Ficha: complexidade computacional

1) Para as máquinas de turing, MT1, MT2, MT3 do link

https://drive.google.com/file/d/1PIvZ0zz0svztWLSzHVB5Kc4jfIcT2Wtu/view?usp=sharing

preencha o quadro que se segue

https://docs.google.com/spreadsheets/d/1pNDANIIhqXYsAYFhWt2W-X6PzsISMDfEmZe1yxFGiX8/edit?usp=sharing

Tabela de Computação para a ordem MT1, MT2, MT3

|w|=1	T(w)	S(w)	|w|=2	T(w)	S(w)	|w|=3	T(w)	S(w)	|w|=4	T(w)	S(w)
0			00			000			0000		
1			01			001			0001		
T(1) =			10			010			0010		
S(1) =	complexidade pelo pior caso		11			011			0011		
média-T(1) =			T(2) =			100			0100		
média-S(1) =	complexidade pelo caso médio		S(2) =	complexidade pelo pior caso		101			0101		
			média-T(2) =			110			0110		
			média-S(2) =	complexidade pelo caso médio		111			0111		
						T(3) =			1000		
						S(3) =	complexidade pelo pior caso		1001		
						média-T(3) =			1010		
						média-S(3) =	complexidade pelo caso médio		1011		
									1100		
									1101		
									1110		
									1111		
									T(4) =		
									S(4) =	complexidade pelo pior caso	
									média-T(4) =		
									média-S(4) =	complexidade pelo caso médio	

2) Tendo em atenção o comportamento computational de cada uma das máquinas de Turing: MT4, MT5, …, MT8;

https://docs.google.com/spreadsheets/d/1cZC7tU24SzlWcDKe_BLa3xdQwVSPQ2EafDxiiIgfx_A/edit?usp=sharing

		MT4				MT5				MT6				MT7				MT8			
		T(n)	média-T(n)	S(n)	média-S(n)	T(n)	média-T(n)	S(n)	média-S(n)	T(n)	média-T(n)	S(n)	média-S(n)	T(n)	média-T(n)	S(n)	média-S(n)	T(n)	média-T(n)	S(n)	média-S(n)
	|w|=1	2	1	2	1	3	2	1	1	5	5	3	2	3	3	3	1	2	1	8	3
	|w|=2	3	2	4	2	6	4	2	2	5	5	9	4	4	3	9	2	4	2	11	6
	|w|=3	4	3	8	4	9	6	3	3	5	5	27	8	3	3	27	6	12	6	14	9
	|w|=4	5	4	16	8	12	8	4	4	5	5	81	16	4	3	81	24	48	24	17	12
	|w|=5	6	5	32	16	15	10	5	5	5	5	243	32	3	3	243	120	240	120	20	15
	|w|=6	7	6	64	32	18	12	6	6	5	5	729	64	4	3	729	720	1440	720	23	18
	|w|=7	8	7	128	64	21	14	7	7	5	5	2187	128	3	3	2187	5040	10080	5040	26	21
	.....	.....		.....		.....	.....	.....	.....	.....	.....	.....	.....	.....		.....	.....	.....	.....	.....	.....
	|w|=n	n+1	n	2^n	2^(n-1)	3n	2n	n	n	5	5	3^n	2^n	3, se n par; 4, se n ímpar	3	3^n	n!	2*(n!)	n!	3n+5	3n
	|w|=n+1	n+2	n+1	2^(n+1)	2^n	3(n+1)	2(n+1)	n+1	n+1	5	5	3^(n+1)	2^(n+1)	3, se n par; 4, se n ímpar	3	3^(n+1)	(n+1)!	2*((n+1)!)	(n+1)!	3(n+1)+5	3(n+1)
	.....	.....		.....		.....	.....	.....	.....	.....	.....	.....	.....	.....		.....	.....	.....	.....	.....	.....
	linguagem reconhecida	L1				L2				L3				L4				L5			
constante	O(1)																				
linear	O(n)																				
polinomial	O(n)																				
polinomial	O(n^2)																				
polinomial	O(n^3)																				
factorial	O(n!)																				
exponencial	O(2^n)																				
exponencial	O(3^n)																				
		Quadro de complexidade computacional																			

preencha o quadro da complexidade computacional que se segue, o big O:
i) para o pior caso, no tempo T(n) e no espaço S(n),
ii) para o caso médio, no tempo médio-T(n) e no espaço médio-S(n)

3) No link, https://wiki.python.org/moin/TimeComplexity, pode ser visto o tempo computacional da implementação de alguns algoritmos em Python, pelo pior caso e pelo caso médio. Preencha o quadro.

		big O, no tempo	
Sr.No.	Methods with Description	complexidade pelo pior caso	complexidade pelo caso médio
1	list.append(obj) Appends object obj to list.		
2	list.clear() Clears the contents of list.		
3	list.copy() Returns a copy of the list object.		
4	list.extend(seq) Appends the contents of seq to list		
5	list.index(obj) Returns the lowest index in list that obj appears		
6	list.insert(index, obj) Inserts object obj into list at offset index		
7	list.pop(obj=list[-1]) Removes and returns last object or obj from list		
8	list.remove(obj) Removes object obj from list		
9	list.sort([func]) Sorts objects of list, use compare func if given		

4) Quando uma máquina de turing, ML, reconhece a linguagem L. Isto significa que a máquina avalia um problema de decisão para cada palavra, w:
w pertence a L ou w não pertence a L

Sejam L'1, L'2, L'3, L'4 linguagem de um alfabeto binário se:

i) foi possível desenhar uma máquina de turing, MT1 tal que MT1 reconhece L'1, em que T(n)=n.

ii) foi possível desenhar uma máquina de turing, MT2, tal que MT2 reconhece L'2, e T(n)=2^n. Provou-se que não existe nenhuma MT determinística que reconheça L'2 e tal que T(n)< 2^n. Foi provado que o problema do Graph Coloring é redutível ao problema L'2.

iii) foi possível desenhar uma máquina de turing não determinística, MT3, tal que MT3 reconhece L'3, em que T(n)=n^3. Tentou-se encontrar uma máquina de turing determinística que reconhece-se L'3, mas não se conseguiu desenhar (o que não significa que não seja possível).

iv) foi possível desenhar uma máquina de turing não determinística, MT4, tal que MT4 reconhece L'4, em que T(n)=n^5. foi também possível desenhar uma máquina de turing não determinística, MT5, tal que MT5 reconheceu L'4, em que T(n)=n^5.

v) foi possível desenhar uma máquina de turing não determinística, MT6, tal que MT6 reconhece L'5, e em que T(n)=2n. Foi provado que o problema Knapsack é redutível a L'5.

	No tempo			
	Classe P	Classe NP	Classe NP-completo	Classe NP-Hard
L'1				
L'2				
L'3				
L'4				
L'5				