# Universidad Nacional Autónoma de México
# Facultad de Ciencias
# Estructuras de Datos 2022-2
# Tarea 01: Cálculo de Complejidades

Pedro Ulises Cervantes González
confundeme@ciencias.unam.mx

Yessica Janeth Pablo Martínez,
yessica_j_pablo@ciencias.unam.mx

Jorge Macías Gómez
jorgemacias@ciencias.unam.mx

Fecha de entrega: 07 de Marzo del 2022
Hora límite de entrega: 23:59

*Realiza lo que se te pide en cada uno de los siguientes ejercicios.

## 1. Ejercicio 1 (5 puntos)

.-Calcula el tiempo de ejecución en el peor de los casos para los siguientes métodos y determina su complejidad.

**Problema 1**

```
problema1(A){
    suma = 0;
    posicion = 0;
    while(posicion <= 10){
        suma = suma + A[posicion];
        posicion++;
    }//end while
    return suma;
}
```

**Problema 2**

```
public static int problema2(int n){
    int x = 0;
    for(int i = 0; i <= n; i ++){
        for(int j = 0; j <= n; j++){
            x++;
        }
    }
    return x;
}
```

**Problema 3**

```
/*
 * n es un entero postivo que ademas es potencia de 2
 **/
public static int problema3(int n){
    int i = n;
    int contador = 0;
    while(i > 1){
        i = i / 2;
        contador++;
    }
    return contador
}
```

**Problema 4**

```
    //Toma en cuenta el valor de t = 5
public static int problema4(int t){
    int suma = 0;
    for(int i = 0; i < t; i++){
        suma++;
    }
    return suma;
}
```

## 2. Ejercicio 2 (5 puntos)

**.-Definición:** Sean $f(n)$ y $g(n)$ funciones de complejidad. Decimos que $f(n)$ es $O-grande$ de $g(n)$ y $g(n)$ representa una cota asintótica superior para $f(n)$ si $\exists c \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N} \cup \{0\}$ tales que $\forall_n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)$.

Demuestra cada uno de los siguientes ejercicios:

- Sea $T(n) = 2n^2 + 6n$, P.D que $T(n) = 2n^2 + 6n \in O(n^3)$
- Sea $T(n) = 3\sqrt{n} + 15n^2$, P.D que $T(n) = 3\sqrt{n} + 15n^2 \in O(n^2)$
- Sea $T(n) = 17n^3 + nlogn + 17$, P.D que $T(n) = 17n^3 + nlogn + 17 \in O(n^3)$
- Sea $T(n) = 53n + 3log(n)$, P.D que $T(n) = 53n + 3log(n) \in O(nlog(n))$
- Sea $T(n) = 17nlog_2(n) + 3n$, P.D que $T(n) = 17nlog_2(n) + 3n \in O(n^2)$

## 3. Reglas importantes

- Cumple con los lineamientos de entrega.
- **¡OjO!** En caso de tener problemas con su compañero de equipo , mandar correo con anticipación de mínimo 3 días antes de la fecha de entrega al correo: `yessica_j_pablo@ciencias.unam.mx` notificando el problema para tomar medidas sobre su compañero de trabajo.
- **No se recibirán tareas en las que estén involucrados más de dos integrantes**
- En caso de no cumplirse alguna de las reglas especificadas, se restará 0.5 puntos en tu calificación final.

**Mucho éxito**

# Respuestas

Equipo:
Dafne Bonilla Reyes
José Camilo García Ponce

## Ejercicio 1

- Problema 1:

```
problema1(A){
    suma = 0;    // 1 operacion
    posicion = 0;    // 1 operacion
    while(posicion <= 10){   // 11 + 1 operaciones
        suma = suma + A[posicion];   // 4 operaciones
        posicion++; // 3 operaciones
    }
    return suma; // 2 operaciones
}
```

En las primeras dos líneas tenemos 2 operaciones, una por asignación respectivamente. Luego, en el while tenemos $11 * (3 + 4) + 1$ operaciones. 11 operaciones para las comparaciones de desigualdad que se repiten $(3 + 4)$ veces, es decir, el número de operaciones dentro del while, más 1 operación cuando la condición no se cumple, por lo tanto, tenemos 78 operaciones. Finalmente, en la última línea tenemos 2 operaciones, una para leer el valor de la suma y otra para regresarlo. Sumando todo obtenemos 82 operaciones en total. De esta manera, la complejidad del algoritmo es $O(1)$, es decir constante, ya que no hay variables.

