→ BINARY EXPONENTIATION : RECURSIVE METHOD 47

→ Why do we need binary exponentiation, we can simply use inbuilt POW( ) method?

→ The inbuilt POW( ) method calculates in double ~~and double~~, and double can give precision errors.

→ The other option is, we can use a FOR loop to compute ans in a variable but it will take O(n) Time.

But we can solve that problem in O(log(n)) time using Binary exponentiation

→ Suppose we have to find 2^{16}, we can achieve this like this:

$a = 2$, $b = 16$

$2^{16} \longrightarrow 2^8 \times 2^8$
$2^8 \longrightarrow 2^4 \times 2^4$
$2^4 \longrightarrow 2^2 \times 2^2$
$2^2 \longrightarrow 2^1 \times 2^1$

∴ In 4 steps we have got our ans and its T.C is O(log(b)).

→ Now for odd power:

$a = 3$, $b = 13$

$3^{13} \longrightarrow 3 \times 3^{12}$
$3^{12} \longrightarrow 3^6 \times 3^6$
$3^6 \longrightarrow 3^3 \times 3^3$
$3^3 \longrightarrow 3 \times 3^2$
$3^2 \longrightarrow 3^1 \times 3^1$
$3^1 \longrightarrow 3 \times 3^0$

∴ If b is even:

$F(a, b) \longrightarrow F(a, b/2) \times F(a, b/2)$

If b is odd :

$F(a, b) \longrightarrow a \times F(a, b/2) \times F(a, b/2)$

```
INT BINEXPRECUR(INT A, INT B)
{
    IF (B == 0)
    {
        RETURN 1;
    }

    LL RES = BINEXPRECUR(A, B/2);
    -> If odd
    IF (B & 1)
    {
        RETURN A * RES * RES;
    }
    ELSE
    {
        RETURN RES * RES;
    }
}
```