COMPLEJIDAD EJERCICIOS

EJERCICIO 1

1.1 Copiar el código en Word

```
public 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 n que faltan por sumarse.

1.3 Etiquetar cuánto se demora cada línea

```
public static int suma (int[]a, int i){
 if (i == a.length){
      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), debido a O

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

T(n) es O(n), debido a la 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. Suma para ver si es posible alcanzar un valor objetivo

2.1 Copiar el código en Word

```
public static boolean sumaVolumen(int inicio, double [] nums, double valor) {
      if(inicio>=nums.length){
         return valor==0;
      }
      if (sumaVolumen(inicio+1,nums,valor-nums[inicio])||sumaVolumen(inicio+1,nums,valor)){
         return true;
      }else{
         return false;
      }
   }
```

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

El tamaño del problema es el numero de elementos del arreglo que se van a verificar

2.3 Etiquetar cuánto se demora cada línea

```
public static boolean sumaVolumen(int inicio, double [] nums, double valor) {
      if(inicio>=nums.length){
         return valor==0;//constante
      }
      if (sumaVolumen(inicio+1,nums,valor-
nums[inicio])||sumaVolumen(inicio+1,nums,valor)){//constante+T(n-1)||constante+T(n-1)
         return true;
      }else{
         return false;
      }
   }
```

2.4 Escribir la ecuación de recurrencia

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

2.5 Resolver la ecuación con Wolfram Alpha

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

T(n) =$c_1 * 2^{n-1}$

2.6 Aplicar la notación O a la solución de la ecuación

T(n)=$O(c * 2^{n-1})$, debido a O

T(n)=$O(2^{n-1})$ debido a la regla del producto

T(n)=$O(2^n)$, debido a la regla de la suma

2.7 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 ver si es posible obtener un valor objetivo entre los elementos, al sumarlos, de un arreglo, recursivamente, es O(2^n).