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**.

|  |
| --- |
| Analisando os dados da tabela vemos que , assim vemos que a ordem de complexidade da nossa função recursiva é dada por . |

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**.

|  |
| --- |
| Analisando os dados da tabela que vem anexo vemos que M(n) – M(n-1) o valor aumenta linearmente e não converge para nenhum valor, mas quando fazemos M(n) – M(n-1) do resultado anterior vemos que a complexidade é linear, ou seja, o valor da diferença entre os dois termos obtidos dá 1, logo temos que O(n2). |

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 .

|  |
| --- |
| O termo (n-2) vem da multiplicação que é feita no somatório de F(n) quando fazemos F(k)\*F(n-3-k) n-2 vezes (por causa do somatório de 0 a n-3).  Para calcular M(n) – M(n-1) primeiro temos que calcular M(n) e M(n-1).  Calculando a diferença temos que: |

**­­**

* 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**.

|  |
| --- |
| A partir da fórmula:  Retiramos o polinómio característico:  Seguindo a sugestão metemos a expressão acima no Wolfram Alpha e obtemos as seguintes soluções:  Apenas retiramos a solução com valor real vistos que é esta que tem sentido para representar a complexidade do nosso algoritmo. |

**­­**

* 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**.

|  |
| --- |
| Retiramos das contas anteriores que x = 2,2056  Neste caso a complexidade é dada por , valor que é igual ao valor de complexidade obtido experimentalmente quando analisamos o caso |

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.

|  |
| --- |
| Neste caso temos que:  Desenvolvendo os somatórios obtemos o seguinte:  Ficamos então a saber que a expressão de multiplicações para o algoritmo de programação dinâmica é dada por: |

**­­**

* 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**.

|  |
| --- |
| Usando a expressão acima para M(n) podemos então dizer que:  Daqui retiramos que por majoração a complexidade é dada por que é exatamente o mesmo valor ao qual cheguei na analise experimental feita anteriormente. |