COMPLEJIDAD ALGORITMO FIBONACCI RECURSIVO

El algoritmo fibonacci iterativo es como sigue, donde n es un numero de la serie 1,2,3…

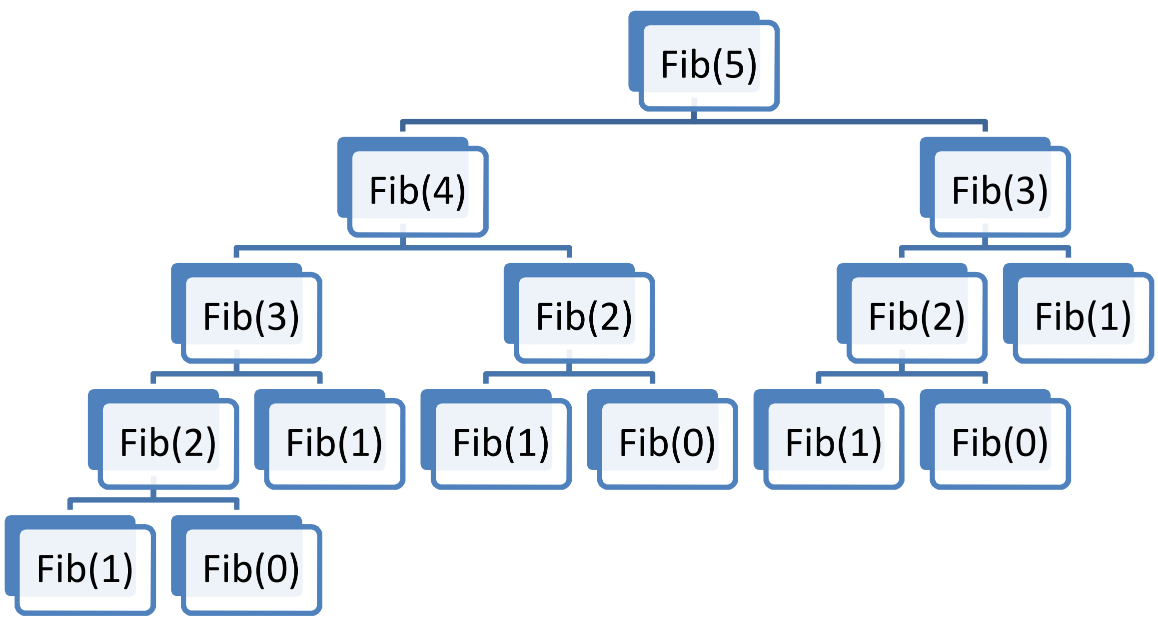
```
Funcion Fib(N: int): int;
{
        if N < 0 then
                error (‘ no valido’)
        case N of 0,1  :
                retornar N
        Else
                Retorno Fib(N-1) + Fib( N-2)
}
```

Es decir, para calcular un elemento Fib(N) llamamos a la misma función dos veces:

Fib(N) → Fib(N-1) y Fib(N-2)

Cada una de dichas funciones Fib(N-1) y Fib(N-2) se llama a si misma otras dos veces y así, etc. Esto dura hasta que se encuentra a Fib(1) o Fib(0) Es decir, crece exponencialmente. Lo demuestro:

El crecimiento exponencial anterior se parece al recorrido de una árbol binario completo (considerando que el algoritmo Fibonacci crece como un árbol binario completo aunque no sea completo como se ve en el gráfico)



Vamos a analizar el número de operaciones desde la perspectiva de Fib(5), donde las llamadas a Fib(4) y Fib(3) representan un número de operaciones que van incrementando a medida que el nivel de profundidad del recorrido avanza hasta llegar a las llamadas finales Fib(1) y Fib(0).

Nivel de Profundidad desde Fib(5)	N° de Operaciones Totales desde la perspectiva de Fib(5)
1 nivel → Fib(4) y Fib(3) representan una operación cada uno	2

2 nivel → Fib(4) y Fib(3) realizan 2 llamadas cada uno	2*2
3 nivel → Fib(4) y Fib(3) realizan 2*2 llamadas cada uno	2*2*2
4 nivel → Fib(4) y Fib(3) realizan 2*2*2 llamadas cada uno - en este punto se llega a Fib(1) y Fib(0).	2*2*2*2

En conclusión:
Siendo k el número de operaciones para la Funcion Fibonacci y N =5 el tamaño de la entrada, del ejemplo diremos:
El árbol Fibonacci al no ser completo => $k \leq 16 \rightarrow k \leq 2^{5-1}$, por tanto en general $k \leq 2^{N-1} \in O(2^N)$.