Survey on Algorithms' Analysis by Master method

Yogini C. Jani[1]
[1] Computer Engineering Department
[1] Vadodara Institute of Engineering, Kotambi, Vadodara, India

Abstract- *In algorithm's analysis, there are basically two types 1) analysis according to space complexity and 2) analysis according to time complexity. The master theorem provides an asymptotic analysis using Big O notation for divide-and-conquer recurrences. Space complexity means during execution of program the amount of space required for program in memory, which is denoted by S (p). Time complexity is amount time required to execute program, which is denoted by T (n).time complexity provides asymptotic notations, which means best case, average case and worst case.*

Keywords- Big O notation, Time complexity, Space complexity, best case, average case, worst case.

I. INTRODUCTION

In computer science, the time complexity of an algorithm defines the amount of time taken by an algorithm to run as a function of the length of the string representing the input. The time complexity of an algorithm is commonly expressed by big O notation, which excludes coefficients and lower order terms. By using asymptotic notation the input size goes to infinity. For example, if the time required by an algorithm on all inputs of size n is at most $17n^3+n^2+13$ for any n, bigger than some n0, the asymptotic time complexity is O (n3). Time complexity is commonly counted by counting the number of elementary operations performed by the algorithm, where an elementary operation takes a fixed amount of time to perform.

II. ANALYSIS OF ALGORITHMS

To analyses an algorithm, while doing analysis there are functions, different types of loops and mathematical methods. Depends upon the behavior of an algorithm time complexity can be measured. While analysis of loop structure, we found recurrence relation of loop in recursive manner. Generally merge sort, quick sort, binary search, are having recursive behavior.

There are mainly three ways to solve recurrence.1) using substitution method 2) recurrence tree method and 3) master method. By comparing all of three methods, master method has direct solution. Some of the algorithms are recursive in nature. When we analyze them, time complexity is relatively matched with recurrence. 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 parts and recursively repeat the process for the two them.

Finally we merge the results with function combine. Time complexity of Merge Sort can be written as T (n) = 2T (n/2) + cn.

III. WORKING OF MASTER MEWTHOD

The master method works only for some types of recurrences. The format for this type of recurrence is as below.

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

In this formula we have three cases for solving recurrence.

1. If f (n) = Θ (nc) where c < Logba then T (n) = Θ (nLogba)
2. If f (n) = Θ (nc) where c = Logba then T (n) = Θ (ncLog n)
3. If f (n) = Θ (nc) where c > Logba then T (n) = Θ (f(n))

Now we need to check how this formula is going to work on problem for recurrence.

Basically master method is working on concept of tree recurrence. It is not necessary that a recurrence of the form T (n) = a T (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 cannot be solved using master method. This method has some limitations. Logarithmic structure cannot be solved by master theorem. For example solve the recursive equation

T (n) = 16T (n/4) + n

For this equation a = 16, b = 4, and f(n) = n. Initially this equation would represent an algorithm that divides the original inputs into fifteen groups each consisting of a forth of

the elements and takes linear time to combine the results. By computing above equation,

$$n^{\log_b^a} = n^{\log_4^{16}} = n^2$$

This is satisfied case 1 of this given formula. So solution is

$$T(n) = \Theta(n^{\log_b^a}) = \Theta\, n^2$$

IV. CONCLUSION

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

REFERENCES

[1] www.geeksforgeeks.org/analysis-algorithm-set-4-master-method-solving-recurrences

[2] https://webcache.googleusercontent.com/search?q=cache:YvY7kseY3kJ:https://www.cs.cornell.edu/courses/cs3110/2012sp/lectures/lec20master/lec20.html+&cd=14&hl=en&ct=clnk&gl=in

[3] techieme.in/solving-recurrences-master-method/

[4] www.cs.ucdavis.edu/~gusfield/cs222f07/mastermethod.pdf

[5] https://www.tutorialspoint.com › ... › DAA - Methodology of Analysis

[6] JD Ullman, AV Aho, JE Hopcroft - Addison-Wesley, Reading, 1974 - kawef.org