Aula 6 - Análise da Complexidade de Algoritmos Recursivos

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

Implemente os seguintes **algoritmos recursivos** – **sem recorrer a funções de arredondamento** (floor e ceil) – e analise o **número de chamadas recursivas** executadas por cada algoritmo.

Deve utilizar **aritmética inteira**: n/4 é igual a e (n+3)/4 é igual a .

* **Preencha a tabela da página seguinte** com o resultado de cada função e o número de chamadas recursivas para os sucessivos valores de n.
* Analisando os dados da tabela, estabeleça uma ordem de complexidade para cada algoritmo.

|  |
| --- |
| Complexidade do primeiro algoritmo ->  Complexidade do Segundo algoritmo ->  Complexidade do terceiro algoritmo - > |

* Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função **.** Obtenha, depois, uma **expressão exata e simplificada;** determine a sua **ordem de complexidade**. Compare a expressão obtida com os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico**.

|  |
| --- |
| Sendo G(n) o numero de chamadas recursivas para T1(n), então o numero de chamadas recursivas efetuadas pela função T1(n) pode ser dado por:  G(0)=1 e portanto a expressão recorrente pode ser: G(n) = .  No entanto, pode-se obter uma expressão exata e simplificada para o numero de chamadas recursivas dependendo apenas do n: Deste modo, a ordem de complexidade é .  Podemos verificar a expressão matemática obitda com os resultados da tabela. Por exemplo, para n = 23 temos , o que corresponde aos valores obtidos na tabela. |

**­­**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | **T1(n)** | **Nº de Chamadas Recursivas** | **T2(n)** | **Nº de Chamadas Recursivas** | **T3(n)** | **Nº de Chamadas Recursivas** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 2 | 2 | 1 | 2 | 0 | 2 | 0 |
| 3 | 3 | 1 | 3 | 0 | 3 | 0 |
| 4 | 5 | 2 | 6 | 2 | 6 | 1 |
| 5 | 6 | 2 | 8 | 2 | 8 | 2 |
| 6 | 7 | 2 | 9 | 2 | 9 | 2 |
| 7 | 8 | 2 | 10 | 2 | 10 | 2 |
| 8 | 10 | 2 | 12 | 2 | 12 | 1 |
| 9 | 11 | 2 | 14 | 2 | 14 | 2 |
| 10 | 12 | 2 | 15 | 2 | 15 | 2 |
| 11 | 13 | 2 | 16 | 2 | 16 | 2 |
| 12 | 15 | 2 | 18 | 2 | 18 | 1 |
| 13 | 16 | 2 | 22 | 4 | 22 | 3 |
| 14 | 17 | 2 | 23 | 4 | 23 | 3 |
| 15 | 18 | 2 | 24 | 4 | 24 | 3 |
| 16 | 21 | 3 | 28 | 6 | 28 | 2 |
| 17 | 22 | 3 | 31 | 6 | 31 | 5 |
| 18 | 23 | 3 | 32 | 6 | 32 | 5 |
| 19 | 24 | 3 | 33 | 6 | 33 | 5 |
| 20 | 26 | 3 | 36 | 6 | 36 | 3 |
| 21 | 27 | 3 | 38 | 6 | 38 | 6 |
| 22 | 28 | 3 | 39 | 6 | 39 | 6 |
| 23 | 29 | 3 | 40 | 6 | 40 | 6 |
| 24 | 31 | 3 | 42 | 6 | 42 | 3 |
| 25 | 32 | 3 | 44 | 6 | 44 | 6 |
| 26 | 33 | 3 | 45 | 6 | 45 | 6 |
| 27 | 34 | 3 | 46 | 6 | 46 | 6 |
| 28 | 36 | 3 | 48 | 6 | 48 | 3 |

* Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função **. Considere o caso particular e** obtenha uma **expressão exata e simplificada;** determine a **ordem de complexidade** para esse caso particular. Compare a expressão obtida com os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

|  |
| --- |
| A expressão recorrente para o número de chamadas recursivas efetuadas na função T2(n) é dada por :  As divisões por 4 são divisões inteiras, G(n) é o número de chamadas recursivas da função T2 para o valor de n.  Para todos os valores de , o valor k representa um qualquer número inteiro, assim podemos concluir que o número de chamdas recursivas da função é dada por . Por exemplo para , e portanto , simplificando a expressão sabendo que . Temos assim , calculando o número de chamadas recursivas para valor de n multiplos de 4. Com esta expressão concluímos assim que a complexidade do algoritmo é .  Com o teorema mestre confirmamos o resultado obtido, , considerando apenas valores de onde podemos identificar que os valores são , portanto podemos chegar ao resultado de . Relacionando todos estes valores temos que . Assim segundo o teorema mestre:  . |

Pode **generalizar a ordem de complexidade** que acabou de obter para todo o n? **Justifique.**

|  |
| --- |
| Sabendo que é uma função não decrescente e F(n) um função “smooth”, e que para valores de n que sejam potências de b, neste caso concreto o valor é 4, assim podemos concluir que para qualquer valor de n , . |

* Obtenha uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função

|  |
| --- |
| é o número de chamadas recursivas da função para um determinado valor n, assim a expressão recorrente é dada por: |

* **Considere o caso particular e** obtenha uma **expressão exata e simplificada;** determine a **ordem de complexidade** para esse caso particular. Compare a expressão obtida com os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

|  |
| --- |
| Para o caso particular onde o , temos uma situação semelhante ao algoritmo T1(n), com uma chamada recursiva a menos.  Portanto, para apenas os valores que são potências de 4, o número de chamadas recursivas pode simplesmente ser dado por .  Por exemplo, para n = 16, temos que o número de chamadas recursivas é igual a 3.  Com o teorema mestre confirmamos o resultado obtido, para , n é multiplo de 4 e assim consideramos o ramo: , verificamos assim que , assim . Relacionando, temos que . segundo o teorema mestre: |

* Pode **generalizar a ordem de complexidade** que acabou de obter para todo o n? **Justifique.**

|  |
| --- |
| Sim podemos generalizar a ordem de complexidade porque para todos os outros valores de n a complexidade nunca será superior à do caso particular em que , logo O(n). |

* Atendendo às **semelhanças entre e**  estabeleça uma **ordem de complexidade para . Justifique.**

|  |
| --- |
| O número de chamadas recursivas em ambas a funções T2(n) e T3(n) tem pouca variação entre elas,portanto podemos considerar que a ordem complexidade de T3(n) está contida na ordem de complexidade de T2(n). Como a ordem de complexidade de T2(n) é O(), assim concluímos que a ordem de complexidade de T3(n) é também O(. |