Ejercicios Propuestos sobre análisis de algoritmos

Semana 2

1. **Determinar la función T(n) que calcula el número de veces que se ejecuta la fracción de código en función al tamaño de la entrada "n". Indique además, la complejidad de los algoritmos en notación Big-O.**

a)

```
for(int i = 0; i <= 200; i++){
    n++;
}
System.out.println(n);
```

b)

```
int suma = 0;
for(int i = 0; i <= n; i++){
    suma = suma + i;
}
```

c)

```
for(int i = 0; i <= n; i++){
    for(int j = 0; j <= n; j++){
        resultado =  resultado + i * j;
    }
}
```

d)

```
int suma = 0;
for(int i = 0; i < 100; i++){
    for(int j = 1; j <= n; j++){
        for(int k = 1; k < n * n; k++){
            suma = suma + i + j + k;
        }
    }
}
```

2. **Determinar la función T(n) que calcula el número de veces que se ejecuta la sentencia subrayada para cada uno de los siguientes algoritmos:**

a)

```
int suma = 0;
for(int i = 1; i <= n; i++){
   for(int j = 1; j < n * n; j++){
       suma += 1;
   }

}
```

b)

```
for(int j = 1; j <= n; j++){
   for(int k = j; k <= n; k++){
       for(int p = 0; p < 8; p++){
           //INSTRUCCION
       }

   }

}
```

c)

```
int suma = 0;
for(int i = 1; i < n; i++){
   for(int k = i; k < n; k++){
       suma +=1;
   }

}
```

d)

```
int r = 0;
for(int i = 1; i <= n - 1; i++){
   for(int j = i + 1; j <= n; j++){
       for(int k = 1; k <= j; k++){
           r += 1;
       }

   }

}
```

e)

```
int suma = 0;
for(int i = 1; i <= n; i++){
   int valor = i;
   for(int j = valor; j <= n; j++){
       int k = 1;
       while(k <= 5){
           suma +=1;
           k += 1;
       }
   }

}
```

f)

```
int suma = 0;
for(int j = 1; j <= n; j++){
   int k = n;
   while(k > 0){
       suma += 1;
       k /= 2;
   }
}
```

g) Sea b un entero mayor que 1:

```
int j = 1;
while(j < n){
   j = b * j;
}
```

3) Realice una análisis teorico y empirico de los siguientes algoritmos

a) Escribir una función que permita recibir como dato de entrada un arreglo de enteros diferente de cero y que no tenga duplicados, para luego devolver como resultado la cantidad de números que tienen pareja. Se dice que un número x tiene pareja en el arreglo si existe otro número opuesto a él es decir existe -x.

Ejemplo:
Input:
A = {**1**,3,5,**2**,**-2**,8,12,**-1**,19}

Output
numParejas = 2

Después de crear el algoritmo debe realizar el análisis de complejidad para determinar el T(n) o la notación asintótica Big O.

b) Escribir una función que permita recibir como entrada tres arreglos de números enteros para luego devolver un arreglo con los elementos comunes entre los tres arreglos. Después de crear el algoritmo debe realizar el análisis de complejidad para determinar el T(n) o la notación asintótica Big O.

4) De las siguientes afirmaciones, indicar cuales son ciertas y cuáles no:

a) Solo marcar con una x o un aspa.

(i)	$n^2 \in O(n^3)$	(ix)	$n^2 \in \Omega(n^3)$
(ii)	$n^3 \in O(n^2)$	(x)	$n^3 \in \Omega(n^2)$
(iii)	$2^{n+1} \in O(2^n)$	(xi)	$2^{n+1} \in \Omega(2^n)$
(iv)	$(n+1)! \in O(n!)$	(xii)	$(n+1)! \in \Omega(n!)$
(v)	$f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$	$(xiii)$	$f(n) \in \Omega(n) \Rightarrow 2^{f(n)} \in \Omega(2^n)$
(vi)	$3^n \in O(2^n)$	(xiv)	$3^n \in \Omega(2^n)$
(vii)	$\log n \in O(n^{1/2})$	(xv)	$\log n \in \Omega(n^{1/2})$
$(viii)$	$n^{1/2} \in O(\log n)$	(xvi)	$n^{1/2} \in \Omega(\log n)$

b) Sea a una constante real, 0<a<1. Usar las relaciones $\subset$ y = para ordenar los órdenes de complejidad de las siguientes funciones:

$$n\log n,\ n^2\log n,\ n^8,\ n^{1+a},\ (1+a)^n,\ (n^2+8n+\log^3 n)^4,\ 2^n$$