

## Laboratorio 8

Para este laboratorio cree un repositorio de github

- Incluya un `README.md` con instrucciones de ejecución.
- Para ejercicios sin código, cree una carpeta separada con respuestas en PDF.
- Grabe un video ( $\leq 10$  min) mostrando la ejecución de sus programas, súbalo a YouTube como no listado y enlázelo en el README.

### Problema 1: 25%

Analice el siguiente programa

```
1 void function(int n) {  
2     int i, j, k, counter = 0;  
3     for (i = n/2; i <= n; i++) {  
4         for (j = 1; j+n/2 <= n; j++) {  
5             for (k = 1; k <= n; k = k*2) {  
6                 counter++;  
7             }  
8         }  
9     }  
10 }
```

#### Parte a

Encuentre la complejidad de tiempo en notación Big-Oh. Muestre todo su procedimiento.

#### Parte b

Implemente el programa en el lenguaje de su elección y utilice profiling para medir el tiempo de ejecución con los siguientes valores de  $n$ : 1, 10, 100, 1000, 10000, 100000, 1000000.

Presente los resultados en:

- Una tabla con tamaño de input vs. tiempo
- Una gráfica de tamaño de input vs. tiempo

## Problema 2: 25%

Analice el siguiente programa:

```
1 void function(int n) {
2     if (n <= 1) return;
3     int i, j;
4     for (i = 1; i <= n; i++) {
5         for (j = 1; j <= n; j++) {
6             printf("Sequence\n");
7             break;
8         }
9     }
10 }
```

### Parte a

Encuentre la complejidad de tiempo en notación Big-Oh. Muestre todo su procedimiento.

### Parte b

Implemente y realice profiling con  $n \in \{1, 10, 100, 1000, 10000, 100000, 1000000\}$ . Presente resultados en tabla y gráfica.

## Problema 3: 50%

Analice el siguiente programa:

```
1 void function(int n) {
2     int i, j;
3     for (i = 1; i <= n/3; i++) {
4         for (j = 1; j <= n; j += 4) {
5             printf("Sequence\n");
6         }
7     }
8 }
```

### Parte a

Encuentre la complejidad de tiempo en notación Big-Oh. Muestre todo su procedimiento.

### Parte b

Implemente y realice profiling con  $n \in \{1, 10, 100, 1000, 10000, 100000, 1000000\}$ . Presente resultados en tabla y gráfica.

## Problema 4: 25%

Encuentre el **mejor caso**, **caso promedio** y **peor caso** del algoritmo de Búsqueda Lineal (Linear Search, Binary Search y Quick sort). Muestre todo su procedimiento.

## Problema 5

Determine si los siguientes enunciados son verdaderos o falsos. **Justifique sus respuestas** para recibir créditos completos.

## Teoría de la computación

Octubre , 2025

- a) Si  $f(n) = \Theta(g(n))$  y  $g(n) = \Theta(h(n))$ , entonces  $h(n) = \Theta(f(n))$ .
- b) Si  $f(n) = O(g(n))$  y  $g(n) = O(h(n))$ , entonces  $h(n) = \Omega(f(n))$ .
- c)  $f(n) = \Theta(n^2)$ , donde  $f(n)$  está definido como el tiempo de ejecución del siguiente programa Python:

```
1 def A(n):  
2     atupla = tuple(range(0, n))  
3     # Una tupla es una version inmutable de una lista  
4     # que puede ser hasheada  
5     S = set()  
6     for i in range(0, n):  
7         for j in range(i + 1, n):  
8             # Anade la tupla (i,...,j-1) al set S  
9             S.add(atupla[i:j])
```