Complexidade de Algoritmos

Lista de exercício 2 - 28/08/24

Grupo: Marcelo Camilo Gomes e Júlio César Toscano Ximenes Júnior

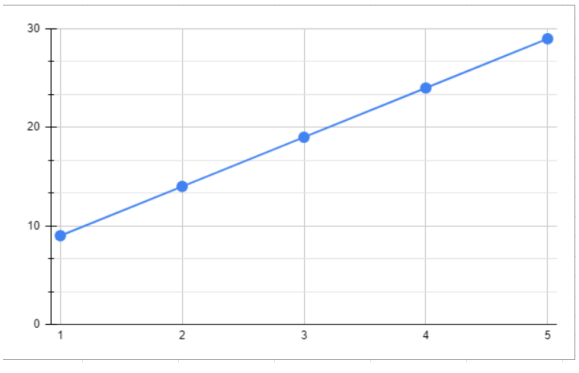
1-) Questão 1

Soma de Elementos de um Array

```
def soma_elementos(arr):
    soma = 0
    for i in range(len(arr)):
        soma += arr[i]
    return soma
```

1-) def soma_elementos		
Constante	Primitiva	Qtd exec
C1	soma = 0	1
C2	i = 0	1
C3	i < len	n+1
C4	i++	n
C5	soma +	n
C6	soma =	n
C7	arr[i]	n
C8	return soma	1

T(n) = 5n+4	
n	T(n)
1	9
2	14
3	19
4	24
5	29



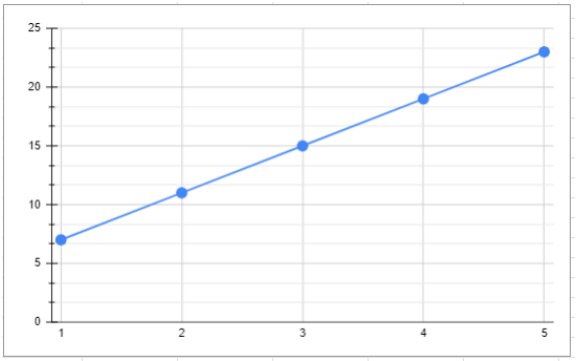
2-) Questão 2

Busca Linear em um Array

```
def busca_linear(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1
```

Constante	Primitiva	Qtd exec
C1	i =	1
C2	i <	n+1
C3	i++	n
C4	arr[i]	n
C5	"=="	n
C6	return i	0
C7	return -1	1

T(n) = 4n+3	
n	T(n)
1	7
2	11
3	15
4	19
5	23



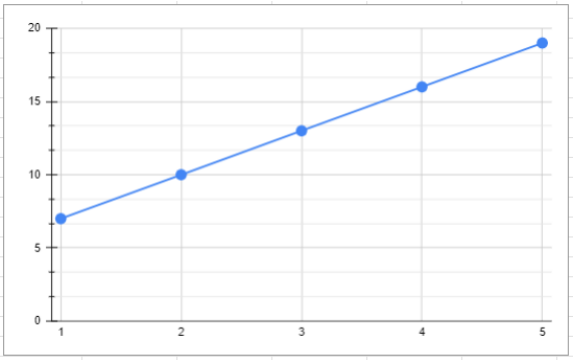
3-) Questão 3

Cópia de um Array

```
def copia_array(arr):
    novo_arr = []
    for i in range(len(arr)):
        novo_arr.append(arr[i])
    return novo_arr
```

Constante	Primitiva	Qtd exec
C1	novo_arr =	1
C2	i =	1
C3	i <	n+1
C4	i++	n
C5	arr[i]	n
C6	return	1

