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).

Complejidad del primer ejercicio del Taller 4

1. Suma de los elementos de un arreglo

1.1 Copiar el código en Word

```
public static boolean groupSum(int start, int[] nums, int target) {
      if(start>=nums.length){
         return target==0;
      } else if(nums[start]==6){
         return groupSum(start+1, nums, target-nums[start]);
      } else {
         return groupSum(start+1, nums, target);
      }
   }
```

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

El tamaño del problema son los elementos que me faltan para lograr el target.

1.3 Etiquetar cuánto se demora cada línea

```
public static boolean groupSum(int start, int[] nums, int target) {
      if(start>=nums.length){   //constante
         return target==0;   //constante
      } else if(nums[start]==6){   //constante
         return groupSum(start+1, nums, target-nums[start]); // T(n-1)
      } else {
         return groupSum(start+1, nums, target);  // T(n-1)
      }
   }
```

1.4 Escribir la ecuación de recurrencia

C1	if n=nums.length
T(n-1)	if n=6 or n!= nums.length

1.5 Resolver la ecuación con Wolfram Alpha

T(n)=2*T(n-1)+c

1.6 Aplicar la notación O a la solución de la ecuación

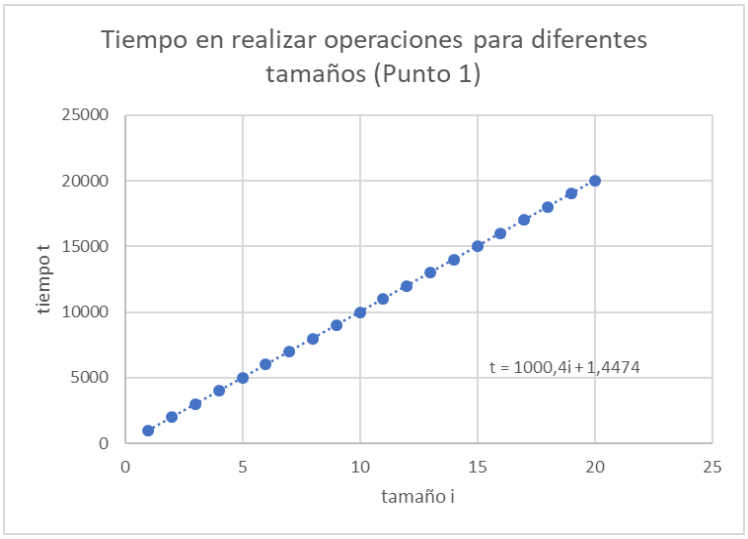
T(n) es O(2^n)

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(2^n).

Graficas

1. De la gráfica podemos concluir que el tiempo que tardan en realizarse las operaciones para diferentes tamaños, siempre incrementará a una razón de 1000,4 milisegundos aproximadamente. Por lo tanto, si aumenta el tamaño, el tiempo incrementará de igual forma.



2. De la gráfica podemos concluir que el tiempo que tardan en realizarse las operaciones para diferentes tamaños, siempre incrementará a una razón de 1000,4 milisegundos aproximadamente. Por lo tanto, si aumenta el tamaño, el tiempo incrementará de igual forma. De esta manerá se podrá predecir el tiempo aproximado que tomará para realizar el procedimiento con cualquier tamaño.