NOME:	RAQUEL RESENDE MILHEIRO PINTO	N.º MEC:	92948

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.

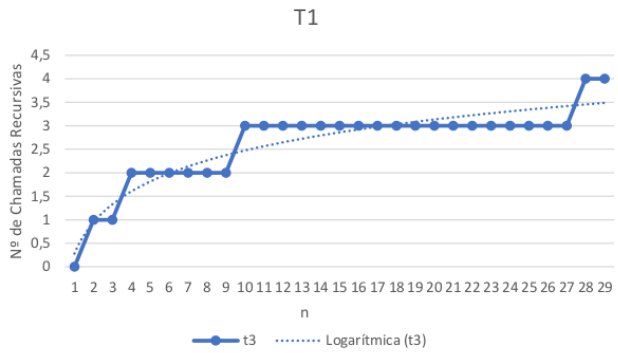
$$T_1(n)=\begin{cases}0, se\ n=0\\ T_1\left(\lfloor \frac{n}{3} \rfloor\right)+n, se\ n>0\end{cases} \qquad T_2(n)=\begin{cases}n, se\ n=0,1,2\\ T_2\left(\lfloor \frac{n}{3} \rfloor\right)+T_2\left(\lceil \frac{n}{3} \rceil\right)+n, se\ n>2\end{cases}$$

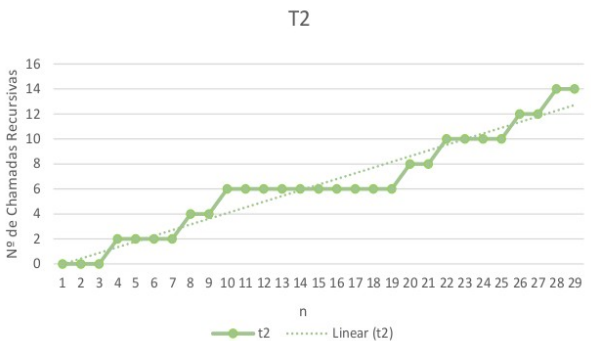
$$T_3(n)=\begin{cases}n, se\ n=0,1,2\\ 2*T_3\left(\frac{n}{3}\right)+n, se\ n\ é\ múltiplo\ de\ 3\\ T_3\left(\lfloor \frac{n}{3} \rfloor\right)+T_3\left(\lceil \frac{n}{3} \rceil\right)+n, caso\ contrário\end{cases}$$

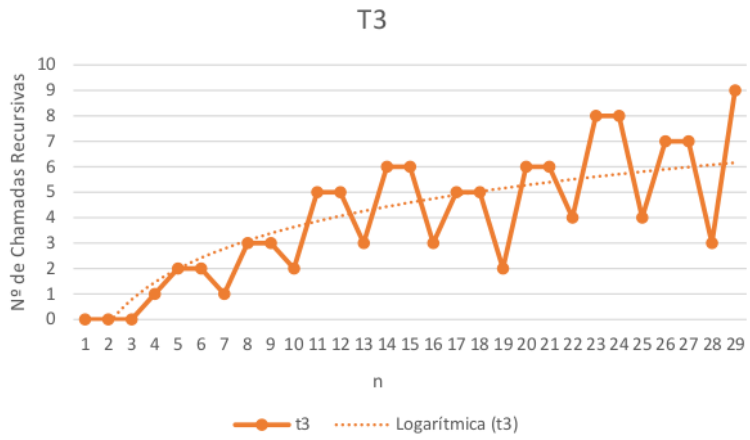
Deve utilizar **aritmética inteira**: n/3 é igual a $\lfloor \frac{n}{3} \rfloor$ e (n+2)/3 é igual a $\lceil \frac{n}{3} \rceil$.

- **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?

Para efetuar uma analise melhor e mais facilmente, fiz três gráficos, tendo então estabelecido para a função $T_1(n)$ ordem de complexidade logarítmica $(O(\log(n)))$, para a função $T_2(n)$ ordem de complexidade linear O(n) e para a função $T_3(n)$ ordem de complexidade logarítmica $(O(\log(n)))$.
Isto quer dizer que tanto $T_1(n)$ como $T_3(n)$ pertencem quando muito à ordem de complexidade $(O(\log(n)))$ e que $T_2(n)$ pertence quando muito à ordem de complexidade O(n).







- Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função $T_1(n)$**.** 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**.

