Recurrence Relation

A recurrence is an equation or inequality that describes a function in terms of its values on smaller inputs. To solve a Recurrence Relation means to obtain a function defined on the natural numbers that satisfy the recurrence.

For Example, the Worst Case Running Time T(n) of the MERGE SORT Procedures is described by the recurrence.

$$T(n) = \theta(1) \text{ if } n=1$$
$$2T\left(\frac{n}{2}\right) + \theta(n) \text{ if } n>1$$

There are four methods for solving Recurrence:

1. Substitution Method
2. Iteration Method
3. Recursion Tree Method
4. Master Method

---

1. Substitution Method:

The Substitution Method Consists of two main steps:

1. Guess the Solution.
2. Use the mathematical induction to find the boundary condition and shows that the guess is correct.

For Example1 Solve the equation by Substitution Method.

$$T(n) = T\left(\frac{n}{2}\right) + n$$

We have to show that it is asymptotically bound by O (log n).

Solution:

```
For T (n) = O (log n)
```

We have to show that for some constant c

1. T (n) ≤c logn.

Put this in given Recurrence Equation.

$$T(n) \leq c \log\left(\frac{n}{2}\right) + 1$$
$$\leq c \log\left(\frac{n}{2}\right) + 1 = c \log n - c\log_2 2 + 1$$
$$\leq c \log n \text{ for } c \geq 1$$

Thus **T (n) =O logn**.

Example2 Consider the Recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n \quad n>1$$

Find an Asymptotic bound on T.

Solution:

```
We guess the solution is O (n (logn)).Thus for constant 'c'.
 T (n) ≤c n logn
Put this in given Recurrence Equation.
Now,
```
$$T(n) \leq 2c\left(\frac{n}{2}\right)\log\left(\frac{n}{2}\right)+n$$
$$\leq cn\log n - cn\log 2 + n$$
$$= cn \log n - n(c\log 2 - 1)$$

```
      ≤cn logn for (c≥1)
Thus  T (n) = O (n logn) .
```

2. Iteration Methods

It means to expand the recurrence and express it as a summation of terms of n and initial condition.

Example1: Consider the Recurrence

1. T (n) = 1 if n=1
2.   = 2T (n-1) if n>1

Solution:

```
T (n) = 2T (n-1)
      = 2[2T (n-2)] = 2²T (n-2)
      = 4[2T (n-3)] = 2³T (n-3)
      = 8[2T (n-4)] = 2⁴T (n-4)    (Eq.1)

Repeat the procedure for i times

T (n) = 2ⁱ T (n-i)
Put n-i=1 or i= n-1 in    (Eq.1)
T (n) = 2ⁿ⁻¹ T (1)
      = 2ⁿ⁻¹ .1    {T (1) =1 .....given}
      = 2ⁿ⁻¹
```

Example2: Consider the Recurrence

1. T (n) = T (n-1) +1 and T (1) = θ (1).

Solution:

```
 T (n) = T (n-1) +1
       = (T (n-2) +1) +1 = (T (n-3) +1) +1+1
       = T (n-4) +4 = T (n-5) +1+4
       = T (n-5) +5= T (n-k) + k
Where k = n-1
   T (n-k) = T (1) = θ (1)
   T (n) = θ (1) + (n-1) = 1+n-1=n= θ (n).
```

Recursion Tree Method

1. Recursion Tree Method is a pictorial representation of an iteration method which is in the form of a tree where at each level nodes are expanded.

2. In general, we consider the second term in recurrence as root.

3. It is useful when the divide & Conquer algorithm is used.

4. It is sometimes difficult to come up with a good guess. In Recursion tree, each root and child represents the cost of a single subproblem.

5. We sum the costs within each of the levels of the tree to obtain a set of pre-level costs and then sum all pre-level costs to determine the total cost of all levels of the recursion.

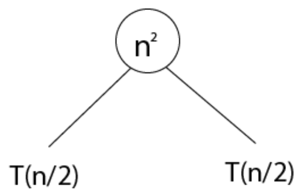
6. A Recursion Tree is best used to generate a good guess, which can be verified by the Substitution Method.

