Substitution Method

The substitution method for solving recurrences consists of two steps:

① Guess the form of the solution

② Use the mathematical induction to find constants in the form & show that the solution works.

- The inductive hypothesis is applied to smaller values similar like recursive calls bring us closer to the base case.

- The substitution method is powerful to establish lower or upper bound on a recurrence.

- All the problems of recurrence relation can be solved using substitution method which is not possible using Master's method.
- Substitution method always gives correct answer but the disadvantage is that the mathematical calculations are more compared to recursive tree & master's method.

① The substitution method

The recurrence relation for Binary Search is

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

$$T(n) = T(n/2) + C$$

The funⁿ is decreasing by division i.e. ~~n2~~ n/2

$$T(n) = T(n/2) + C \quad \text{—} \ (1)$$

$$T(n/2) = T(n/4) + C \quad \text{—} \ (2)$$

$$T(n/4) = T(n/8) + C \quad \text{—} \ (3)$$

Camlin

Let's back substitute the values of equation 2 to equation 1

$$T(n) = T(n/4) + C + C$$

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

Now again put values of equation 3 in this equation

$$= T\left(\frac{n}{8}\right) + C + 2C$$

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

(Again put values of equation 4 in this equation)

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

So from first iteration we have $T\left(\frac{n}{2^2}\right) + 2C$

from second iteration we have $T\left(\frac{n}{2^3}\right) + 3C$

from third iteration we have $T\left(\frac{n}{2^4}\right) + 4C$

If we go till k iterations we get

$$T\left(\frac{n}{2^k}\right) + k \cdot C$$

we have observed that 2^{x} & $\cancel{x \cdot c}$ for 2^{x} we have $x \cdot c$ constant statements.

So, $T\left(\frac{n}{2^k}\right) + k \cdot C$

Now to make $T\left(\frac{n}{2^k}\right)$ as 1 or to terminate the execution for the function
Let's take,

$$n = 2^k$$

So we have

$$T\left(\frac{n}{n}\right) + k \cdot C$$

$$T(1) + k \cdot C$$

As $T(1)$ is constant time

$$1 + k \cdot C$$

But time complexity is to be specified in terms of n, so we need to find value of k.

$$n = 2^k$$

$$\log n = \log 2^k$$

$$\log n = k \cdot \log 2$$

We know that value of $\log 2 = 1$

So, $\log n = k \cdot 1$

i.e. $\boxed{k = \log n}$

Page :
Date :

Let's put value of k in our equation

$$1 + kc$$

$$1 + \log n \cdot c$$

As 1 & c are constant we can write it as $O(\log_2 n)$

So the time complexity of given recurrence relation or time complexity Binary search algorithm is

$$\boxed{O(\log n)}$$