Aula 6 - Análise da Complexidade de Algoritmos Recursivos (Números de Motzkin)

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

* Os números de Motzkin

[1](https://en.wikipedia.org/wiki/3_(number)), [1](https://en.wikipedia.org/wiki/0_(number)), [2](https://en.wikipedia.org/wiki/2_(number)), 4, 9, [21](https://en.wikipedia.org/wiki/5_(number)), 51,... (<https://oeis.org/A001006>)

são definidos pela seguinte relação de recorrência:

Função Recursiva

* Implemente uma **função recursiva Motzkin(n)** que use diretamente a relação de recorrência acima, **sem qualquer simplificação**.
* Construa um programa para executar a função **Motzkin(n)** 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** | **Motzkin(n) – Versão Recursiva** | **Nº de Multiplicações** | **Motzkin(n) – Versão de Programação Dinâmica** | **Nº de Multiplicações** |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 2 | 2 | 1 | 2 | 1 |
| 3 | 4 | 3 | 4 | 3 |
| 4 | 9 | 8 | 9 | 6 |
| 5 | 21 | 20 | 21 | 10 |
| 6 | 51 | 49 | 51 | 15 |
| 7 | 127 | 119 | 127 | 21 |
| 8 | 323 | 288 | 323 | 28 |
| 9 | 835 | 696 | 835 | 36 |
| 10 | 2188 | 1681 | 2188 | 45 |
| 11 | 5798 | 4059 | 5798 | 55 |
| 12 | 15511 | 9800 | 15511 | 66 |
| 13 | 41835 | 23660 | 41835 | 78 |
| 14 | 113634 | 57121 | 113634 | 91 |
| 15 | 310572 | 137903 | 310572 | 105 |

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

|  |
| --- |
| Para estabelecer uma ordem de complexidade para esta função analisando apenas os dados da tabela, podemos suspeitar que esta será exponencial (progressão geométrica isso significa que a razão entre termos sucessivos é fixa). Isto pode ser confirmado se fizermos a razão entre termos sucessivos do número de multiplicações da versão recursiva (azul).    Através dos dados da tabela podemos ver que o valor da razão tende para 2,414, assim conseguimos estabelecer ordem de complexidade exponencial , de razão , logo . |

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 **Motzkin(0)** para **Motzkin(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 Motzkin(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**.

|  |
| --- |
| Para estabelecer uma ordem de complexidade para esta função analisando apenas os dados da tabela, podemos suspeitar que esta será exponencial. Para termos a certeza, podemos fazer a razão entre termos sucessivos do número de multiplicações da versão iterativa (verde).    Através disto, podemos ver que o valor da razão tende para 1, logo não poderá ser exponencial. Por tender para 1 suspeitamos que seja polinomial.  Para analisarmos este caso, em vez de calcularmos a razão entre termos sucessivos podemos calcular a razão entre n e 2n.  n = 4 → 6 e para n = 8 → 28 assim 28/6 = 4,66  n = 5 → 10 e para n = 10 → 45 assim 45/10 = 4,5  n = 6 → 15 e para n = 12 → 66 assim 66/15 = 4,4  n = 7 → 21 e para n = 14 → 91 assim 91/21 = 4,33  Podemos ver que tende para 4 (2²), logo é uma polinomial de grau 2, assim podemos dizer que este algoritmo tem ordem de complexidade quadrática . |

Função Recursiva – Análise Formal da Complexidade

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

|  |
| --- |
| A partir daqui podemos ver que .  Expressão recorrente (direta) :  R: A expressão recorrente simplificada obtida é . |

* 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**. Sem determinar as constantes associadas, escreva a **solução da equação de recorrência linear não homogénea**.

|  |
| --- |
| Equação de recorrência linear não homogénea:  Equação de recorrência linear homogénea:  Grau  , com A e B constantes e com . |

* 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**.

|  |
| --- |
| , com A e B constantes e com .  Partindo desta expressão e ignorando os termos de menor ordem e as constantes de multiplicação, chegamos a . A partir daqui vemos que a ordem de complexidade do número de multiplicações é exponencial, .  A complexidade obtida na análise experimental também era exponencial, , mas tinha um expoente diferente. Desta análise sabemos que este algoritmo estava limitado por , ou seja, o que obtivemos na análise formal não pode ser superior a . Isto observou-se pois na análise formal obtivemos .  Através do gráfico seguinte podemos verificar que a função está limitada pela função . |

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.

|  |
| --- |
| Seja o número de multiplicações.  A expressão exata e simplificada para o número de multiplicações é  Podemos confirmar substituindo números:  n = 1 → n = 3 → n = 5 → n = 7 →  n = 10 → n = 12 → n = 15 → |

* 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**.

|  |
| --- |
| Efetuando a análise formar deste algoritmo quanto ao nível do número de multiplicações, podemos ver que tem ordem de complexidade quadrática - .  Na análise experimental obtivemos que a ordem de complexidade deste algoritmo séria polinomial de grau 2, ou seja, quadrática.  Os resultado obtidos na análise experimental e na análise formal condizem. |