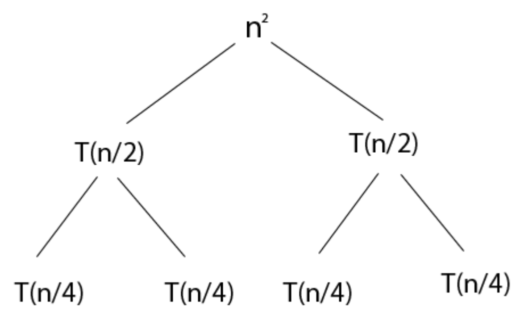
Example 1

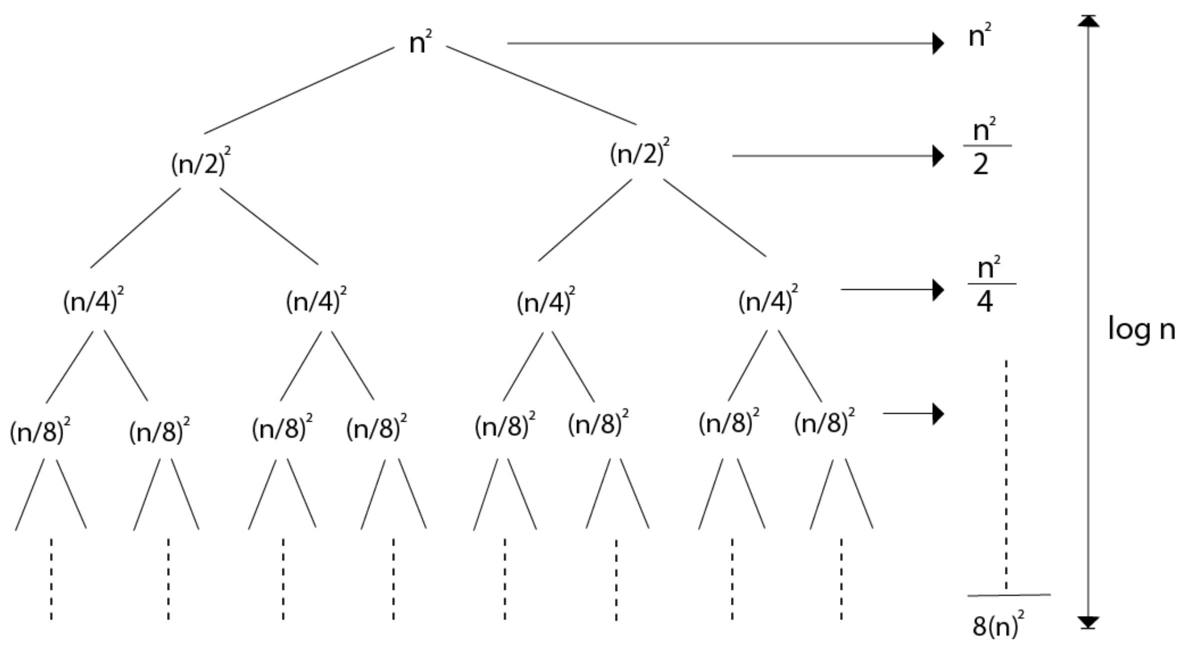
Consider $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

We have to obtain the asymptotic bound using recursion tree method.

Solution: The Recursion tree for the above recurrence is







$$T(n) = n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \ldots\ldots \log n \text{ times.}$$

$$\leq n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2^i}\right)$$

$$\leq n^2 \left(\frac{1}{1-\frac{1}{2}}\right) \leq 2n^2$$

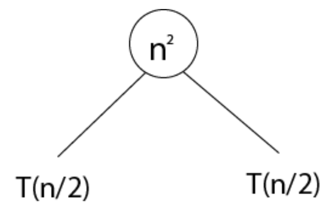
$$\mathbf{T(n) = \theta} \boldsymbol{n^2}$$

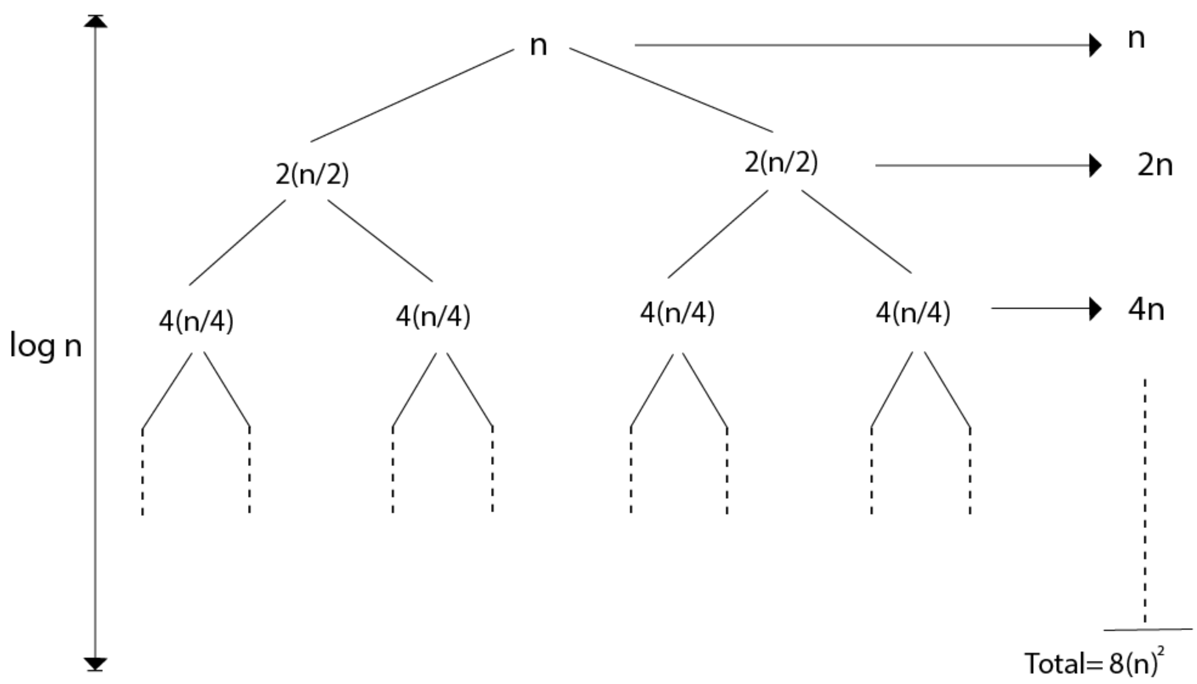
Example 2: Consider the following recurrence

