

## Analysis of Algorithm | Set 4 (Solving Recurrences)

In the previous post, we discussed [analysis of loops](#). Many algorithms are recursive in nature. When we analyze them, we get a recurrence relation for time complexity. We get running time on an input of size  $n$  as a function of  $n$  and the running time on inputs of smaller sizes. For example in [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 many other algorithms like Binary Search, Tower of Hanoi, etc.

There are mainly three ways for solving recurrences.

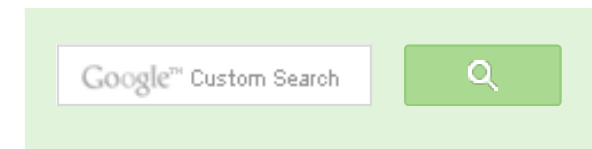
**1) Substitution Method:** We make a guess for the solution and then we use mathematical induction to prove the the guess is correct or incorrect.

For example consider the recurrence  $T(n) = 2T(n/2) + n$

We guess the solution as  $T(n) = O(n \log n)$ . Now we use induction to prove our guess.

We need to prove that  $T(n) \leq cn \log n$ . We can assume that it is true for values smaller than  $n$ .

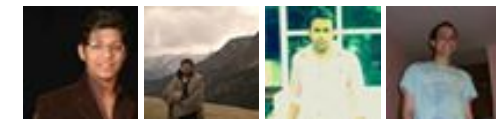
$$\begin{aligned}
 T(n) &= 2T(n/2) + n \\
 &\leq cn/2 \log(n/2) + n \\
 &= cn \log n - cn \log 2 + n \\
 &= cn \log n - cn + n \\
 &\leq cn \log n
 \end{aligned}$$



GeeksforGeeks



53,522 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

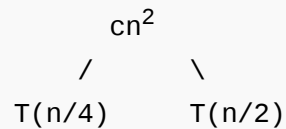
[Mathematical Algorithms](#)

[Recursion](#)

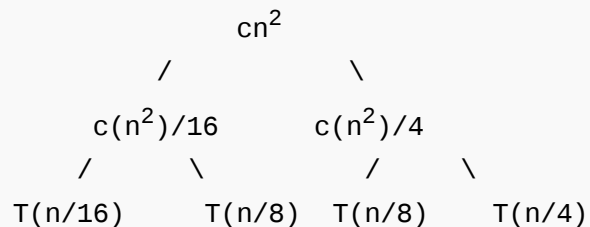
**2) Recurrence Tree Method:** In this method, we draw a recurrence tree and calculate the time taken by every level of tree. Finally, we sum the work done at all levels. To draw the recurrence tree, we start from the given recurrence and keep drawing till we find a pattern among levels. The pattern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

For example consider the recurrence relation

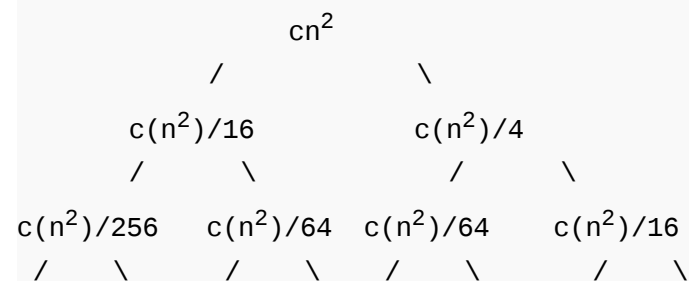
$$T(n) = T(n/4) + T(n/2) + cn^2$$



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



Breaking down further gives us following



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

$$T(n) = c(n^2 + 5(n^2)/16 + 25(n^2)/256) + \dots$$

The above series is geometrical progression with ratio  $5/16$ .



## Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

To get an upper bound, we can sum the infinite series.