C(n) = número de invocações sucessivas de$T_1(n)$
C(0) = 0 → caso base

Expressão recorrente e dada por:

$$C(n)=\begin{cases} 0, se\ n=0 \\ C\left(\lfloor \frac{n}{3} \rfloor\right)+1, se\ n>0 \end{cases}$$

$C(n)=1+C(\lfloor (n/3) \rfloor)=1+1+C(\lfloor (n/3/3) \rfloor)=1+1+1+C(\lfloor n/27 \rfloor)$
Para o caso particular de $n=3^k$
$C\left(\lfloor \frac{n}{3} \rfloor\right)=C\left(\frac{n}{3}\right)$ $C(n)=k+C(n/3^k)$
Se escolher-mos um k para que $n/3^k=1 \equiv k=\log_3(n)$
$C(n)=k+C(1) \equiv C(n)=1+\log_3(n)$, logo concluímos que o algoritmo é de complexidade logarítmica, $O(\log_3(n))$.

Este resultado pode ser confirmado pelo Teorema Mestre, pois a = 1, b=3, c=1, f(n)=1 → d=0, logo $a=b^d \equiv 1=3^0$. Assim verificamos que a complexidade é logarítmica, $O(n^d \log_3(n))=O(n^0 \log_3(n))=O(\log_3(n))$.

Através da expressão $C(n)=1+\log_3(n)$ podemos obter o número de chamadas recursivas do algoritmo podemos ver que para:
n = 0 → C(0) = 0 n = 1 → C(1) = 1 n = 2 → C(2) = 1 n = 3 → C(3) = 2
n = 4 → C(4) = 2 n = 9 → C(9) = 3 n = 18 → C(18) = 3 n = 27 → C(27) = 4

Comparando a ordem de complexidade estabelecida anteriormente com a ordem de complexidade obtida $(O(\log_3(n)))$, podemos dizer que esta complexidade é mais específica. A complexidade que obtive aqui não podia ser superior à estabelecia anteriormente, tendo sido verificado.

Tabela:

n	$T_1(n)$	Nº de Chamadas Recursivas	$T_2(n)$	Nº de Chamadas Recursivas	$T_3(n)$	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 $T_2(n)$**. Considere o caso particular** $n=3^k$ **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**.

C(n) = número de invocações sucessivas de$T_2(n)$
C(0) = C(1) = C(2) = 0 → casos base
Triângulo de Pascal → $2^0+2^1+2^2=2^3-1$e $2^1+2^2=2^3-2$
Para o caso particular $n=3^k \equiv k=\log_3(n)$
A expressão recorrente é dada por:

$$C(n)=\begin{cases} 0, se\, n=0,1,2 \\ C\left(\lfloor \frac{n}{3} \rfloor\right)+C\left(\lceil \frac{n}{3} \rceil\right)+2, se\, n>2 \end{cases}$$

Para este caso particular $n=3^k$ sabemos que $C\left(\lfloor \frac{n}{3} \rfloor\right)=C\left(\lceil \frac{n}{3} \rceil\right)$então:

$C\left(\lfloor \frac{n}{3} \rfloor\right)+C\left(\lceil \frac{n}{3} \rceil\right)+2=2\times C\left(\frac{n}{3}\right)+2$ ou seja:

$$C(n)=\begin{cases} 0, se\, n=0,1,2 \\ C\left(\lfloor \frac{n}{3} \rfloor\right)+C\left(\lceil \frac{n}{3} \rceil\right)+2, se\, n>2 \end{cases} \equiv C(n)=\begin{cases} 0, se\, n=0,1,2 \\ 2\times C\left(\frac{n}{3}\right)+2, se\, n>2 \end{cases}$$

Usando o desenvolvimento telescópico temos:
$C(n)=2+2\times C(n/3)=2+2\left(2+2\left(C(n/9)\right)\right)=2+4+4C(n/9)=2^1+2^2+2^2C(n/9)=$
$=2^3-2+2^2C(n/9)=2\times\left(2^k-1\right)+2^k\left(C\left(n/3^k\right)\right)$
$C(n)=2\times\left(2^{(\log_3(n))}-1\right)+2^k\times C(1)=2\times\left(2^{(\log_3(n))}-1\right)=2n^{(\log_3(2))}-2$, logo este algoritmo tem ordem de complexidade polinomial $O\left(n^{(\log_3(2))}\right)$.

Através do Teorema Mestre podemos confirmar este resultado, pois a = 2, b = 3, c = 0, f(n) = 2 → d = 0, logo $a>b^d \equiv 2>3^0$ o que significa que a ordem de complexidade é polinomial $O\left(n^{(\log_3(2))}\right)$.

Através da expressão $C(n)=2n^{\log_3(2)}-2$ podemos obter o número de chamadas recursivas do algoritmo podemos ver que para:
n = 0,1,2 → C(0) = C(1) = C(2) = 0 n = 3 → C(3) = 2 n = 4 → C(4) = 2
n = 7 → C(7) = 4 n = 9 → C(9) = 6 n = 27 → C(27) = 14
n = 18 → C(18) = 10≠6, isto acontece porque 18 não é potência de 3 e porque a expressão calculada é só para valores que sejam potências de 3, pois só ai é que $C\left(\lfloor \frac{n}{3} \rfloor\right)=C\left(\lceil \frac{n}{3} \rceil\right)$.

