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**.

|  |
| --- |
| Por observação dos valores tabela num gráfico, o algoritmo tem uma ordem de complexidade exponencial, O(a^n), com a ≈ 2,414. |

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**.

|  |
| --- |
| Por observação dos valores tabela num gráfico, o algoritmo tem uma ordem de complexidade quadrática, O(n^2). |

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 .

|  |
| --- |
| \*[∑] → somatório de k = 0 até k = n-2  \*(∑) → somatório de k = 0 até k = n-3  Mult(0) = Mult(1) = 0  Mult(2) = Mult(1) + 1 + [∑](Mult(k) + Mult(n-2-k)) = 0 + 1 + [∑](0 + 0) = 1  Mult(4) = Mult(2) + 3 + [∑](Mult(k) + Mult(n-2-k)) = 3 + 3 + [∑](Mult(k)) + [∑](Mult(n-2-k) =  = 6 + (0 + 0 + 1) + (1 + 0 + 0) = 8  Mult(n) = Mult(n-1) + (n-1) + [∑](Mult(k) + Mult(n-2-k)) <=>  Mult(n) = Mult(n-1) + (n-1) + 2 \* [∑](Mult(k)) e Mult(n-1) = Mult(n-2) + (n-2) + 2 \* (∑)(Mult(k))  Mult(n) – Mult(n-1) = Mult(n-1) + (n-1) + 2 \* [∑](Mult(k)) -(Mult(n-2) + (n-2) + 2 \* (∑)(Mult(k))<=>  Mult(n) – Mult(n-1) = Mult(n-1) – Mult(n-2) + (n-1) – (n-2) + 2 \* [∑](Mult(k)) – 2 \* (∑)(Mult(k))<=>  Mult(n) – Mult(n-1) = Mult(n-1) – Mult(n-2) + 1 + 2\*(Mult(n-2) + (∑)(Mult(k)) – 2 \*(∑)(Mult(k))<=>  Mult(n) – Mult(n-1) = Mult(n-1) – Mult(n-2) + 2 \* Mult(n-2) + 1 +2\*(∑)(Mult(k) – 2\*(∑)(Mult(k))<=>    Mult(n) – Mult(n-1) = Mult(n-1) + Mult(n-2) + 1 <=>  Mult(n) = 2 \* Mult(n-1) + Mult(n-2) + 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**. Sem determinar as constantes associadas, escreva a **solução da equação de recorrência linear não homogénea**.

|  |
| --- |
| 0 = -Mult(n) + 2 \* Mult(n-1) + Mult(n-2)  Substiuindo  Mult(n) = a \* b^n  Mult(n-1) = a \* b^(n-1)  Mult(n-2) = a \* b^(n-2)  Podemos dizer  a\*b^n = 2\*a\*b^(n-1) + a\*b^(n-2)  Dividindo por ab^(n-2) ficamos com  b^2 = 2b + 1 <=> -1 \* b^2 + 2b +1 = 0  Concluimos, que  b = 1 ± √(2)  Se Mult(n) = a1\*(1-√(2))^n →  → a1\*(1-√(2))^n = 2 \* a1\*(1-√(2))^(n-1) + a1\*(1-√(2))^(n-2)  Se Mult(n) = a2\*(1+√(2))^n →  → a2\*(1+√(2))^n = 2 \* a2\*(1+√(2))^(n-1) + a2\*(1+√(2))^(n-2)  Solução:  → Mult(n) = a1\*(1-√(2))^n + a2\*(1+√(2))^n + C |

**­­**

* 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**.

|  |
| --- |
| A ordem de complexidade é exponencial, O(a^n), assim como 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.

|  |
| --- |
| Mult(n) = n\*(n-1)/2, ou seja um somatório de k=2 até k=n de um somatório de i=0 até i=k-2 somando 1 |

**­­**

* 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**.

|  |
| --- |
| Mult(n) = n\*(n-1)/2  A ordem de complexidade é quadrática, O(n^2),  como visto na análise experimental. |