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.

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

NOME: RAQUEL RESENDE MILHEIRO PINTO Nº MEC: 92948

- 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).

n	ResMorRec	NmulRec	ResMorIter	NmulIter		RNmulRec
0	1	0	1	0		
1	1	0	1	0		#DIV/0!
2	2	1	2	1		#DIV/0!
3	4	3	4	3		3
4	9	8	9	6		2,666667
5	21	20	21	10		2,5
6	51	49	51	15		2,45
7	127	119	127	21		2,428571
8	323	288	323	28		2,420168
9	835	696	835	36		2,416667
10	2188	1681	2188	45		2,41523
11	5798	4059	5798	55		2,414634
12	15511	9800	15511	66		2,414388
13	41835	23660	41835	78		2,414286
14	113634	57121	113634	91		2,414243
15	310572	137903	310572	105		2,414226

Através dos dados da tabela podemos ver que o valor da razão tende para 2,414, assim conseguimos estabelecer ordem de complexidade exponencial $O(r^n)$, de razão $\approx$ 2,414, logo $O(2{,}414^n)$.

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.

NOME: RAQUEL RESENDE MILHEIRO PINTO Nº MEC: 92948

- **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).

n	ResMorRec	NmulRec	ResMorIter	NmulIter		RNmulIter
0	1	0	1	0		
1	1	0	1	0		#DIV/0!
2	2	1	2	1		#DIV/0!
3	4	3	4	3		3
4	9	8	9	6		2
5	21	20	21	10		1,666667
6	51	49	51	15		1,5
7	127	119	127	21		1,4
8	323	288	323	28		1,333333
9	835	696	835	36		1,285714
10	2188	1681	2188	45		1,25
11	5798	4059	5798	55		1,222222
12	15511	9800	15511	66		1,2
13	41835	23660	41835	78		1,181818
14	113634	57121	113634	91		1,166667
15	310572	137903	310572	105		1,153846

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 (2^2), logo é uma polinomial de grau 2, assim podemos dizer que este algoritmo tem ordem de complexidade quadrática $O(n^2)$.

NOME: RAQUEL RESENDE MILHEIRO PINTO Nº MEC: 92948

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 $\text{Mult}(n) - \text{Mult}(n-1)$.

$\text{Mult}(4) = \text{Mult}(3) + (4-1) + (\text{Mult}(0) + \text{Mult}(2)) + (\text{Mult}(1) + \text{Mult}(1)) + (\text{Mult}(2) + \text{Mult}(0))$

A partir daqui podemos ver que $\sum_{k=0}^{n-2} \text{Mult}(k) = \sum_{k=0}^{n-2} \text{Mult}(n-2-k)$.

Expressão recorrente (direta) :

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

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

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

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

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

$$\text{Mult}(n) - \text{Mult}(n-1) =$$

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

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

$$= \text{Mult}(n-1) + (n-1) + 2 * \text{Mult}(n-2) - \text{Mult}(n-2) - (n-2) = \text{Mult}(n-1) + \text{Mult}(n-2) + 1$$

$$\text{Mult}(n) - \text{Mult}(n-1) = \text{Mult}(n-1) + \text{Mult}(n-2) + 1 \equiv \text{Mult}(n) = 2 * \text{Mult}(n-1) + \text{Mult}(n-2) + 1$$

R: A expressão recorrente simplificada obtida é $\text{Mult}(n) = 2 * \text{Mult}(n-1) + \text{Mult}(n-2) + 1$.

NOME: RAQUEL RESENDE MILHEIRO PINTO Nº MEC: 92948

- 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:

$\text{Mult}(n)=2*\text{Mult}(n-1)+\text{Mult}(n-2)+1$

Equação de recorrência linear homogénea:

$\text{Mult}(n)=2*\text{Mult}(n-1)+\text{Mult}(n-2)\equiv \text{Mult}(n)-2*\text{Mult}(n-1)-Mult(n-2)=0$

$\text{Grau}=n-(n-2)=2$

$n^2-2n-1=0\equiv n=\frac{2\pm\sqrt{(-2)^2-4*1*(-1)}}{2*1}\equiv n=\frac{2\pm\sqrt{4+4}}{2}\equiv n=\frac{2\pm\sqrt{8}}{2}\equiv n=\frac{2\pm 2\sqrt{2}}{2}\equiv n=1\pm\sqrt{2}\equiv$
$\equiv n=1+\sqrt{2}$ v $n=1-\sqrt{2}$

$\text{Mult}(n)=A(1+\sqrt{2})^n+B(1-\sqrt{2})^n$, com A e B constantes e com $n\geq 0$.

