Lab 8

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.

$2n \times \frac{n}{2} \times \log_2(n)$

$j + \frac{n}{2} \leq n$

$\frac{n}{2} \approx O(n)$

$\therefore O(n) \times O(n) \times O(\log_2 n) = O(n \log_2 n)$

$f(n) = \frac{n}{2} \cdot \frac{n}{2} \cdot \log_2 n = \frac{n^2}{4} \log_2 n \quad \wedge \quad g(n) = n^2 \log_2 n$

$\Rightarrow f(n) \leq c \cdot g(n) = \frac{n^2}{4} \log_2 n \leq c \cdot n^2 \log_2 n \Rightarrow \frac{1}{4} < c$

$\therefore c = 1$

$\frac{n^2}{4} \log_2 n \leq 1 \cdot n^2 \log_2 n$

$n_0 = 2$ $\qquad f(n) = O(n^2 \log_2 n)$

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.

$1 + n \times 1 = n = O(n)$

$f(n) = n + 1 \quad \wedge \quad g(n) = n \Rightarrow f(n) \leq c \cdot g(n) \Rightarrow 1 + n \leq c \cdot n \; ; \; 1 < c \Rightarrow c = 2$

$c = 2 \qquad n + 1 \leq 2n \Rightarrow n_0 = 1$

$f(n) = O(n)$

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.

$\frac{n}{3} \times \frac{n}{4} = \frac{n^2}{12} = O(n^2)$

$O(n) + O(n) = O(n^2)$

$f(n) = \frac{n^2}{12} \quad \wedge \quad g(n) = n^2$

$\Rightarrow f(n) \leq c \cdot g(n) = \frac{n^2}{12} \leq c \cdot n^2$

$\Rightarrow \frac{1}{12} < c \Rightarrow c = 1$

$\frac{n^2}{12} \leq 1 \cdot n^2 \Rightarrow n_0 = 1$

$f(n) = O(n^2)$

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

```
look_for = "something"
for i in array
    if (i == look_for)
        print ("encontrado")
```

Sea array un array con .len() = n entonces existen:

- mejor caso array[0] = "something"
- caso medio array[int(n/2)] = "something"
- peor caso array[n-1] = "something"

∴ Ω(1)
O(n) ← lineal
Θ(n)

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
```

a) Verdadero por simetría y transitividad

b) Verdadero si $f(n) \in c_1 g(n)$ y $g(n) \in c_2 h(n)$

$f(n) \in c_1 g(n) \in c_1 c_2 h(n) \Rightarrow f(n) \in c_3 h(n) \Rightarrow f(n) = O(h(n))$

por equivalencia → Ω

c) falso

```
def A(n):
    atupla = tuple(range(0, n))

S = set()
for i in range(0, n):          ← O(n)
    for j in range (i + 1, n):  ← O(n)
        S.add(atupla[i:j])      ← O(j-i) ≈ O(n)
```

$O(n) \times O(n) \times O(n) = O(n^3)$