Diseño y Análisis de Algoritmos **Name:** *Jaime Andres Torres Bermejo*
Tarea 1
16/02/2023

Parte 1: Algoritmos a Analizar.

Para los siguientes problemas:

1. Proponga un algoritmo que solucione el problema. (Python)
2. roponga una tabla que enumere las operaciones que se puede suponer que se ejecutan en tiempo constante
3. Derive la función de costo, T(n), para el algoritmo que propuso, basado en las operaciones constantes del punto anterior.
4. En caso de que la función sea una ecuación de recurrencia, resolver la ecuación.
5. Determinar el orden de complejidad del algoritmo.

Dada una matriz cuadrada de números enteros sumar los elementos de la diagonal principal.

1. Código Fuente.

```
def matrixsum(nums):
i = len(nums)
j = 0
ret = 0
while j ¡ i:
    ret += nums[j][j]
    j += 1
return ret
```

2. Tabla de costos

C_1	Asignación de Variables
C_2	Consultar 'len()'
C_3	Sumar variable
C_4	Comparación

3. Derivar la función de costo para el algoritmo propuesto.

$T(n) = 3C_1 + C_2 + T(n(C_4 + 2C_1))$

4. esta NO es una ecuación de recurrencia.

5. Determinar el orden de complejidad del algoritmo.

$$3C_1 + C_2 + T(n(C_4 + 2C_1))$$

$$< \forall c \exists \mathbb{N} | cf = O(f) >$$

$$T(n) = O(n)$$

El algoritmo es de complejidad O(n)

Dada una matriz de números enteros y un numero entero, se debe devolver True si el numero se encuentra en la matriz y False en otro caso.

1. Código Fuente.

```
def matrixsearch(nums, target):
    for i in range(len(nums)):
        for j in range(len(nums[0])):
            if nums[i][j] == target:
                return True
    return False
```

2. Tabla de costos

C_1	Consultar 'len()'
C_2	Comparación

3. Derivar la función de costo para el algoritmo propuesto.

$$T(n) = T(n(C_1)) + T(m(C_1 + C_2))$$

4. esta NO es una ecuación de recurrencia.

5. Determinar el orden de complejidad del algoritmo.

$$T(n(C_1)) + T(m(C_1 + C_2))$$

$$< \forall c \exists \mathbb{N} | cf = O(f) >$$

$$T(n) = O(n) + O(m)$$

$$< Factorizacion >$$

$$T(n) = O(n + m)$$

El algoritmo es de complejidad O(n+m), con n y m siendo ancho y largo de la matriz.

Parte 2: Soluciones de las siguientes ecuaciones de recurrencia.

determinar el orden de complejidad, para las ecuaciones no lineales debe:

1. Proponer una cota usando arboles de recurrencia.
2. Demostrar la cota propuesta usando el método de sustitución.
3. De ser posible compruebe su respuesta usando el método maestro

I.

$2T(n) - T(n-1), T(0) = 1, T(1) = 5$

Arbol de Recurrencia

```
2T(n) - T(n-1)

.
‘-- T(0) = 1
    —- T(1) = 5
    ‘-- T(2) = 3
        —- T(2) = 3
        ‘-- T(3) = 4
--¿ O(n)
```

Problema de Strassen

El profesor Marco desea desarrollar un algoritmo de multiplicación de matrices que sea asintóticamente más rápido que el algoritmo de Strassen. Su algoritmo utilizará el método de dividir y conquistar, dividiendo cada matriz en trozos de tamaño n/4 x n/4, y los pasos de dividir y combinar juntos tardarán O(n2). Necesita determinar cuántos subproblemas tiene que crear su algoritmo para vencer al algoritmo de Strassen. Si su algoritmo crea un subproblema, entonces la recurrencia para el tiempo de ejecución T(n) se convierte en T(n) = aT (n/4) + O(n2) ¿Cuál es el mayor valor entero de a para el que el algoritmo del profesor Marco sería asintóticamente más rápido que el algoritmo de Strassen?

Solución