T(n) = 3n+4	
n	T(n)
1	7
2	10
3	13
4	16
5	19



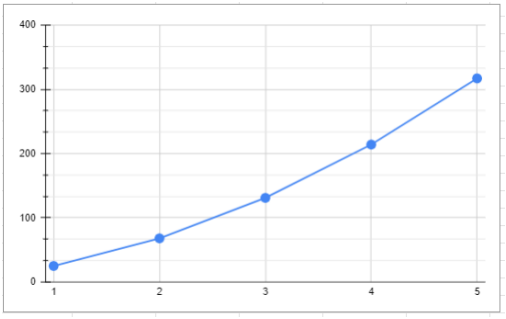
4-) Questão 4

Ordenação por Inserção (Insertion Sort)

```
def insertion_sort(arr):
  for i in range(1, len(arr)):
    chave = arr[i]
    j = i - 1
    while j >= 0 and chave < arr[j]:
      arr[j + 1] = arr[j]
      j -= 1
    arr[j + 1] = chave
  return arr
```

Constante	Primitiva	Qtd exec
C1	i =	1
C2	i <	n
C3	i++	n
C4	chave =	n
C5	arr[i]	n
C6	j =	n
C7	i - 1	n
C8	j <	n^2+n
C9	and	n^2+n
C10	chave <	n^2+n
C11	arr[j]	n^2+n
C12	j+1	n^2
C13	arr[j+1]	n^2
C14	"="	n^2
C15	arr[j]	n^2
C16	j =	n^2
C17	j - 1	n^2
C18	j + 1	n
C19	arr[j+1]	n
C20	"="	n
C21	return arr	1

$T(n) = 10n^2 + 13n + 2$	
n	T(n)
1	25
2	68
3	131
4	214
5	317



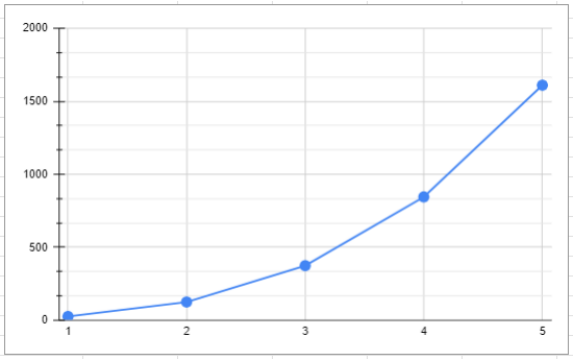
5-) Questão 5

Multiplicação de Matrizes Simples

```
def multiplica_matrizes(A, B):
    n = len(A)
    C = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                C[i][j] += A[i][k] * B[k][j]
    return C
```

Constante	Primitiva	Qtd exec
C1	n =	1
C2	_ =	1
C3	_ <	n+1
C4	_ ++	n+1
C5	[0]	n+1
C6	c =	1
C7	i =	1
C8	i <	n+1
C9	i++	n
C10	j =	n
C11	j <	n^2+n
C12	j ++	n^2
C13	k =	n^3
C14	k <	n^3+n^2
C15	k ++	n^3
C16	A[i]	n^3
C17	A[i][k]	n^3
C18	B[k]	n^3
C19	B[k][j]	n^3
C20	*	n^3
C21	"+"	n^3
C22	C[i]	n^3
C23	C[i][j]	n^3
C24	C =	n^3
C25	return	1

$T(n) = 12n^3 + 3n^2 + 6n + 5$	
n	T(n)
1	26
2	125
3	374
4	845
5	1610



6-) Questão 6

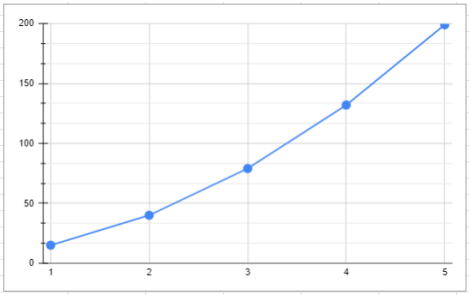
Contagem de Pares em um Array

```
def conta_pares(arr):
    count = 0
    for i in range(len(arr)):

        for j in range(i + 1, len(arr)):
            if arr[i] == arr[j]:
                count += 1
    return count
```

