Difficulty Level : Easy • Last Updated : 23 Sep, 2020

Master Theorem is used to determine running time of algorithms (divide and conquer algorithms) in terms of asymptotic notations.
Consider a problem that be solved using recursion.

```
function f(input x size n)
if(n < k)
solve x directly and return
else
divide x into a subproblems of size n/b
call f recursively to solve each subproblem
Combine the results of all sub-problems
```

The above algorithm divides the problem into **a** subproblems, each of size n/b and solve them recursively to compute the problem and the extra work done for problem is given by f(n), i.e., the time to create the subproblems and combine their results in the above procedure.

So, according to master theorem the runtime of the above algorithm can be expressed as:

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

where n = size of the problem
a = number of subproblems in the recursion and a >= 1
n/b = size of each subproblem
f(n) = cost of work done outside the recursive calls like dividing into subproblems and cost of combining them to get the solution.

Not all recurrence relations can be solved with the use of the master theorem i.e. if

- T(n) is not monotone, ex: T(n) = sin n
- f(n) is not a polynomial, ex: T(n) = 2T(n/2) + 2^n

This theorem is an advance version of master theorem that can be used to determine running time of divide and conquer algorithms if the recurrence is of the following form :-

$$T(n) = aT(n/b) + \theta(n^k \log^p n)$$

where n = size of the problem
a = number of subproblems in the recursion and a >= 1
n/b = size of each subproblem
b > 1, k >= 0 and p is a real number.

Then,

1. if a > b^k, then T(n) = $\theta(n^{\log_b a})$
2. if a = b^k, then
   (a) if p > -1, then T(n) = $\theta(n^{\log_b a} \log^{p+1} n)$
   (b) if p = -1, then T(n) = $\theta(n^{\log_b a} \log\log n)$
   (c) if p < -1, then T(n) = $\theta(n^{\log_b a})$
3. if a < b^k, then
   (a) if p >= 0, then T(n) = $\theta(n^k \log^p n)$
   (b) if p < 0, then T(n) = $\theta(n^k)$

Time Complexity Analysis –

- **Example-1: Binary Search –** T(n) = T(n/2) + O(1)
  a = 1, b = 2, k = 0 and p = 0
  b^k = 1. So, a = b^k and p > -1 [Case 2.(a)]
  T(n) = $\theta(n^{\log_b a} \log^{p+1} n)$
  T(n) = $\theta(\log n)$
- **Example-2: Merge Sort –** T(n) = 2T(n/2) + O(n)
  a = 2, b = 2, k = 1, p = 0
  b^k = 2. So, a = b^k and p > -1 [Case 2.(a)]
  T(n) = $\theta(n^{\log_b a} \log^{p+1} n)$
  T(n) = $\theta(n \log n)$
- **Example-3:** T(n) = 3T(n/2) + n^2
  a = 3, b = 2, k = 2, p = 0
  b^k = 4. So, a < b^k and p = 0 [Case 3.(a)]
  T(n) = $\theta(n^k \log^p n)$
  T(n) = $\theta(n^2)$

- **Example-4:** T(n) = 3T(n/2) + $\log^2 n$
  a = 3, b = 2, k = 0, p = 2
  b^k = 1. So, a > b^k [Case 1]
  T(n) = $\theta(n^{\log_b a})$
  T(n) = $\theta(n^{\log_2 3})$

- **Example-5:** T(n) = 2T(n/2) + n$\log^2 n$
  a = 2, b = 2, k = 1, p = 2
  b^k = 2. So, a = b^k [Case 2.(a)]
  T(n) = $\theta(n^{\log_b a} \log^{p+1} n)$
  T(n) = $\theta(n^{\log_2 2} \log^3 n)$
  T(n) = $\theta(n \log^3 n)$

- **Example-6:** T(n) = 2^nT(n/2) + n^n
  This recurrence can't be solved using above method since function is not of form T(n) = aT(n/b) + $\theta(n^k \log^p n)$

GATE Practice questions –

- GATE-CS-2017 (Set 2) | Question 56
- GATE IT 2008 | Question 42
- GATE CS 2009 | Question 35

Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the **DSA Self Paced Course** at a student-friendly price and become industry ready.

RECOMMENDED ARTICLES

Page : 1 2 3

01 Master Theorem For Subtract and Conquer Recurrences
12, Jul 17

02 Regularity condition in the master theorem.
06, Oct 17

03 Search in a Row-wise and Column-wise Sorted 2D Array using Divide and Conquer algorithm
27, Apr 14

04 Karatsuba algorithm for fast multiplication using Divide and Conquer algorithm
18, Apr 13

05 Closest Pair of Points using Divide and Conquer algorithm
28, Nov 12

06 Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
29, Mar 14

07 The Skyline Problem using Divide and Conquer algorithm
08, Mar 15

08 Longest Common Prefix using Divide and Conquer Algorithm
22, Jun 16

Previous
Making zero array by decrementing pairs of adjacent

Next
Address Resolution in DNS (Domain Name Server)

Article Contributed By :
Ankit87
@Ankit87

Improved By : Niranjan_Hegde, nasveeda111, slsuryawanshi4

Article Tags : Analysis , Divide and Conquer , GATE CS , Technical Scripter

Practice Tags : Divide and Conquer

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org