|  |  |  |  |
| --- | --- | --- | --- |
| **Nome:** | **Sofia Teixeira Vaz** | **N.º Mec:** | **92968** |

Aula 5 - 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/3 é igual a e (n+2)/3 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?

|  |
| --- |
| T1: logarítmico (base 3)  T2: n^b, b entre 1 e 2  T3: sensivelmente a mesma que T2 |

* 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 a os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico**.

|  |
| --- |
| Sendo G(n) o número de chamadas recursivas, G(n) = 1 + G(n/3) = 1+(1+G(n/(3\*3)) = 2 + G(n/(3\*3)) = 3 + G(n/(3\*3\*3)) = … = k + G(n/3^k) Assim, a ordem de complexidade deste algoritmo será o logaritmo em base 3 de n. |

**­­**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 4 | 2 | 5 | 2 | 5 | 1 |
| 4 | 5 | 2 | 7 | 2 | 7 | 2 |
| 5 | 6 | 2 | 8 | 2 | 8 | 2 |
| 6 | 8 | 2 | 10 | 2 | 10 | 1 |
| 7 | 9 | 2 | 14 | 4 | 14 | 3 |
| 8 | 10 | 2 | 15 | 4 | 15 | 3 |
| 9 | 13 | 3 | 19 | 6 | 19 | 2 |
| 10 | 14 | 3 | 22 | 6 | 22 | 5 |
| 11 | 15 | 3 | 23 | 6 | 23 | 5 |
| 12 | 17 | 3 | 26 | 6 | 26 | 3 |
| 13 | 18 | 3 | 28 | 6 | 28 | 6 |
| 14 | 19 | 3 | 29 | 6 | 29 | 6 |
| 15 | 21 | 3 | 31 | 6 | 31 | 3 |
| 16 | 22 | 3 | 34 | 6 | 34 | 5 |
| 17 | 23 | 3 | 35 | 6 | 35 | 5 |
| 18 | 26 | 3 | 38 | 6 | 38 | 2 |
| 19 | 27 | 3 | 43 | 8 | 43 | 6 |
| 20 | 28 | 3 | 44 | 8 | 44 | 6 |
| 21 | 30 | 3 | 49 | 10 | 49 | 4 |
| 22 | 31 | 3 | 51 | 10 | 51 | 8 |
| 23 | 32 | 3 | 52 | 10 | 52 | 8 |
| 24 | 34 | 3 | 54 | 10 | 54 | 4 |
| 25 | 35 | 3 | 59 | 12 | 59 | 7 |
| 26 | 36 | 3 | 60 | 12 | 60 | 7 |
| 27 | 40 | 4 | 65 | 14 | 65 | 3 |
| 28 | 41 | 4 | 69 | 14 | 69 | 9 |

* 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 a os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

|  |
| --- |
| T(n) = 2 + 2\*T(n/3)  n = 3^k, logo, b=3, k>=1  T(n) = a\*T(n/b) + f(n) -> a=2 e f(n) =2 , logo, d = 0  a = 2 e d = 0 -> b^d = 1<2  Logo, T(n) pertence a Θ(n^(logb(a))) (logaritmo em base b de a) = Θ(n^(log3(2))) (Teorema Mestre) |

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

|  |
| --- |
| Sim, uma vez que a ordem de complexidade se mantém uniforme para qualquer valor, isto é, não há casos nos quais T(n) > T(n+1). Além disso, testando, todos os valores aparentam obedecer à ordem de complexidade. |

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

|  |
| --- |
| Caso n<4, o número de chamadas será 0  Para n divisível por 3, T(n) = 1+T(n/3)  Para todos os outros casos, T(n) = 2+ T(n/3) + T((n+2)/3) |

* **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 a os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

|  |
| --- |
| T(n) = 1 + T(n/3)  a = 1, f(n) = 1, logo, d= 0.  a = b^0 = 1 -> T(n) pertence a Θ(n^0 \* log(n) ) = Θ(log(n)) |

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

|  |
| --- |
| Não, uma vez que T(n) é composto por vários ramos que não os casos base e apenas estamos a ter em conta um. Para além disso, se observarmos a tabela, verificar-se-há que há casos que não obedecem ao valor calculado. |

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

|  |
| --- |
| O número de chamadas recursivas de T3 nunca excede as de T2, logo, a ordem de complexidade de T3 será O(n^(log3(2))) |