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, 1, 2, 4, 9, 21, 51,... (https://oeis.org/A001006)

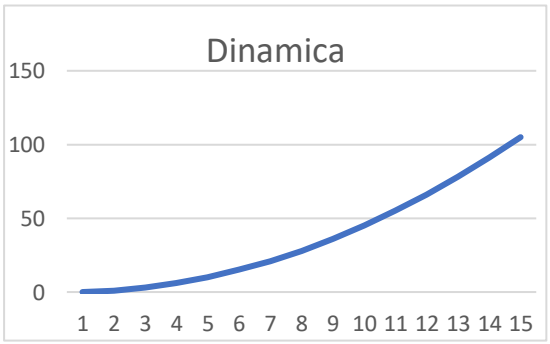
  são definidos pela seguinte relação de recorrência:

$$\text{Motzkin}(n) = \begin{cases} 1, \text{ se } n = 0 \text{ e } n = 1 \\ \text{Motzkin}(n-1) + \sum_{k=0}^{n-2} \text{Motzkin}(k) \times \text{Motzkin}(n-2-k), \text{ se } n > 1 \end{cases}$$

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.

index	value	recursive	dynamic
0	1	0	0
1	1	0	0
2	2	1	1
3	4	3	3
4	9	8	6
5	21	20	10
6	51	49	15
7	127	119	21
8	323	288	28
9	835	696	36
10	2188	1681	45
11	5798	4059	55
12	15511	9800	66
13	41835	23660	78
14	113634	57121	91
15	310572	137903	105



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

Observando a tabela, o algoritmo parece ter uma ordem de complexidade de multiplicações exponencial.

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**.

Observando a tabela, o algoritmo parece ter uma ordem de complexidade de multiplicações polinomial (quadrática).

Função Recursiva – Análise Formal da Complexidade

- Escreva uma **expressão recorrente** (direta) para o **número de multiplicações** efetuadas pela função recursiva Motzkin(n). Obtenha, depois, uma **expressão recorrente simplificada**. Note que $\sum_{k=0}^{n-2} \text{Mult}(k) = \sum_{k=0}^{n-2} \text{Mult}(n-2-k)$. **Sugestão:** efetue a subtração $\mathbf{Mult(n) - Mult(n-1)}$.

M(n) = 1, n== 0 ou n == 1

M(n) = $M(n-1)+\sum_{k=0}^{n-2}(1+M(k)+M(n-2-k))$, para n > 1

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

$$M(n-1)+\sum_{k=0}^{n-2}(1+2*M(k)) = M(n-1)+2*\sum_{k=0}^{n-2}(\frac{1}{2}+M(k))$$

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

$$M(n-1)-M(n-2)+2*\left(\sum_{k=0}^{n-2}(\frac{1}{2}+M(k))-\sum_{k=0}^{n-3}(\frac{1}{2}+M(k))\right) =$$

$$M(n-1)-M(n-2)+2*\left(\sum_{k=0}^{n-3}(\frac{1}{2}+M(k))-\sum_{k=0}^{n-3}(\frac{1}{2}+M(k))+(\frac{1}{2}+M(n-2))\right) =$$

$$M(n-1)-M(n-2)+2M(n-2)+1$$

$$M(n)-M(n-1) = M(n-1)+M(n-2)+1 \equiv$$

$$M(n) = 2*M(n-1)+M(n-2)+1 \equiv$$

$$M(n) = 1/4((1-\sqrt{(2)})^n+(1+\sqrt{(2)})^n-2)$$

n	$M(n)$
0	0
1	0
2	1
3	3
4	8
5	20
6	49
7	119
8	288
9	696

- 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**.

Raízes: 1 +/- sqrt(2)
M(n) = c1 (1 - sqrt(2))^n + c2 (1 + sqrt(2))^n – 1/2

- 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 do algoritmo é exponencial, a mesma ordem de complexidade prevista através da analise experimental. (O(k^n))

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.

Dos for loops no programa podemos extrair os somatórios

$$M_d(n) = \sum_{i=2}^{n} \left(\sum_{k=0}^{i-2} (1) \right) = \frac{n(n-1)}{2}$$

Os valores obtidos com a fórmula fechada corroboram os valores experimentais obtidos

n	$\frac{1}{2}(n-1)n$
1	0
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45

- 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**.

A ordem de complexidade do algoritmo iterativo é polinomial (quadrática), a mesma obtida através da observação dos dados experimentais. O(n^2)