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	0	1	0
1	1	0	1	0
2	1	0	1	0
3	3	1	3	1
4	6	3	6	3
5	12	7	12	6
6	26	16	26	10
7	57	36	57	15
8	125	80	125	21
9	279	177	279	28
10	630	391	630	36
11	1433	863	1433	45
12	3285	1904	3285	55
13	7584	4200	7584	66
14	17611	9264	17611	78
15	41109	20433	41109	91
16	96416	45067	96416	105
17	227088	99399	227088	120
18	536896	219232	536896	136
19	1273763	483532	1273763	153
20	3031485	1066464	3031485	171
21	7235573	2352161	7235573	190
22	17315668	5187855	17315668	210
23	41539777	11442175	41539777	231
24	99877435	25236512	99877435	253
25	240645375	55660880	240645375	276

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

Com base nos resultados experimentais a ordem de complexidade do algoritmo é O(2^n) porque à medida que aumentamos o n verifica-se que Mult(n)/Mult(n-1) ≈ 2 , logo o número de multiplicações cresce exponencialmente.

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**.

Com base nos resultados experimentais a ordem de complexidade do algoritmo é O(n^2) porque à medida que aumentamos o n verifica-se que f(2n)/f(n) ≈ 4
n= 5 → f(10)/f(5)=6
n=10 →f(20)/f(10)=4,75
n=20 →f(40)/f(20)=4,33

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 $\mathbf{Mult(n) - Mult(n-1)}$.

- $\mathbf{Mult(n)} = \text{Mult(n}-1) + \text{Mult(n}-2) + \sum_{k=0}^{n-3}(Mult(k) + Mult(n-3-k) + 1)=$

$\text{Mult(n}-1) + \text{Mult(n}-2) + \sum_{k=0}^{n-3} Mult(k) + \sum_{k=0}^{n-3} Mult(n-3-k) + \sum_{k=0}^{n-3} 1 =$

Como $\sum_{k=0}^{n-3} \text{Mult}(k) = \sum_{k=0}^{n-3} \text{Mult}(n-3-k)$ temos:

$\mathbf{Mult(n)} = \text{Mult(n}-1) + \text{Mult(n}-2) + \sum_{k=0}^{n-3} Mult(k) + \sum_{k=0}^{n-3} Mult(k) + \sum_{k=0}^{n-3} 1 =$

$\text{Mult(n}-1) + \text{Mult(n}-2) + 2\sum_{k=0}^{n-3} Mult(k) + n - 2$

- $\boldsymbol{Mult(n-1)} = Mult(n-2) + Mult(n-3) + 2\sum_{k=0}^{n-4} Mult(k) + n - 1 - 2$

- $\boldsymbol{Mult(n)} - \boldsymbol{Mult(n-1)} = \text{Mult(n}-1) + \text{Mult(n}-2) + 2\sum_{k=0}^{n-4} Mult(k) + n - 2$

$-Mult(n-2) - Mult(n-3) - 2\sum_{k=0}^{n-4} Mult(k) - n + 3 =$

$\text{Mult(n}-1) + 2\sum_{k=0}^{n-3} Mult(k) - Mult(n-3) - 2(-\text{Mult(n}-3) + \sum_{k=0}^{n-3} Mult(k)) + 1 =$

$\text{Mult(n}-1) + 2\sum_{k=0}^{n-3} Mult(k) - Mult(n-3) + 2Mult(n-3) - 2\sum_{k=0}^{n-3} Mult(k)) + 1 =$

$\text{Mult(n}-1) + Mult(n-3) + 1$

$Mult(n) - Mult(n-1) = \text{Mult(n}-1) + Mult(n-3) + 1 \Leftrightarrow$

$\Leftrightarrow Mult(n) = 2\text{Mult(n}-1) + Mult(n-3) + 1$

- 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**.

- Equação de recorrência linear não homogénea**:**
  $Mult(n) = 2\text{Mult}(\text{n} - 1) + Mult(n - 3) + 1$

- Equação de recorrência linear homogénea**:**
  $Mult(n) = 2\text{Mult}(\text{n} - 1) + Mult(n - 3) \Leftrightarrow x^3 - 2x^2 - 1 = 0 \Leftrightarrow$
  $\Leftrightarrow x = -0.10278 - 0.66546i \mid x = -0.10278 + 0.66546i \mid x = 2.2$

- Solução da equação de recorrência linear não homogénea:

  $$Mult(n) = A \times 2.2^n + B \times (-0.10278 + 0.66546i)^n + C \times (-0.10278 - 0.66546i)^n + D$$

- 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**.

De acordo com a expressão obtida na questão anterior conclui-se que a ordem de complexidade do número de multiplicações efetuadas pela função recursiva é O(2^n) visto que o fator que cresce mais na expressão obtida é o $A \times 2.2^n$.

Conclui-se que se obteve o mesmo resultado tanto na análise experimental como na análise formal.

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.

$\sum_{i=3}^{n} \sum_{k=0}^{i-3} 1 = \sum_{i=3}^{n} (i-2) = \sum_{i=3}^{n} i - 2\sum_{i=3}^{n} 1 = (-3 + \sum_{i=0}^{n} i) - 2(n-2) =$

$-3 + \frac{n(n+1)}{2} - 2n + 4 = 1 + \frac{n^2-3n}{2}$

- 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**.

De acordo com a expressão obtida acima a ordem de complexidade do número de multiplicações é O(n^2). Conclui-se que se obteve o mesmo resultado tanto na análise experimental como na análise formal.