Universidad del Valle de Guatemala
Facultad de Ingeniería
Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Laboratorio No. 8

Para los siguientes incisos, siga estas instrucciones:

- Cree un repositorio **privado** en GitHub.com para alojar el código correspondiente para este laboratorio.
- Añada a los siguientes usuarios con rol de *collaborator*:
  - AristondoAux
  - Daniel14gonc
  - gbrolo
- Coloque todas las instrucciones que considere pertinentes en un archivo denominado README.md en la raíz del repositorio.
- Para los incisos que no involucren la generación de código, cree una carpeta por separado en su repositorio y coloque las respuestas en un documento con formato PDF
- Grabe un video de no más de 10 minutos donde muestre la ejecución de sus programas para los ejercicios que soliciten programar. Súbalo a YouTube como video no listado y adjúntelo en el README de su repositorio.

Ejercicio No. 1 (25%) – Para el siguiente programa:

```
void function (int n) {
	int i, j, k, counter = 0;
	for (i = n/2; i <= n; i++) {
		for (j = 1; j+n/2 <= n; j++) {
			for (k = 1; k <= n; k = k*2) {
				counter++;
			}
		}
	}
}
```

a) Encuentre la complejidad de tiempo en notación Big-Oh. Deje todo su procedimiento.
b) Ahora, escriba un programa en el lenguaje de programación de su elección para implementar el programa anterior y utilice un método de profiling para medir el tiempo de ejecución de su programa con distintos tamaños de input n: 1, 10, 100, 1000, 10000, 100000, 1000000. Coloque los resultados en una tabla y grafique los resultados obtenidos en una gráfica de tamaño de input vs. tiempo.

Ejercicio No. 2 (25%) – Para el siguiente programa:

```
void function (int n) {
	if (n <= 1) return;
	int i, j;
	for (i = 1; i <= n; i++) {
		for (j = 1; j <= n; j++) {
			printf (“Sequence\n”);
			break;
		}
	}
}
```

a) Encuentre la complejidad de tiempo en notación Big-Oh. Deje todo su procedimiento.
b) Ahora, escriba un programa en el lenguaje de programación de su elección para implementar el programa anterior y utilice un método de profiling para medir el tiempo de ejecución de su programa con distintos tamaños de input n: 1, 10, 100, 1000, 10000, 100000, 1000000. Coloque los resultados en una tabla y grafique los resultados obtenidos en una gráfica de tamaño de input vs. tiempo.

Ejercicio No. 3 (25%) – Para el siguiente programa:

```
void function (int n) {
	int i, j;
	for (i=1; i<=n/3; i++) {
		for (j=1; j<=n; j+=4) {
			printf(“Sequence\n”);
		}
	}
}
```

a) Encuentre la complejidad de tiempo en notación Big-Oh. Deje todo su procedimiento.
b) Ahora, escriba un programa en el lenguaje de programación de su elección para implementar el programa anterior y utilice un método de profiling para medir el tiempo de ejecución de su programa con distintos tamaños de input n: 1, 10, 100, 1000, 10000, 100000, 1000000. Coloque los resultados en una tabla y grafique los resultados obtenidos en una gráfica de tamaño de input vs. tiempo.

Ejercicio No. 4 (10%) – Encuentre el mejor caso, caso promedio y peor caso del algoritmo de Búsqueda Lineal (Linear Search). Deje todo su procedimiento.

Ejercicio No. 5 (15%) – Decida si los siguientes enunciados son verdaderos o falsos. Debe justificar sus respuestas para recibir los créditos completos.

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 por ser el tiempo de ejecución del programa de Python** `A(n)`:

```
def A(n):
    atupla = tuple(range(0, n))  # una tupla es una versión immutable de una
                                 # lista, que puede ser hasheada
S = set()
for i in range(0, n):
    for j in range (i + 1, n):
        S.add(atupla[i:j])       # añada la tupla (i,...,j-1) al set S
```