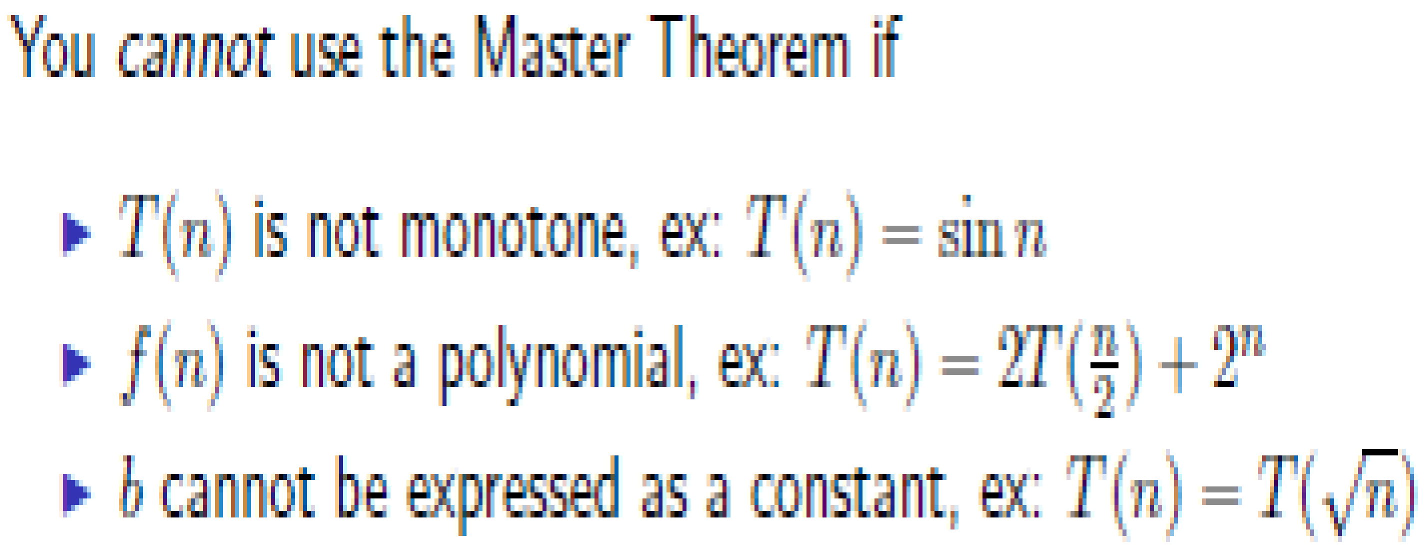
Since $p = 0$, so we have-

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

$T(n) = \theta (n^{0.51} \log^0 n)$

Thus, $\mathbf{T(n) = \theta (\sqrt{n})}$

Limitations of Master Theorem



You *cannot* use the Master Theorem if

- $T(n)$ is not monotone, 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: $T(n) = T(\sqrt{n})$

Exercises: Solve the following using master theorem

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

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

- Answers here

$T(n) = \Theta(n^2 \log n).$

$T(n) = \Theta(n^2).$