Constante	Primitiva	Qtd exec
C1	count =	1
C2	i =	1
C3	i <	n+1
C4	i++	n
C5	j =	n
C6	i+	n
C7	j <	n^2
C8	j ++	n^2
C9	arr[i]	n^2
C10	"=="	n^2
C11	arr[j]	n^2
C12	count =	n^2
C13	count +	n^2
C14	return	1

$T(n) = 7n^2 + 4n + 4$	
n	T(n)
1	15
2	40
3	79
4	132
5	199



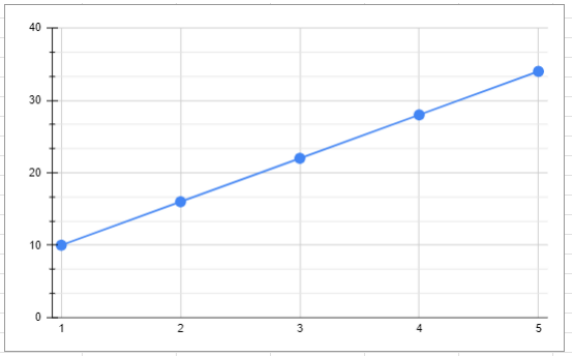
7-) Questão 7

Contagem de Ocorrências de um Elemento em um Array

```
def conta_ocorrencias(arr, x):
    count = 0
    for i in range(len(arr)):
        if arr[i] == x:
            count += 1
    return count
```

Constante	Primitiva	Qtd exec
C1	count =	1
C2	i =	1
C3	i <	n+1
C4	i++	n+1
C5	arr[i]	n
C6	"=="	n
C7	count =	n
C8	count ++	n
C9	return count	1

T(n) = 6n + 4	
n	T(n)
1	10
2	16
3	22
4	28
5	34



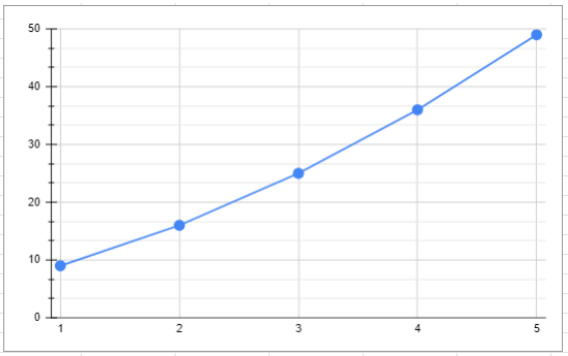
8-) Questão 8

Verificação de Unicidade em um Array

```
def verifica_unicidade(arr):
    elementos_vistos = set()
    for i in range(len(arr)):
        if arr[i] in elementos_vistos:
            return False
        elementos_vistos.add(arr[i])
    return True
```

Constante	Primitiva	Qtd exec
C1	elementos_vistos =	1
C2	i =	1
C3	i <	n+1
C4	i++	n
C5	arr[i]	n
C6	in	n^2
C7	return False	0
C8	arr[i]	n
C9	return True	1

$T(n) = n^2+4n+4$	
n	T(n)
1	9
2	16
3	25
4	36
5	49



9-) Questão 9

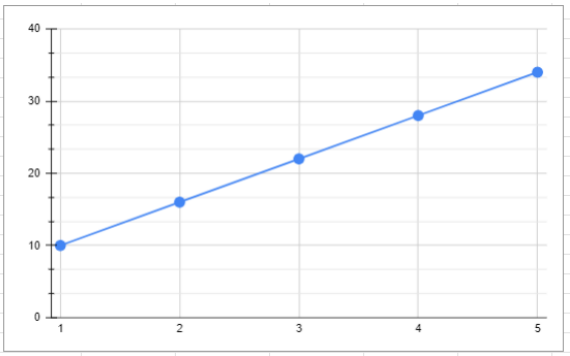
Maior Elemento em um Array

```
def maior_elemento(arr):
    maior = arr[0]
    for i in range(1, len(arr)):

        if arr[i] > maior:
            maior = arr[i]
    return maior
```