- Problema 2:

```
1 public static int problema2(int n){
2     int x = 0;   // 1 operacion
3     for(int i = 0; i <= n; i++){    // 1 + (n+1)  + 1 + n operaciones
4        for(int j = 0; j <= n; j++){ // 1 + (n+1)  + 1 + n operaciones
5            x++; // 3 operaciones
6        }
7     }
8     return x; // 2 operaciones
9 }
```

En la primera línea tenemos 1 operación de asignación. Luego, en el primer for tenemos $1 + (n + 1) * (\text{operaciones del for interno}) + 1 + n$ operaciones. Es decir, 1 operación de asignación, $(n + 1)$ operaciones multilpicadas por el número de operaciones del for interno, 1 operación para cuando la condición de desigualdad no se cumple y n operaciones en el postincremento de i. Después, para el segundo for tenemos $1 + (n + 1) * 3 + 1 + n$ operaciones. Analogamente al for exterior, tenemos 1 operación de asignación, $(n + 1)$ operaciones multilplicadas por las 3 operaciones que se encuentran dentro de este for, 1 operación para cuando la condición de desigualdad no se cumple y n operaciones en el postincremento de j. Así, el segundo for tiene $4n + 5$ operaciones y por consecuencia el for externo de la línea 3 tiene $1 + (n + 1)(4n + 5) + 1 + n$ que es equivalente a $4n^2 + 10n + 7$ operaciones. Finalmente, sumando todo obtenemos $4n^2 + 10n + 10$ operaciones en total. Por lo tanto, la complejidad del algoritmo es $O(n^2)$, ya que tiene una variable cuadrática.

- Problema 3:

```java
/*
 * n es un entero postivo que ademas es potencia de 2
 **/
public static int problema3(int n){
    int i = n;  // 1 operacion
    int contador = 0; // 1 operacion
    while(i > 1){   // log_2(n) + 1 operaciones
        i = i / 2;  // 3 operaciones
        contador++; // 3 operaciones
    }
    return contador // 2 operaciones
}
```

En las primeras dos líneas tenemos 2 operaciones, una por asignación respectivamente. Luego, en el while tenemos $log_2(n) * (3 + 3) + 1$ operaciones, ya que while solo realizará $x$ ciclos, donde $x$ es $2^x = n$, así dándanos $log_2(n)$, esto porque la inversa del exponencial es el logaritmo. Entonces, tenemos $log_2(n)$ multiplicado por $(3 + 3)$ operaciones que se encuentran dentro del while, más una operación para cuando la condición de desigualdad no se cumple. Además, en la última línea tenemos 2 operaciones. Finalmente, sumando todo obtenemos $6log_2(n) + 4$ operaciones en total. Por lo tanto, la complejidad del algoritmo es $O(log(n))$, ya que el tiempo de ejecución tiene un logaritmo.

- Problema 4:

```java
//Toma en cuenta el valor de t = 5
public static int problema4(int t){
    int suma = 0;   // 1 operacion
    for(int i = 0; i < t; i++){ // 1 + (t) + 1 + t operaciones
        suma++; // 3 operaciones
    }
    return suma;    // 2 operaciones
}
```

En la primera línea tenemos 1 operación de asignación. Luego, en el for tenemos $1 + (t) * (3) + 1 + t$ y como $t = 5$ entonces son $1 + 5 * (3) + 1 + 1$ operaciones. Esto es, 1 operación de asignación, 5 operación de $t$ multiplicadas por las 3 operaciones que se encuentran dentro del for ó $(t) * 3$ operaciones, 1 operación para cuando la condición de desigualdad no se cumple y $t$ operaciones de postincremento de i. Por otro lado, en la última línea tenemos 2 operaciones. Finalmente, sumando todo obtenemos 21 operaciones en total si $t = 5$ ó $4t + 5$ para cualquier $t$. Por lo tanto, su complejidad es $O(1)$ si $t = 5$, pero si $t$ es cualquier otro, tenemos una complejidad lineal $O(t)$.