Comparando a ordem de complexidade estabelecida anteriormente com a ordem de complexidade obtida $O\left(n^{(\log_3(2))}\right)$, podemos dizer que esta complexidade é mais especifica. A complexidade que obtive aqui não podia ser superior à estabelecia anteriormente, tendo sido verificado.

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

Como $n^{(\log_3(2))}$, é uma função suave ($n^{(\log_3(2))}$ é eventualmente decrescente e $2\times n^{(\log_3(2))}$ também pertence a $O\left(n^{(\log_3(2))}\right)$), como C(n) é uma função eventualmente não decrescente e como a função $T_2(n)$ tem complexidade polinomial para os valores de n potências de 3 (b = 3) onde $3\geq 2$, então pela regra da suavidade, concluímos que é possível generalizar a ordem de complexidade que obtivemos anteriormente para todo o n.

- Obtenha uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função $T_3(n)$.

C(n) = número de invocações sucessivas de $T_3(n)$
Expressão recorrente é dada por:

$$C(n)=\begin{cases} 0, se\ n=0,1,2 \\ 2\times C\left(\frac{n}{3}\right)+1, se\ n\ é\ múltiplo\ de\ 3 \\ C\left(\lfloor \frac{n}{3} \rfloor\right)+C\left(\lceil \frac{n}{3} \rceil\right)+2, caso\ contrário \end{cases}$$

- **Considere o caso particular** $n=3^k$ **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**.

C(0) = C(1) = C(2) = 0 → casos base

Para o caso particular $n=3^k$ usamos a equação $C(n)=2\times C\left(\frac{n}{3}\right)+1$, pois todas as potências de 3 são múltiplos de 3.

$C(n)=2\times C(n/3)+1=2\left(2C(n/9)+1\right)+1=4\times C(n/9)+2+1=2^k C\left(n/3^k\right)+k+1$

$C(n)=2^k C(1)+k+1=k+1=\log_3(n)+1$, logo concluímos que o algoritmo é de complexidade logarítmica, $O\left(\log_3(n)\right)$.

Usando o Teorema Mestre, sabemos que a = 1, b = 3, c = 0 , f(n) = 1 → d = 0, logo $a=b^d \equiv 1=3^0$. Assim segundo este teorema, concluímos que algoritmo é de complexidade logarítmica, $O\left(\log_3(n)\right)$, tal como quando usamos o desenvolvimento telescópico.

Através da expressão $C(n)=\log_3(n)+1$ podemos obter o número de chamadas recursivas mais um (Nº de chamadas +1) do algoritmo podemos ver que para:
n = 0,1,2 → C(0) = C(1) = C(2) = 0 n = 3 → C(3) = 2 (1+1)
n = 9 → C(9) = 3 (2+1) n = 27 → C(27) = 4 (3+1)
Podemos ver também que a expressão não é valida para números que não sejam múltiplos de 3 (pois para isso teria de ser usado outro ramo):
n = 17 → C(17)=3 quando devia de ser 5 n = 22 → C(22)=3 quando devia de ser 8

Comparando a ordem de complexidade estabelecida anteriormente com a ordem de complexidade obtida $O(\log_3(n))$, podemos dizer que esta complexidade é mais especifica. A complexidade que obtive aqui não podia ser superior à estabelecia anteriormente, tendo sido verificado.

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

A ordem de complexidade anterior não se pode generalizar para todo o n, pois para o caso particular $n=3^k$ usamos a equação $C(n)=2\times C\left(\frac{n}{3}\right)+1$. Esta equação só e usada quando n for múltiplo de 3. Para obter a ordem de complexidade para outros casos (n não múltiplo de 3 e n diferente de 0, 1 ou 2) teríamos de usar a equação $C\left(\lfloor\frac{n}{3}\rfloor\right)+C\left(\lceil\frac{n}{3}\rceil\right)+2$, prosseguindo com o desenvolvimento telescópico ou o Teorema Mestre.

- Atendendo às **semelhanças entre** $T_2(n)$ **e** $T_3(n)$ estabeleça uma **ordem de complexidade para** $T_3(n)$**. Justifique.**

Atendendo às semelhanças entre $T_2(n)$ e $T_3(n)$, podemos observar em relação ao esforço computacional que $T_2(n)$ é majorante $T_3(n)$. Isto significa que a ordem de complexidade de $T_3(n)$ não pode ser superior que à de $T_2(n)$ uma vez que a ordem de complexidade de $T_3(n)$ está limitada pela ordem de complexidade de $T_2(n)$.