We get the sum as  $(n^2)/(1 - 5/16)$  which is  $O(n^2)$

### 3) Master Method:

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

$$T(n) = aT(n/b) + f(n) \text{ where } a \geq 1 \text{ and } b > 1$$

There are following three cases:

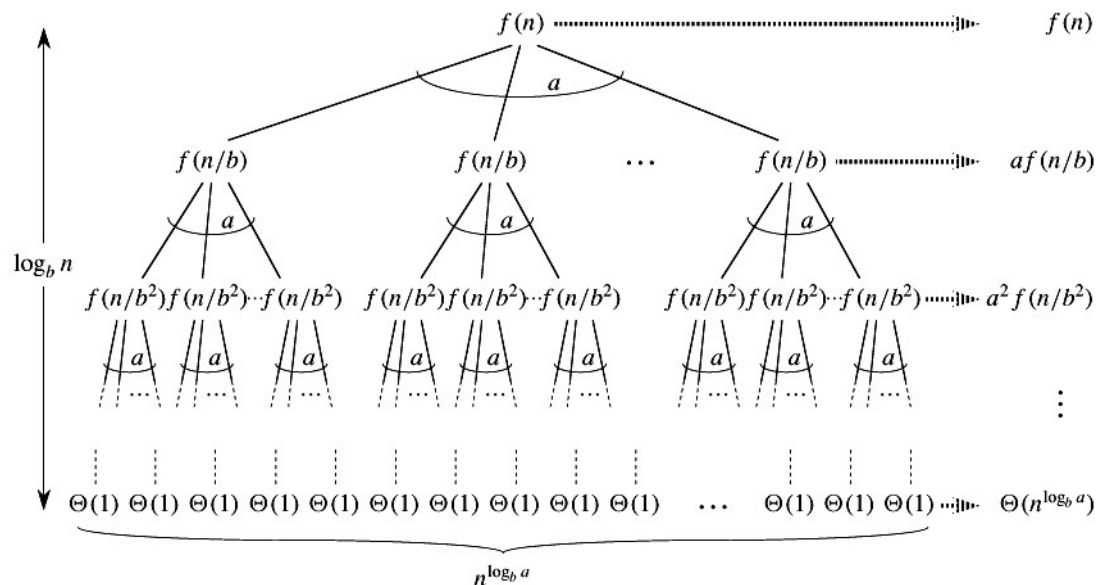
1. If  $f(n) = \Theta(n^c)$  where  $c < \log_b a$  then  $T(n) = \Theta(n^{\log_b a})$

2. If  $f(n) = \Theta(n^c)$  where  $c = \log_b a$  then  $T(n) = \Theta(n^c)$

3. If  $f(n) = \Theta(n^c)$  where  $c > \log_b a$  then  $T(n) = \Theta(f(n))$

#### How does this work?

Master method is mainly derived from recurrence tree method. If we draw recurrence tree of  $T(n) = aT(n/b) + f(n)$ , we can see that the work done at root is  $f(n)$  and work done at all leaves is  $\Theta(n^c)$  where  $c$  is  $\log_b a$ . And the height of recurrence tree is  $\log_b n$



In recurrence tree method, we calculate total work done. If the work done at leaves is

# HP Chromebook 11



[google.com/chromebook](https://google.com/chromebook)

Everything you need in one laptop. Made with Google. Learn more.

JDBC to Informix

Free C# Code  
Generator

Html5 Tutorial

NuoDB: The  
Elastic SQL DB

Math Problem  
Solving

Digi-Key Official  
Site

## Recent Comments

**Aman** Hi, Why arent we checking for conditions...

[Write a C program to Delete a Tree.](#) · 29 minutes ago

kzs please provide solution for the problem...

[Backtracking | Set 2 \(Rat in a Maze\)](#) · 32 minutes ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...

[Root to leaf path sum equal to a given number](#) · 57 minutes ago

**GOPI GOPINATH** @admin Highlight this sentence "We can easily...

[Count trailing zeroes in factorial of a number](#) · 59 minutes ago

**newCoder3006** If the array contains negative numbers also. We...

[Find subarray with given sum](#) · 1 hour ago

**newCoder3006** Code without using while loop. We can do it...

[Find subarray with given sum](#) · 1 hour ago

AdChoice

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 same, then our result becomes height multiplied by work done at any level (Case 2). If work done at root is asymptotically more, then our result becomes work done at root (Case 3).