## Ejercicio 2

- Problema 1:

  Sea $T(n) = 2n^2 + 6n$, P.D que $T(n) = 2n^2 + 6n \in O(n^3)$
  Veamos que $2n^2 <= 2n^3$ y $6n <= 6n^3$ para toda $n >= 1$
  Ahora, sumando las desigualdades tenemos que $2n^2 + 6n <= 2n^3 + 6n^3 = 8n^3$.
  Por lo tanto, $T(n) = 2n^2 + 6n \in O(n^3)$ con $c = 8$ y $n_0 = 1$. $\square$

- Problema 2:

  Sea $T(n) = 3\sqrt{n} + 15n^2$, P.D que $T(n) = 3\sqrt{n} + 15n^2 \in O(n^2)$
  Veamos que $3\sqrt{n} <= 3n^2$ y $15n^2 <= 15n^2$ para toda $n >= 1$
  Ahora, sumando las desigualdades tenemos que $3\sqrt{n} + 15n^2 <= 3n^2 + 15n^2 = 18n^3$.
  Por lo tanto, $T(n) = 3\sqrt{n} + 15n^2 \in O(n^2)$ con $c = 18$ y $n_0 = 1$. $\square$

- Problema 3:

  Sea $T(n) = 17n^3 + nlogn + 17$, P.D que $T(n) = 17n^3 + nlogn + 17 \in O(n^3)$
  Veamos que $17n^3 <= 17n^3$, $nlogn <= n^3$ y $17 <= 17n^3$ para toda $n >= 1$, ya que $nlogn <= n^3$ para toda $n$. Ahora, sumando las desigualdades tenemos que $17n^3 + nlog(n) + 17 <= 17n^3 + n^3 + 17n^3 = 35n^3$. Por lo tanto, $T(n) = 17n^3 + nlogn + 17 \in O(n^3)$ con $c = 35$ y $n_0 = 1$. $\square$

- Problema 4:

  Sea $T(n) = 53n + 3log(n)$, P.D que $T(n) = 53n + 3log(n) \in O(nlog(n))$
  Veamos que $53n <= 53nlog(n)$ y $3log(n) <= 3nlog(n)$ para toda $n >= 10$, ya que $log(n) <= nlog(n)$ para toda $n$ y que $n <= nlog(n)$ para toda $n >= 10$ si es $log_{10}$ o para toda $n >= 2$ si es $log_2$. Ahora, sumando las desigualdades tenemos que $53n + 3log(n) <= 53nlog(n) + 3nlog(n) = 56nlog(n)$. Por lo tanto, $T(n) = 53n + 3log(n) \in O(nlog(n))$ con $c = 56$ y $n_0 = 10$ si es $log_{10}$ o $n_0 = 2$ si es $log_2$. $\square$

- Problema 5:

  Sea $T(n) = 17nlog_2(n) + 3n$, P.D que $T(n) = 17nlog_2(n) + 3n \in O(n^2)$
  Veamos que $17nlog_2(n) <= 17n^2$ y $3n <= 3n^2$ para toda $n >= 1$, ya que $nlog_2(n) <= n^2$ para toda $n$. Ahora, sumando las desigualdades tenemos que $17nlog_2(n) + 3n <= 17n^2 + 3n^2 = 20n^2$. Por lo tanto, $T(n) = 17nlog_2(n) + 3n \in O(n^2)$ con $c = 20$ y $n_0 = 1$. $\square$