Backward Substitution:

In forward substitution, put $n = 0, 1, 2, \ldots$ in the given recurrence relation until, a pattern is seen

In backward substitution, do the reverse $n = n, n-1, n-2, \ldots$ / or $n = n, n/2, n/4 \ldots$ until a pattern is seen.

After a pattern is seen, make a guess

Consider the recurrence relation:

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + n, & \text{Otherwise} \end{cases} \quad \text{——— } (1)$$

Given $T(n)$, find $T(n/2)$ in (1)

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

back substitute value of $T(n/2)$ in $T(n)$ (1)

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

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

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

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

Apply boundary condition ie., $T(1) = 1$

$\therefore$ for $T(1) = 1$, $\frac{n}{2^k} = 1$ ——— (2)

Taking $\log_2$ on both sides

$$n = \log_2 n$$

The equation becomes

$$T(n) = 2^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \cdot n$$

$$= nT(1) + n\log_2 n$$

$$= n\log_2 n + n$$

The correctiens is verified using induction,

$n = 2, 4, 8, 16 \ldots \ldots$

✓ Verified $T(2) = 2\log_2 2 + 2$

$= 2(1) + 2 = 4$

× $T(3) = 3\log_2 3 + 3$

Not applicable

✓ Verified $T(4) = 4\log_2 4 + 4$

$= 4\log_2 2^2 + 4$

$= 8\log_2 2 + 4$

$= 8 + 4 = 12$

✓ Given

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

$T(2) = 2T(1) + 2$

$= 2(1) + 2$

$= 4$

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

$= 2$

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

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

$T(4) = 2T(2) + 4$

$= 2(4) + 4$

$= 8 + 4 = 12$

Solving Recurrence Relations: Substitution

Forward Substitution Method:

Step 1: Solve the recurrence relation for $n = 0, 1, 2, \dots$ until a pattern is obtained.

Step 2: Make a guess and predict the running time

Step 3: Verify the guess work using induction.

Consider the recurrence relation:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + 1, & \text{otherwise} \end{cases}$$

Calculate the running time for $n = 0, 1, 2$

n	$T(n)$
1	$T(1-1) + 1 = T(0) + 1$ as defined in $T(n)$, $T(n) = 1$, if $n = 1$
2	$T(2-1) + 1 = 1 + 1 = 2$
3	$T(3-1) + 1 = 1 + 1 + 1 = 3$
4	$T(4-1) + 1 = 1 + 1 + 1 + 1 = 4$

$\therefore$ when $n = k$, $T(n) = k$,

Hence, running time is

$$\boxed{T(n) = n}$$

To Verify by Induction,

$T(n) = n$, $T(1) = 1$, $T(2) = 2$, $T(3) = 3$, . . .

Example: 2

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T(n-1)+1, & \text{otherwise} \end{cases}$$

n	$T(n)$
1	1
2	$2.T(2-1)+1 = 2.1+1 = 3$
3	$2.T(3-1)+1 = 2.2.1+2.1+1 = 7$
4	$2.T(4-1)+1 = 2.2.2.1+2.2.1+2.1+1 = 15$
5	$2.T(5-1)+1 = 2.2.2.2.1+2.2.2.1+2.2.1+2.1+1 = 31$

To see a pattern, when value of n is k

$$T(k) = 2^{k-1} + 2^{k-2} + \cdots + 2^0$$

This is a geometric series, sum is calculated as

$$2^k - 1$$

[*Refer to Lecture materia Asymptotic Notation Review on Summat and functions]

The running time of algorithm is

$$T(n) = 2^n - 1 \quad \text{——— } (1)$$

The correctness of this running time can be verified for $n = 1, 2, 3$ in (1)