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

**Merge Sort:**  $T(n) = 2T(n/2) + \Theta(n)$ . It falls in case 2 as  $c$  is 1 and  $\log_b a$  is also 1. So the solution is  $\Theta(n \log n)$

**Binary Search:**  $T(n) = T(n/2) + \Theta(1)$ . It also falls in case 2 as  $c$  is 0 and  $\log_b a$  is also 0. So the solution is  $\Theta(\log n)$

### Notes:

1) It is not necessary that a recurrence of the form  $T(n) = aT(n/b) + f(n)$  can be solved using Master Theorem. The given three cases have some gaps between them. For example, the recurrence  $T(n) = 2T(n/2) + n/\log n$  cannot be solved using master method.

2) Case 2 can be extended for  $f(n) = \Theta(n^c \log^k n)$ .

If  $f(n) = \Theta(n^c \log^k n)$  for some constant  $k \geq 0$  and  $c = \log_b a$ , then  $T(n) = \Theta(n^c \log^{k+1} n)$

[Practice Problems and Solutions on Master Theorem.](#)

### References:

[http://en.wikipedia.org/wiki/Master\\_theorem](http://en.wikipedia.org/wiki/Master_theorem)

[MIT Video Lecture on Asymptotic Notation | Recurrences | Substitution, Master Method](#)  
[Introduction to Algorithms 3rd Edition by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

# JDBC to Informix

 [progress.com/Informix](https://progress.com/Informix)

Supports Latest Data Connections  
J2EE Certified. Download Eval Now!




Advertisements 

[▶ Java Algorithm](#)

[▶ C Algorithm](#)


[▶ Solving](#)

AdChoices 

[▶ Log Analysis](#)

[▶ CN Analysis](#)

[▶ It Analysis](#)

AdChoices 

[▶ It Analysis](#)

[▶ Analysis of Data](#)

[▶ C Code Analysis](#)

## Related Tpoics:

- [Backtracking | Set 8 \(Solving Cryptarithmic Puzzles\)](#)
- [Tail Recursion](#)
- [Find if two rectangles overlap](#)
- [Print all possible paths from top left to bottom right of a mXn matrix](#)
- [Generate all unique partitions of an integer](#)
- [Russian Peasant Multiplication](#)
- [Closest Pair of Points | O\(nlogn\) Implementation](#)
- [Find the maximum distance covered using n bikes](#)



6



Tweet

3



0

Writing code in comment? Please use [ideone.com](https://ideone.com) and share the link here.

6 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion



with the recurrence...



**manish** · a day ago

" $T(n) = aT(n/b) + f(n)$  where  $a \geq 1$  and  $b > 1$ "

also  $a$  and  $b$  should be constants, otherwise master theorem not applied.

e.g.  $T(n) = 2^n T(n/2) + n^n$ .. what will solution in this case?

^ | v ·



**xxmajia** · 3 months ago

Regarding "Binary Search:  $T(n) = T(n/2) + .$  It also falls in case 2 as  $c$  is 0 and INCORRECT.

1) Binary search should be  $T(N) = T(N/2) + O(1)$

2) And  $T(N) = T(N/2) + O(N)$ , you can image its quick select, which would be  $n$

^ | v ·



**GeeksforGeeks** Mod → xxmajia · 3 months ago

Thanks for pointing this out. We have corrected the recurrence for Bin:

^ | v ·



**Guest** · 3 months ago

In master's method if  $f(n)$  contains some term of  $\log n$  the is it possible to solve

^ | v ·



**GeeksforGeeks** Mod → Guest · 3 months ago

Prateek, the case 2 can be extended to handle Log in multiplication. W this in notes as point 2.

^ | v ·



**Guest** → GeeksforGeeks · 3 months ago

But a/c CRLF if  $f(n)$  contains any logarithmic function then mast

^ | v ·



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

**Contact Us!**

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team