0.1 Práctica Complejidad

TUIA - Programación 2

Nociones de Complejidad

A continuación se plantean una serie de funciones implementando algoritmos. Realice un análisis de los algoritmos implementados para determinar el orden de su complejidad temporal.

Ejercicio 0

```python
def condicional0(num: int) -> int:
    if num > 10:
        print(f"{num} es mayor que 10")
        aux = num % 2
    else:
        print(f"{num} es menor que 10")
        aux = num // 2
    return aux
```

Ejercicio 1

```python
def condicional1(num: int) -> int:
    if num > 10:
        print(f"{num} es mayor que 10")
        aux = num % 2
        if aux == 0:
            print(f"{num} es divisible por 2")
        else:
            print(f"{num} no es divisible por 2")
            aux = num * num * num
    else:
        print(f"{num} es menor que 10")
        aux = num // 2
        producto = num * 2
        if producto % 3 == 0:
            print("producto es divisible por 3")
    return aux
```

Ejercicio 2

```python
def ciclo0(n: int) -> None:
    lista = list(range(n))
```

```
    m = n // 2
    for i in range(0, n):
        print("inicio ciclo j")
        for j in range(m):
            print("inicio ciclo k")
            for k in range(0, 3):
                print(lista[k])
```

Ejercicio 3

```
def ciclo1(n: int) -> None:
    lista = list(range(n))
    m = n % 2
    for i in range(0, n):
        print("inicio ciclo j")
        for j in range(m):
            print("inicio ciclo k")
            for k in range(0, 3):
                print(lista[k])
```

Ejercicio 4

```
def ciclo2(n: int, m: int) -> None:
    lista = [1, 2, 3, 4]
    for i in range(0, n):
        print("inicio ciclo j")
        for j in range(0, m):
            print("inicio ciclo k")
            for k in range(0, 3):
                print(lista[k])
```

Ejercicio 5

```
def ciclo3(n: int, m: int, b: bool) -> None:
    for i in range(0, n):
        if b:
            print("inicio ciclo j")
            for j in range(0, m):
                print(n + j)
        else:
            print("inicio ciclo k")
            for k in range(1000):
                print(n)
```

Ejercicio 6

```
def ciclo4(n: int) -> None:
    if n > 10:
        print("Tabla de multiplicar")
```

```
        for i in range(1, 11):
            print(i, n * i)
    else:
        print(n // 2, n * 2)
```

Ejercicio 7

```
def ciclo5() -> None:
    numero = int(input("Ingrese un valor"))
    lista = []
    while numero != 0 and len(lista) < 10000:
        lista.append(numero)
        numero = int(input("Ingrese un valor"))

    for i in range(len(lista)):
        print(lista[i])
```

Ejercicio 8

La siguiente función recibe dos listas ordenadas y devuelve una sola lista ordenada que contiene los mismos elementos que la concatenación de las dos anteriores, pero en orden.

```
def mezcla_de_listas(L1: list[int], L2: list[int]) -> list[int]:
    # precond: L1 y L2 estan ordenadas
    i, j = 0, 0
    L3 = []
    while i < len(L1) and j < len(L2):
        if L1[i] < L2[j]:
            L3.append(L1[i])
            i += 1
        else:
            L3.append(L2[j])
            j += 1

    if i == len(L1):
        for k in range(j, len(L2)):
            L3.append(L2[k])
    else:
        for k in range(i, len(L1)):
            L3.append(L1[k])
    return L3
```

Ejercicio 9

La siguiente función recibe una matriz `M1` de tamaño `n x n` y calcula la traza de la misma:

```
def traza(M1: list[list[int]]) -> int:
    # precond: M1 es cuadrada
    traza = 0
    for i in range(0, n):
        traza += M1[i][i]
    return traza
```

Ejercicio 10

La siguiente función determina si un número es primo:

```
def primo(numero: int) -> bool:
    # Precond: numero es no negativo
    if numero <= 1:
        return False
    else:
        cont = 0
        for i in range(2, numero + 1):
            if numero % i == 0:
                cont += 1
        if cont == 1:
            return True
        else:
            return False
```

Ejercicio 11

La siguiente función calcula el factorial de forma iterativa:

```
def factorial(numero: int):
    fact = 1
    for i in range(1, numero + 1):
        fact *= 1
    return fact
```

Ejercicio 12

La siguiente función calcula el *n-ésimo* número de Fibonacci de forma iterativa:

```
def fibo(numero: int) -> int:
    fib0 = 0
    fib1 = 1
    if numero == 0:
        return fib1
    elif numero == 1:
        return fib2
    else:
        resultado = 0
        for i in range(2, numero + 1):
            resultado = fib1 + fib2
            fib1 = fib2
            fib2 = resultado
        return resultado
```