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:

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**.

|  |
| --- |
|  |

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**.

|  |
| --- |
| Tendo em conta que os valores de resultado apresentados para as duas função são iguais, também a sua complexidade será igual. |

Função Recursiva – Análise Formal da Complexidade

Escreva uma **expressão recorrente** (direta) para o **número de multiplicações** efetuadas pela função recursivaF(n). Obtenha, depois, uma **expressão recorrente simplificada**. Note que . **Sugestão:** efetue a subtração .

|  |
| --- |
| * Expressão Recorrente * Sabendo que * Fazendo * Simplificando a formula |

**­­**

* 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 implica que de um lado estão todos os termos de uma sequência e no outro uma função diferente de 0. Adotando o mesmo modelo de uma função de 1º grau.      * Calcular ***Mult( n )***      * Calcular ***Mult2( n )*** * *Sendo assim a expressão fica* |

**­­**

* 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**.

|  |
| --- |
| Sabendo que podemos concluir que a complexidade é de 2.2n, o que significa que , ou seja, exponencial.  Comparando como os valores que observamos [aqui](#_toc491), para os dados experimentais, constatamos que se ambos apontam para uma complexidade exponencial, de facto esta é a complexidade apresentada pelo algoritmo. |

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**.

|  |
| --- |
|  |