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

Obtain the asymptotic bound using recursion tree method.

Solution: The recursion trees for the above recurrence





We have $n + 2n + 4n + \ldots\ldots \log_2 n$ times

$$= n\,(1 + 2 + 4 + \ldots\ldots \log_2 n \text{ times})$$

$$= n\frac{(2\ \log_2 n - 1)}{(2-1)} = \frac{n(n-1)}{1} = n^2 - n = \theta(n^2)$$

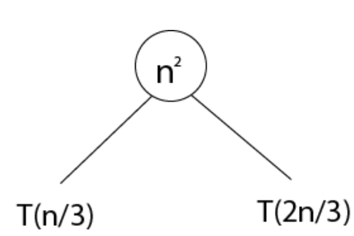
$$\mathbf{T\ (n) = \theta(n^2)}$$

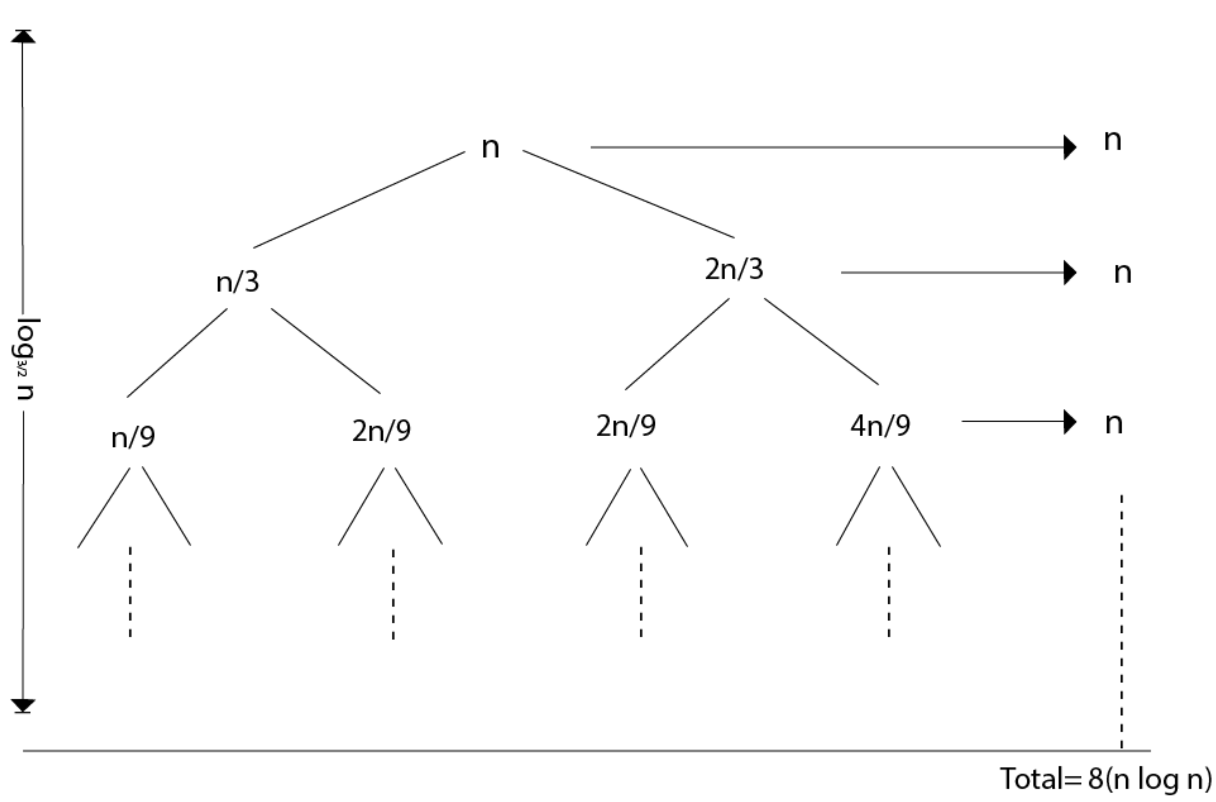
Example 3: Consider the following recurrence

$$T\ (n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$

Obtain the asymptotic bound using recursion tree method.

