Master Theorm

The master method is a formula for solving recurrence relations of the form.

Master Method is a direct way to get the solution. The master method works only for the following type of recurrences or for recurrences that can be transformed into the following type.

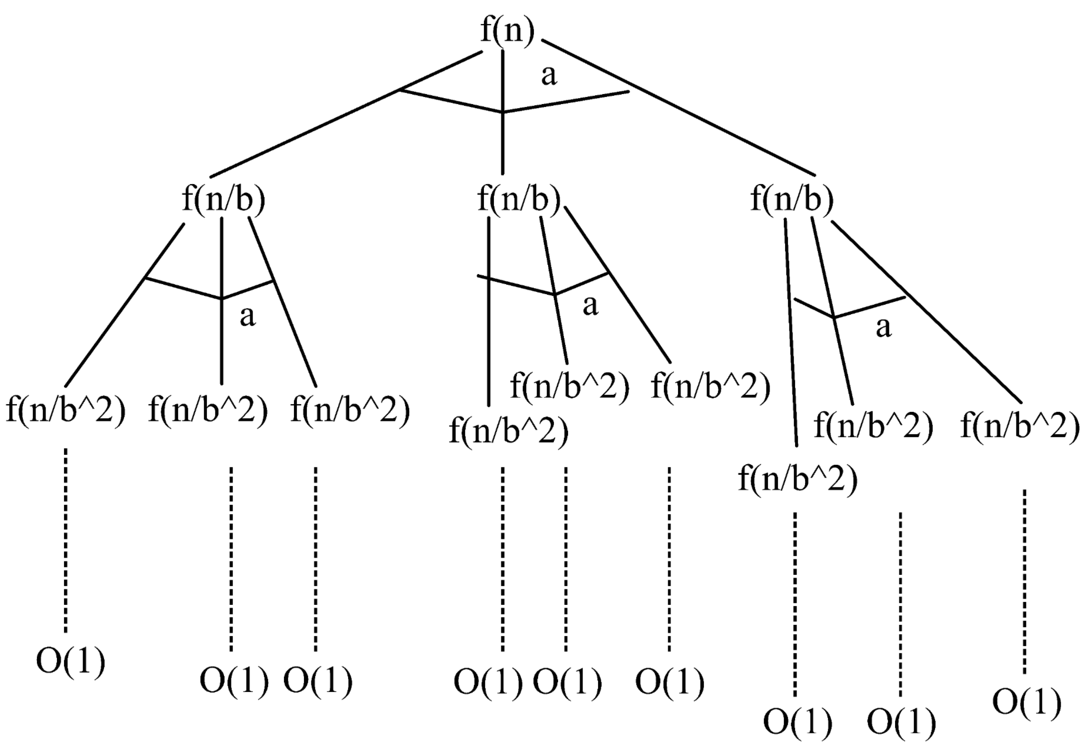
T(n) = aT(n/b) + f(n) where a >= 1 and b > 1

There are the following three cases:

- If f(n) = O(nc) where c < Logba then T(n) = Θ(nLogba)
- If f(n) = Θ(nc) where c = Logba then T(n) = Θ(ncLog n)
- If f(n) = Ω(nc) where c > Logba then T(n) = Θ(f(n))

How does this work?

We have an absolute path for a file (Unix-style), simplify it. Note that absolute path always begin with '/' (root directory), a dot in path represent current directory and double dot represents parent directory.



In the recurrence tree method, we calculate the total work done. If the work done at leaves is polynomially more, then leaves are the dominant part, and our result becomes the work done at leaves (Case 1). If work done at leaves and root is asymptotically the same, then our result becomes height multiplied by work done at any level (Case 2). If work done at the root is asymptotically more, then our result becomes work done at the root (Case 3).

Examples of some standard algorithms whose time complexity can be evaluated using the Master Method :

- Merge Sort: T(n) = 2T(n/2) + Θ(n). It falls in case 2 as c is 1 and Logba] is also 1. So the solution is Θ(n Logn)
- Binary Search: T(n) = T(n/2) + Θ(1). It also falls in case 2 as c is 0 and Logba is also 0. So the solution is Θ(Logn)

Master Theorm Limitation?

The master theorem cannot be used if:

- T(n) is not monotone. eg. T(n) = sin n
- f(n) is not a polynomial. eg. f(n) = 2n
- a is not a constant. eg. a = 2n
- a < 1

Thnka for reading :)