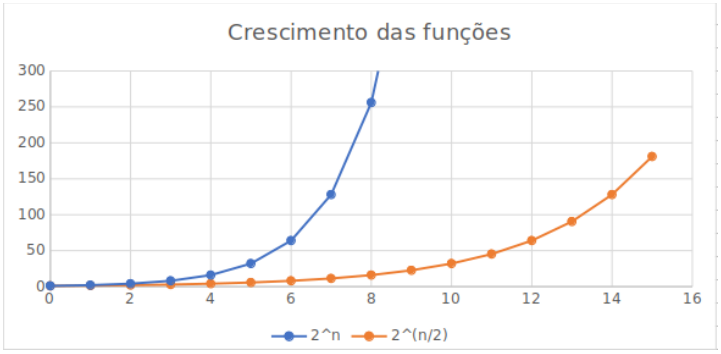
- Usando a solução da equação de recorrência obtida acima, determine a **ordem de complexidade do número de multiplicações** efetuadas pela função recursiva. **Compare** a ordem de complexidade que acabou de obter com o resultado da **análise experimental**.

$\text{Mult}(n)=A(1+\sqrt{2})^n+B(1-\sqrt{2})^n$, com A e B constantes e com $n\geq 0$.

Partindo desta expressão e ignorando os termos de menor ordem e as constantes de multiplicação, chegamos a $(\sqrt{2})^n=2^{\left(\frac{n}{2}\right)}$. A partir daqui vemos que a ordem de complexidade do número de multiplicações é exponencial, $O\left(2^{\left(\frac{n}{2}\right)}\right)$.

A complexidade obtida na análise experimental também era exponencial, $O(2^n)$, mas tinha um expoente diferente. Desta análise sabemos que este algoritmo estava limitado por $O(2^n)$, ou seja, o que obtivemos na análise formal não pode ser superior a $O(2^n)$. Isto observou-se pois na análise formal obtivemos $O\left(2^{\left(\frac{n}{2}\right)}\right)$.
Através do gráfico seguinte podemos verificar que a função $2^{\left(\frac{n}{2}\right)}$ está limitada pela função 2^n.

NOME: RAQUEL RESENDE MILHEIRO PINTO Nº MEC: 92948



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 $\text{Mult}(n)$ o número de multiplicações.

$$\text{Mult}(n)=\sum_{k=2}^{n}\left(\sum_{l=0}^{k-2}1\right)=\sum_{k=2}^{n}(k-1)=\frac{n(n-1)}{2}=\frac{n^2-n}{2}$$

A expressão exata e simplificada para o número de multiplicações é $\text{Mult}(n)=\frac{n^2-n}{2}$

Podemos confirmar substituindo números:

$n = 1 \rightarrow \frac{1^2-1}{2}=0$ $\quad n = 3 \rightarrow \frac{3^2-3}{2}=3$ $\quad n = 5 \rightarrow \frac{5^2-5}{2}=10$ $\quad n = 7 \rightarrow \frac{7^2-7}{2}=21$

$n = 10 \rightarrow \frac{10^2-10}{2}=45$ $\quad n = 12 \rightarrow \frac{12^2-12}{2}=66$ $\quad n = 15 \rightarrow \frac{15^2-15}{2}=105$

- 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**.

$$\text{Mult}(n)=\frac{n^2-n}{2}$$

Efetuando a análise formar deste algoritmo quanto ao nível do número de multiplicações, podemos ver que tem ordem de complexidade quadrática - $O(N^2)$.

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.

NOME: RAQUEL RESENDE MILHEIRO PINTO Nº MEC: 92948