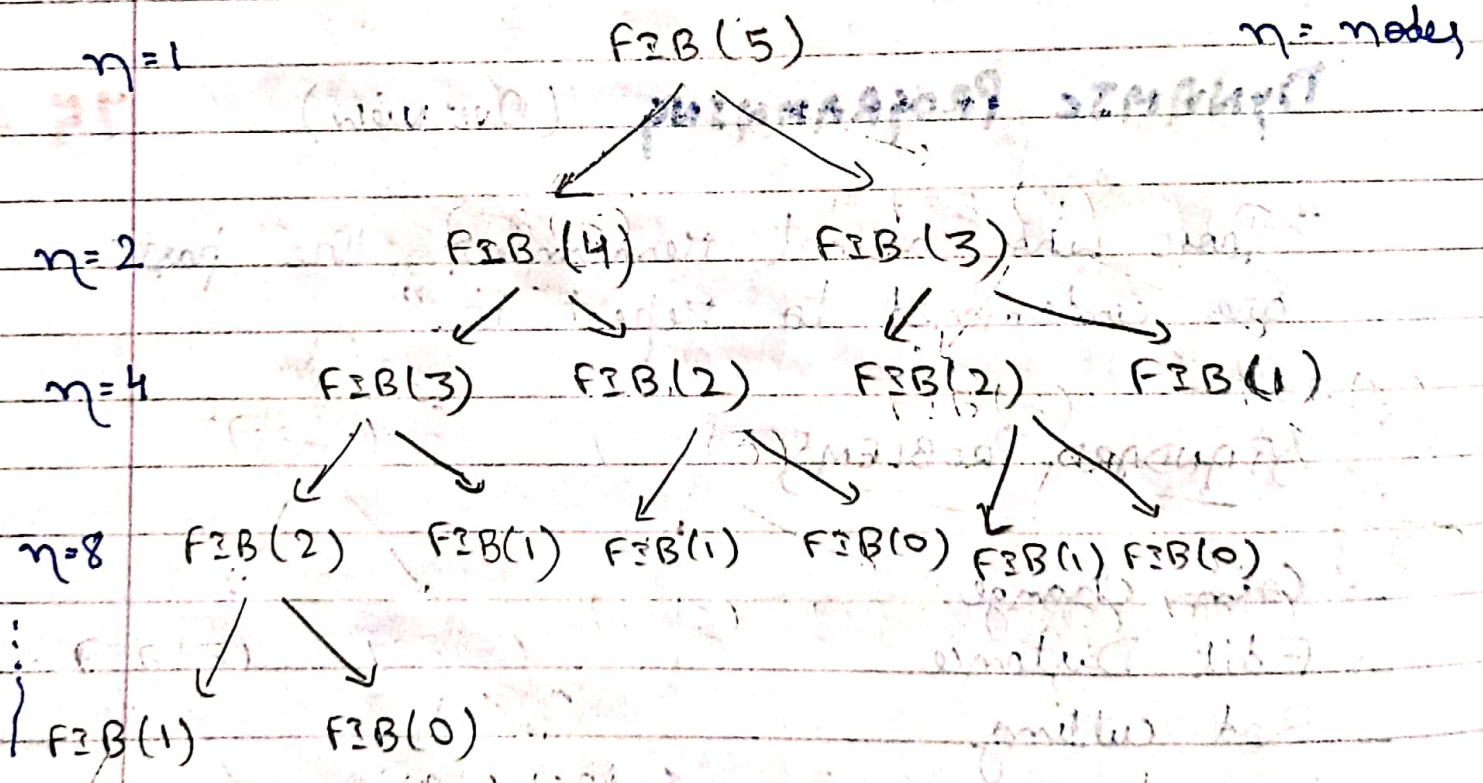
INTRODUCTION TO DYNAMIC PROGRAMMING...

FIBONACCI SERIES: 0 1 1 2 3 5 8

In the simple recursive function of Fibonacci series (FIB(n)).
FIB(n) gives n^{th} Fibonacci number.

Its Recursion Tree: For: FIB(5)



Here, we can see no. of calls in this Recursion tree is going exponentially

1, 2, 4, 8, ---- n

$S_n = 1(2^n - 1)$

$S_n = a\left(\frac{r^n - 1}{r - 1}\right)$

∴ Its Time complexity is: $O(2^n)$

∴ We can compute till $n = 20$, in normal recursive function of Fibonacci series, ~~or~~ ~~b~~ if we use n any more it will give TLE.

→ But with Dynamic Programming using memoisation technique, we will reduce T.C ~~to~~ to $O(n)$.

→ In this technique we try to remember what we have computed by past and use it for further computation, instead calculating the whole thing again.
eg: Let's suppose we are computing FIB(5), in its recursion tree we can see we are computing FIB(3) twice, but we will only compute it once and store it to use it further directly.
∴ With DP we can compute n upto 10^6

→ Top Down Approach: Bade problems ko chote mai todke use solve karna, usually with recursion.

→ Bottom Up Approach: Chote problems ko compute karte hue usse bdi problems solve karna.

Page No.
Date

CODE

```
CONST INT N = 1e5+10;
INT DP[N];

-> Top Down Approach
INT FIB (INT n)
{
    IF (n == 0) RETURN 0;
    IF (n == 1) RETURN 1;
    IF (DP[n] != -1) RETURN DP[n];

    -> memoise
    RETURN DP[n] = FIB(n-1) + FIB(n-2);
}

INT MAIN()
{
    MEMSET (DP, -1, SIZEOF (DP)); // memset
    function to store array with same
    default value here with -1
    INT n;
    CIN >> n;

    COUT << FIB (n) << "\n";
    RETURN 0;
}
```