Analysis of Algorithms

Amypo Technologies Pvt Ltd

Agenda

- Substitution methods
- Master Theorem

Substitution Method

- The substitution method is a kind of method in which a guess for the solution is made.
- There are two types of substitution -
- Forward substitution
- Backward substitution.

Forward substitution

- This method makes use of an initial condition in the initial term and value for the next term is generated.
- This process is continued until some formula is guessed.
- Thus in this kind of substitution method, we use recurrence equations to generate the few terms.

Backward Substitution Method

- In this method backward values are substituted recursively in order to derive some formula.

Tree Method:

- The recurrence relation can also be solved using tree method.
- In this method, a recursion tree is built in which each node represents the cost of a single subproblem in the form of recursive function invocations.
- Then we sum up the cost at each level to determine the overall cost.
- Thus recursion tree helps us to make a good guess of the time complexity.
- Let us understand this method with the help of some examples.

Master's Method

- We can solve recurrence relation using a formula denoted by Master's method.

- T(n) = aT (n/b) + Fn)   where n 2 d and d is some constant.

- Then the Master theorem can be stated for efficiency analysis as -

Master Method

- The Master Method is used for solving the following types of recurrence
- T (n) = a T+ f (n) with a≥1 and b≥1 be constant & f(n) be a function and can be interpreted as
- Let T (n) is defined on non-negative integers by the recurrence.

  T (n) = a T(n/b)+ f (n)
- n is the size of the problem.
- a is the number of subproblems in the recursion.
- n/b is the size of each subproblem. (Here it is assumed that all subproblems are essentially the same size.)
- f (n) is the sum of the work done outside the recursive calls, which includes the sum of dividing the problem and the sum of combining the solutions to the subproblems.

Master Theorem

$$T(n)=\begin{cases}\Theta\left(n^{\log_b a}\right) & f(n)=O\left(n^{\log_b a-\varepsilon}\right) \\ \Theta\left(n^{\log_b a}\log n\right) & f(n)=\Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n)=\Omega\left(n^{\log_b a+\varepsilon}\right)\text{AND} \\ & af(n/b)<cf(n)\ \text{for large}\ n\end{cases}\Bigg\}\ \begin{matrix}\varepsilon>0 \\ c<1\end{matrix}$$

- **Case1:** If f (n) = O(n to the power of log b a- ε) for some constant ε >0, then it follows that:
- T (n) = Θ (n to the power of log b a)

Master Theorem

Example:

- $T(n) = 8\,T(n/2) + 1000n^2$ apply master theorem on it.

Solution:

Compare $T(n) = 8\,T(n/2)+1000n^2$ with $T(n) = a\,T(n/b)+f(n)$ with $a>=1$ and $b>1$

$a = 8$, $b=2$, $f(n) = 1000\,n^2$, $\log_b a = \log_2 8 = 3$

Put all the values in: f (n) = O(n to the power of log b a- ε)

$1000\,n^2 = O(n^{3-\varepsilon})$ If we choose $\varepsilon=1$, we get: $1000\,n^2 = O(n^{3-1}) = O(n^2)$

Case 2: If it is true, for some constant $k \geq 0$ that:

- F (n) = Θ (n to the power of log b a log to the power of k n)then it follows that: T (n) = Θ(n to the power of log b a log to the power of k+1 n)

Example:

- T (n) = 2 T(n/2) + 10n, solve the recurrence by using the master method.
- As compare the given problem with

  T (n) = a T(n/2) + f(n) with a>=1 and b>1

  a = 2, b=2, k=0, f (n) = 10n, $\log_b a = \log_2 2 = 1$
- Put all the values in f (n) =Θ (n to the power of log b a log to the power of k n) ,

  we will get 10n = Θ (n^1) = Θ (n) which is true.

Therefore: T (n) = Θ Θ(n to the power of log b a log to the power of k+1 n) = Θ (n log n)

Case 3: If it is true f(n) = Ω (n to the power of log b a+Ɛ) for some constant ε >0 and it also true that: a f(n/b)<=cf(n) for some constant c<1 for large value of n ,then :

$$T(n) = \Theta((f(n))$$

Example: Solve the recurrence relation:

$T(n) = 2\,T(n/2)+n^2$

Solution:

- Compare the given problem with

  $T(n) = a\,T(n/2) + f(n)$ with a>=1 and b>1

$a= 2$, $b =2$, $f(n) = n^2$, $\log_b a = \log_2 2 =1$

Put all the values in f (n) = Ω (n to the power of logb a+$^{\varepsilon}$).. (Eq. 1)

If we insert all the value in (Eq.1),

we will get $n^2 = \Omega(n^{1+\varepsilon})$ put $\varepsilon =1$, then the equality will hold. $n^2 = \Omega(n^{1+1}) = \Omega(n^2)$