AULA 8 - ANÁLISE DA COMPLEXIDADE DE ALGORITMOS RECURSIVOS (NÚMERO DE SCHRÖDER)

- Implemente uma função recursiva para calcular o número de Schröder, definido pela seguinte relação de recorrência:

$$\text{Schröder}(n) = \begin{cases} 1, & \text{se } n = 0 \\ \text{Schröder}(n-1) + \sum_{i=0}^{n-1} \text{Schröder}(i) \times \text{Schröder}(n-i-1), & \text{se } n > 0 \end{cases}$$

Construa um programa para executar a função para sucessivos valores de n e que permita determinar experimentalmente a ordem de **complexidade das operações de multiplicação** do seu algoritmo. Efetue a análise empírica da complexidade construindo uma tabela com o número de multiplicações efetuadas para diferentes valores de n. Qual é a ordem de complexidade da função recursiva?

Uma forma de resolver problemas recursivos de maneira a evitar o cálculo repetido de valores, consiste em calcular os valores de baixo para cima, ou seja, de Schröder (0) para Schröder (n) e utilizar um *array* para manter os valores entretanto calculados. Este método designa-se por programação dinâmica e reduz o tempo de cálculo à custa da utilização de mais memória para armazenar valores intermédios.

- Usando a técnica de programação dinâmica, implemente uma função repetitiva alternativa e efetue a análise empírica da sua complexidade. Qual é a ordem de complexidade da função repetitiva?

- Faça a análise formal (no verso da folha) das implementações da função e confirme as ordens de complexidade obtidas experimentalmente.

N	Recursivo (N)	Nº de Multiplicações	Dinâmico (N)	Nº de Multiplicações
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

O(N)		

Na análise formal da solução recursiva – para simplificar a expressão recorrente – tenha em atenção que $\sum_{i=0}^{N-1} E(i) = \sum_{i=0}^{N-1} E(N-i-1)$**.**

NOME: **Nº MEC:**

NOME: **Nº MEC:**