Text:

Solving Recurrences (Dividing Functions)

In this section, we will look at some recurrences of the form

T (n) = a T (n / b) + (n) where *a >1, b > 1,* and *(n)* is a given function.

Since the sub problem divides the initial input length, it is called as dividing function. As mentioned before, we are studying this class of recurrences separately because it will help us understand and memorize the master's theorem effectively. Let us see and solve some recurrences of such type using iteration and recursion tree methods.

Solve using Iteration method

Solving the recurrences using iteration method.

Example 9.1:

```
void test (int n)
{
        If(n>1)
        {
                printf("%d", n);          ----------> 1
                test(n/2);                ----------> T(n/2)
        }                                             ---------
}                                                     T(n/2) + 1
```

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + 1 & n > 1 \end{cases}$$

T(n) = *T(n/2) + 1* ---------- *(1)*

= *[T(n/2²)+ 1] + 1* *Using equation (1) for n/2*

= *T(n/2²)+ 2*

= *[T(n/2³)+ 1] + 2* *Using equation (1) for n/2²*

= *T(n/2³)+ 3*

.

.

.

K times

T(n) = *T(n/2ᵏ)+ k*

Here, stopping condition is *T(1)*

=> *n/2ᵏ = 1*

2ᵏ = n

k = log n

=> *T(n)* = *T(1) + log n*

$= 1 + \log n$

$\Rightarrow \quad T(n) = O(\log n)$

Example 9.2:

```
void disp (int n)
{
    If(n>2)
    {
        printf("%d", n);          ----------> 1
        test(√n);                 ----------> T(√n)
    }                                          ----------
}                                              T(√n) + 1
```

$$T(n) = \begin{cases} 1 & n = 2 \\ T(\sqrt{n}) + 1 & n > 2 \end{cases}$$

$T(n) = T(\sqrt{n}) + 1$

$T(n) = T(n^{1/2}) + 1$ ---------- (1)

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

$= T(n^{1/2^2}) + 2$

$= T\left[\left\{\left(n^{1/2^2}\right)\right\}^{1/2} + 1\right] + 2$

$T(n) = T(n^{1/2^3}) + 3$

.

.

.

k times

$T(n) = T(n^{1/2^k}) + k$ ----------(2)

Assume, $n = 2^m$ ----------(3)

$\Rightarrow m = \log n$ ----------(4)

Using (3) in (4), we get

$T(2^m) = T((2^m)^{1/2^k}) + k$

$T(2^m) = T(2^{m/2^k}) + k$

Here, stopping condition is T(2)

$\Rightarrow \quad 2^{m/2^k} = 2$

$m/2^k = 1$

$2^k = m$

$\Rightarrow \quad k = \log m$

But, $m = \log n$ using (4)

$\Rightarrow \quad k = \log \log n$

$T(n) = T(2) + \log \log n$

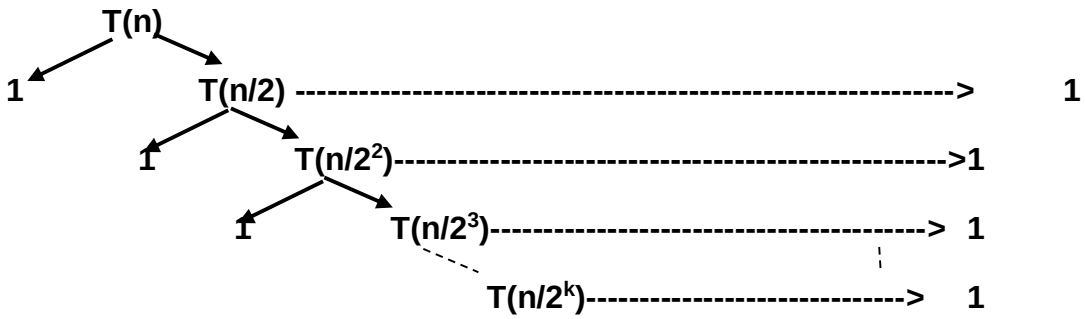
$= 1 + \log \log n$

$\Rightarrow \quad T(n) = O(\log \log n)$

Solve using Recursion Tree method

Example 9.3:

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + 1 & n > 1 \end{cases}$$



T(n) = 1 + 1 + 1 + 1+ … k times

T(n) = k

Here, stopping condition is T(1)

=> $n/2^k = 1$ or $n = 2^k$

=> k = log n

T(n) = log n

=> T(n) = O(log n)