|  |  |  |  |
| --- | --- | --- | --- |
| **Nome:** | **Lúcia Maria Bessa de Sousa** | **N.º Mec:** | **93086** |

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 – O(log3(n));  T2 – O(n) é linear;  T3 – O(n) é linear; |

* 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**.

|  |
| --- |
| C1(0) = 0  C1(n) = C1(n/3) + 1 = C1((n/3)/3) + 1 + 1= C1(n/9) +2  C1(n) = C1(n/3k) + k =C1(3k/3k) + k  k = log3(n) + 1  C1(n)= 1 + log3(n) + C1(0)  O(log3(n)) |

**­­**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | **T1(n)** | **Nº de Chamadas Recursivas** | **T2(n)** | **Nº de Chamadas Recursivas** | **T3(n)** | **Nº de Chamadas Recursiva** |
| 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**.

|  |
| --- |
| Para T2: Triângulo de Pascal: 20+21+22=23-1; 21+22=23-2  T2(0) = 0  T2(1) = 1  T2(2) = 2  C2(n) = C2(n/3) + C2(n/3) + 2 = 2\*C2(n/3) + 2  = 2 + 2(2 + 2\*C(n/9)) = 2 + 4 + 4\*C(n/9) = 21 + 22 + (22)\*C(n/9)  = 23 - 2 + (22)\*C(n/9) = 2\*(22-1) + (2k)\*C(n/3k) = 2\*(2k-1) + (2k)\*C(n/3k)  n = 3k;  2\*(2k-1) + (2k)\*C(n/3k) = 2\*(2k-1) + (2k)\*C(3k/3k) = 2\*(2k-1) + (2k)\*C(1) → C(1) = 0  k = log3(n);  C2(n) = 2\*(2log3(n) – 1) = 2\*(nlog3(2) – 1)  O(nlog3(2))  Aplicando o Teorema Mestre:  C2(n) = 2\*C2(n/3) + 2  a = 2; b = 3  f(n) = 2 → d = 0  a = 2 > bd = 30 = 1  Podemos assim conluir que O(nlog3(2)). |

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

|  |
| --- |
| → nlog3(2) é uma função suave;  → C2(n) é uma função eventualmente não decrescente;  → C2(n) = O(nlog3(2)), para n = 3k, ou seja, para todo o n que seja potência de 3, sendo que 3 ≥ 2.  Pela regra da suavidade, C2(n) = O(nlog3(2)) para qualquer valor de n.  Sendo assim possível generalizar a ordem de complexidade para todo o n. |

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

|  |
| --- |
| Para T3:  T3(0) = 0  T3(1) = 1  T3(2) = 2  No ramo para n múltiplo de 3:  → T3(n) = 2\*T3(n/3)+n  C3(n) = 2\*C3(n/3)+1 = 2\*(2\*C3((n/3)/3)+1) + 1 = 4\*C3(n/9) + 3  C3(n) = 2k \* C3(n/3k) + (k+1) |

* **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**.

|  |
| --- |
| n = 3k; k = log3(n); n é múltiplo de 3  C3(n) = 4\*C3(n/9) + 3 = 2k \* C3(n/3k) + (k+1)  C(n) = log3(n) + 1    O(log3(n))  Aplicando o Teorema Mestre:  → T3(n) = 2\*T3(n/3)+n  a = 1; b = 3;  f(n) = 1 → d = 0;  1 = 30, logo podemos concluir que O(log3(n)). |

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

|  |
| --- |
| → log3(n) é uma função suave;  → C3(n) é uma função eventualmente não decrescente;  → C3(n) = O(log3(n)), para n = 3k, ou seja, para todo o n que seja potência de 3, sendo que 3 ≥ 2.  Pela regra da suavidade, C3(n) = O(log3(n)) para qualquer valor de n.  Sendo assim possível generalizar a ordem de complexidade para todo o n. |

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

|  |
| --- |
| O comportamento de T2 limita comportamento de T3.  T2 realiza mais operações e majora T3, logo podemos dizer que ordem de complexidade de T3 não pode ser maior que a ordem de complexidade de T2.  Podemos assim concluir que uma possível ordem de complexidade de T3 é igual à ordem complexidade de T2. |