Complejidad del primer ejercicio del Taller 4

1. Suma de los elementos de un arreglo

1.1 Copiar el código en Word

```
private static int suma(int[] a, int i){
    if (i == a.length)
      return 0;
    else
      return a[i] + suma(a,i+1);
}
```

1.2 Identificar quién es el tamaño del problema (llamado también "n")

El tamaño del problema son los elementos que me falta por sumar en el arreglo.

1.3 Etiquetar cuánto se demora cada línea

```
private static int suma(int[] a, int i){
    if (i == a.length)  // constante
      return 0;  // constante
    else
      return a[i] + suma(a,i+1); //constante + T(n-1)
}
```

1.4 Escribir la ecuación de recurrencia

$$T(n) = \begin{cases} c_1 & \text{if } n = 1 \\ c_2 + T(n-1) & \text{if } n > 1 \end{cases}$$

1.5 Resolver la ecuación con Wolfram Alpha

T(n) = c_2 + T(n-1)

T(n) = c_2*n + c_1

1.6 Aplicar la notación O a la solución de la ecuación

T(n) es O(c_2*n + c_1), por definición de O

T(n) es O(c_2*n), por Regla de la Suma

T(n) es O(n), por Regla del Producto

1.7 Explicar en palabras

La complejidad asintótica (es decir, para valores grandes de n) para el peor de los casos (es decir, en el que el algoritmo hace un más operaciones) para el algoritmo de sumar los elementos de un arreglo recursivamente es O(n).

2. Volumen máximo

2.1 Copiar el código en Word

```
public static boolean groupSum(int start, int[] nums, int target) {
      if(start==nums.length && target==0) {
         return true;
      }else if (start<nums.length){
         return groupSum(start+1,nums,target-nums[start]) || groupSum(start+1,nums,target);
      }else{
      return false;
      }
   }
```

2.2 Identificar quién es el tamaño del problema (llamado también "n")

El tamaño del problema son los elementos que me falta por sumar para lograr el volumen máximo.

2.3 Etiquetar cuánto se demora cada línea

```
public static boolean groupSum(int start, int[] nums, int target) {
      if(start==nums.length && target==0) {   //constante
         return true;                          //constante
      }else if (start<nums.length){            //constante
         return groupSum(start+1,nums,target-nums[start]) ||
groupSum(start+1,nums,target);                  //constant + T(n-1)+T(n-1)  porque toma dos casos
      }else{
      return false;
      }
   }
```

2.4 Escribir la ecuación de recurrencia

$$T(n) \begin{cases} c_1 \text{ if } x = 1 \\ c_2 + T(n-1)^2 \text{ if } x > 1 \end{cases}$$

2.5 Aplicar la notación O a la solución de la ecuación

T(n) es O(n) ^2

1.6 Explicar en palabras

La complejidad asintótica (es decir, para valores grandes de n) para el peor de los casos (es decir, en el que el algoritmo hace un más operaciones) para el algoritmo de sumar los volumenes de un arreglo recursivamente es O(n) ^2. Debido a que se toman dos opciones al hacer la recursión para calcular todas las posibles combinaciones que den el volumen maximo