GUIÃO 05 – ANÁLISE DA COMPLEXIDADE DE ALGORITMOS RECURSIVOS

Implemente os seguintes **algoritmos recursivos**, sem recorrer a funções de arredondamento (**floor** e **ceil**). Note que, considerando o quociente da divisão inteira, temos que n/2 é igual a $\left\lfloor \frac{n}{2} \right\rfloor$ e (n+1)/2 é igual a $\left\lceil \frac{n}{2} \right\rceil$.

Determine o **número de chamadas recursivas** executadas por cada função.

- $T(n) = \begin{cases} 1, se\ n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n, \text{se } n > 1 \end{cases}$
- $T(n) = \begin{cases} 1, se\ n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n, \text{se } n > 1 \end{cases}$
- $T(n) = \begin{cases} 1, se\ n = 1 \\ T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n, \text{se } n \text{ é impar} \\ 2 \times T\left(\frac{n}{2}\right) + n, \text{se } n \text{ é par} \end{cases}$

Preencha a tabela, com o valor do resultado da função e o número de chamadas recursivas efetuadas, para os sucessivos valores de n, por exemplo, até 32 ou 64 ou 128.

N	1ª Função (N)	Nº de Chamadas	2ª Função (N)	Nº de Chamadas	3ª Função (N)	Nº de Chamadas
1	1	0	1	0	1	0
2	3	1	4	2	4	1
3	4	1	8	4	8	3
4	7	2	12	6	12	2
5	8	2	17	8	17	6
6	10	2	22	10	22	4
7	11	2	27	12	27	7
8	15	3	32	14	32	3
9	16	3	38	16	38	10
10	18	3	44	18	44	7
11	19	3	50	20	50	12
12	22	3	56	22	56	5
13	23	3	62	24	62	13
14	25	3	68	26	68	8
15	26	3	74	28	74	12

Analisando os dados da tabela, qual é a **ordem de complexidade** de cada algoritmo?

Algoritmo 1: log_2 n Algoritmo 2: 2(n-1)

Determine formalmente a **ordem de complexidade dos dois primeiros algoritmos**, obtendo **expressões matemáticas** exatas e simplificadas.

Algortimo 1: Θ(log_2 n) e Algoritmo 2: Θ(n)

No caso do **terceiro algoritmo** indique para que valores de N se obtém o **melhor e o pior caso** e faça a respetiva análise da complexidade.

O melhor caso é 2^k, gera sempre par. O pior caso é um numero "muito impar"

Nº Bits	Nº Pior	N3
2	3	3
3	7	7
4	13	13

O melhor caso é quando n é 2^k, pois vai sempre pelo caminho par, gera poucas recursões. Se tiver um numero impar, que leva-nos sempre para caminhos impares, esse é o pior caso.

Nº Bits	Nº Pior	N3
5	27	23
6	53	39
7	107	65