Constante	Primitiva	Qtd exec
C1	maior =	1
C2	arr[0]	1
C3	i =	1
C4	i <	n
C5	i++	n
C6	arr[i]	n
C7	>	n
C8	maior =	n
C9	arr[i]	n
C10	return maior	1

T(n) = 6n+4	
n	T(n)
1	10
2	16
3	22
4	28
5	34



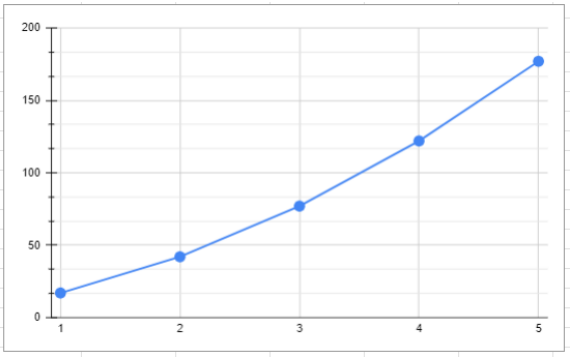
10-) Questão 10

Ordenação por Seleção (Selection Sort)

```
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i+1, len(arr)):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

Constante	Primitiva	Qtd exec
C1	i =	1
C2	i <	n+1
C3	i++	n
C4	min_idx =	n
C5	j =	n
C6	j <	n^2
C7	j++	n
C8	arr[j]	n^2
C9	<	n^2
C10	arr[min_idx]	n^2
C11	min_idx =	n^2
C12	arr[i]	n
C13	arr[min_idx]	n
C14	"="	n
C15	arr[min_idx]	n
C16	arr[i]	n
C17	return arr	1

$T(n) = 5n^2+10n+2$	
n	**T(n)**
1	17
2	42
3	77
4	122
5	177



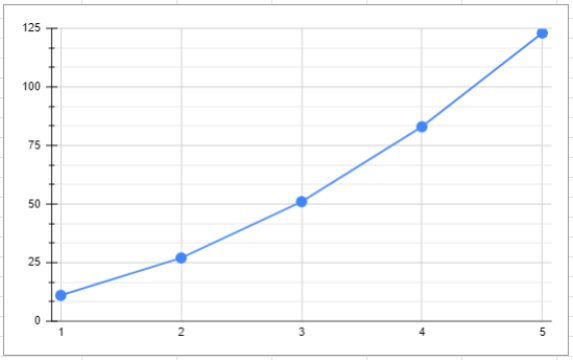
11-) Questão 11

Verificação de Duplicatas em um Array

```
def verifica_duplicatas(arr):
    for i in range(len(arr)):
        for j in range(i + 1, len(arr)):
            if arr[i] == arr[j]:
                return True
    return False
```

Constante	Primitiva	Qtd exec
C1	i =	1
C2	i >	n+1
C3	i++	n
C4	j =	n
C5	j <	n^2
C6	j++	n
C7	arr[i]	n^2
C8	"=="	n^2
C9	arr[j]	n^2
C10	return True	0
C11	return False	1

$T(n) = 4n^2+4n+3$	
n	T(n)
1	11
2	27
3	51
4	83
5	123



12-) Questão 12

Soma de Todos os Pares de Elementos em um Array

```
def soma_pares(arr):

    soma = 0
    for i in range(len(arr)):
        for j in range(i + 1, len(arr)):
            soma += arr[i] + arr[j]
    return soma
```

Constante	Primitiva	Qtd exec
C1	soma =	1
C2	i =	1
C3	i <	n+1
C4	i++	n
C5	j =	n
C6	j <	n^2
C7	j++	n
C8	soma =	n^2
C9	soma +	n^2
C10	arr[i]	n^2
C11	"="	n^2
C12	arr[j]	n^2
C13	return soma	1

$T(n) = 6n^2+4n+4$	
n	T(n)
1	14
2	36
3	70
4	116
5	174