AULA 7 - ANÁLISE DA COMPLEXIDADE DE ALGORITMOS RECURSIVOS

***** Entregue, num ficheiro ZIP, este guião preenchido e o código desenvolvido *****

Considere a seguinte relação de recorrência:

$$F(n) = \begin{cases} 1, \text{ se } n = 0 \text{ ou } n = 1 \text{ ou } n = 2 \\ F(n-1) + F(n-2) + \sum_{k=0}^{n-3} F(k) \times F(n-3-k), \text{ se } n > 2 \end{cases}$$

Função Recursiva

- Implemente uma **função recursiva** que use diretamente a relação de recorrência acima, **sem qualquer simplificação**.
- Construa um programa para executar essa função para **sucessivos valores de n** e que permita **contar o número total de multiplicações efetuadas** para cada valor de n.
- **Preencha a as primeiras colunas tabela seguinte** com o resultado da função recursiva e o número de multiplicações efetuadas para os sucessivos valores de n.

n	F(n) – Versão Recursiva	Nº de Multiplicações	F(n) – Versão de Programação Dinâmica	Nº de Multiplicações
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				

NOME: **Nº MEC:**

- Analisando os dados da tabela, estabeleça uma **ordem de complexidade** para a **função recursiva**.

Programação Dinâmica

- Uma forma alternativa de resolver alguns problemas recursivos, para evitar o cálculo repetido de valores, consiste em efetuar esse cálculo de baixo para cima (*"bottom-up"*), ou seja, de **F(0)** para **F(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 os valores intermédios.
- Usando **programação dinâmica**, implemente uma **função iterativa** para calcular F(n). **Não utilize um array global.**
- Construa um programa para executar a função iterativa que desenvolveu para **sucessivos valores de n** e que permita **contar o número de multiplicações efetuadas** para cada valor de n.
- **Preencha as últimas colunas tabela anterior** com o resultado da função iterativa e o número de multiplicações efetuadas para os sucessivos valores de n.
- Analisando os dados da tabela, estabeleça uma **ordem de complexidade** para a **função iterativa**.

Função Recursiva – Análise Formal da Complexidade

- Escreva uma **expressão recorrente** (direta) para o **número de multiplicações** efetuadas pela função recursiva F(n). Obtenha, depois, uma **expressão recorrente simplificada**. Note que $\sum_{k=0}^{n-3} \text{Mult}(k) = \sum_{k=0}^{n-3} \text{Mult}(n-3-k)$. **Sugestão:** efetue a subtração $\mathbf{Mult(n) - Mult(n-1)}$.

NOME: **Nº MEC:**

- A equação de recorrência obtida é uma **equação de recorrência linear não homogénea**. Considere a correspondente **equação de recorrência linear homogénea**. Determine as raízes do seu **polinómio característico** (Sugestão: use o **Wolfram Alpha**). Sem determinar as constantes associadas, escreva a **solução da equação de recorrência linear não homogénea**.

- Usando a solução da equação de recorrência obtida acima, determine a **ordem de complexidade do número de multiplicações** efetuadas pela função recursiva. **Compare** a ordem de complexidade que acabou de obter com o resultado da **análise experimental**.

Programação Dinâmica – Análise Formal da Complexidade

- Considerando o número de multiplicações efetuadas pela função iterativa, efetue a análise formal da sua complexidade. Obtenha uma **expressão exata e simplificada para o número de multiplicações** efetuadas.

- Usando a expressão obtida acima, determine a **ordem de complexidade do número de multiplicações** efetuadas pela função iterativa. **Compare** a ordem de complexidade que acabou de obter com o resultado da **análise experimental**.