Solution: The given Recurrence has the following recursion tree





When we add the values across the levels of the recursion trees, we get a value of n for every level. The longest path from the root to leaf is

$$n \longrightarrow \frac{2}{3}n \longrightarrow \left(\frac{2}{3}\right) n \longrightarrow \ldots 1$$

Since $\left(\frac{2}{3}\right)$ n=1 when i=$\log\frac{3}{2}$ n.

Thus the height of the tree is $\log\frac{3}{2}$ n.

$$T(n) = n + n + n + \ldots\ldots + \log\frac{3}{2} n \text{ times.} = \boldsymbol{\theta}(\mathbf{n \log n})$$

Master Method

The Master Method is used for solving the following types of recurrence

T (n) = a T $\left(\frac{n}{b}\right)$ + f (n) with a≥1 and b≥1 be constant & f(n) be a function and $\frac{n}{b}$ can be interpreted as

Let T (n) is defined on non-negative integers by the recurrence.

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

In the function to the analysis of a recursive algorithm, the constants and function take on the following significance:

- 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.
- It is not possible always bound the function according to the requirement, so we make three cases which will tell us what kind of bound we can apply on the function.

Master Theorem:

It is possible to complete an asymptotic tight bound in these three cases:

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

Case1: If f (n) = $O\left(n^{\log_b a - \varepsilon}\right)$ for some constant ε >0, then it follows that:

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

Example:

$T(n) = 8\,T\left(\frac{n}{2}\right) + 1000n^2$ 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 n², log_b a = log_2 8 = 3
 Put all the values in: f (n) = O(n^(log_b a−ε))
     1000 n² = O (n^(3-ε) )
     If we choose ε=1, we get: 1000 n² = O (n^(3-1)) = O (n²)
```

Since this equation holds, the first case of the master theorem applies to the given recurrence relation, thus resulting in the conclusion:

```
T (n) = Θ (n^(log_b a))
   Therefore: T (n) = Θ (n³)
```

Case 2: If it is true, for some constant k ≥ 0 that:

F (n) = Θ $\left(n^{\log_b a} \log^k n\right)$ then it follows that: T (n) = Θ $\left(n^{\log_b a} \log^{k+1} n\right)$

Example:

T (n) = 2 $T\left(\frac{n}{2}\right)+10n$, solve the recurrence by using the master method.

As compare the given problem with T (n) = a T $\left(\frac{n}{b}\right)+f(n)$ *with* $a \geq 1$ *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 $\left(n^{\log_b a} \log^k n\right)$, we will get

10n = Θ (n^1) = Θ (n) which is true.

Therefore: T (n) = Θ $\left(n^{\log_b a} \log^{k+1} n\right)$

= Θ (n log n)

Case 3: If it is true f(n) = Ω $\left(n^{\log_b a+\varepsilon}\right)$ for some constant ε >0 and it also true that: a f $\left(\frac{n}{b}\right) \leq cf(n)$ for some constant c<1 for large value of n ,then :

1. T (n) = Θ((f (n))

Example: Solve the recurrence relation:

T (n) = 2 $T\left(\frac{n}{2}\right)+n^2$

Solution:

Compare the given problem with T (n) = a T $\left(\frac{n}{b}\right)+f(n)$ *with* $a \geq 1$ *and* $b>1$

a= 2, b =2, f (n) = n², $\log_b a = \log_2 2 = 1$

Put all the values in f (n) = Ω $\left(n^{\log_b a+\varepsilon}\right)$ ..... (Eq. 1)

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

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

$n^2 = \Omega(n^{1+1}) = \Omega(n^2)$

Now we will also check the second condition:

$$2\left(\frac{n}{2}\right)^2 \leq cn^2 \Rightarrow \frac{1}{2}n^2 \leq cn^2$$

If we will choose c =1/2, it is true:

$$\frac{1}{2}n^2 \leq \frac{1}{2}n^2 \quad \forall\ n \geq 1$$

So it follows: T (n) = Θ ((f (n))

$T(n) = \Theta(n^2)$