Taller en Sala No. 7
Notación O con recursión

Vida real: En la vida real, la serie de Fibonacci se utiliza para diseñar la distribución de objetos en los videojuegos de tal forma que las escenas cumplan con una cierta estética determinada por la proporción aurea de Fibonacci
https://i.ytimg.com/vi/QQoj2_4SvX4/maxresdefault.jpg

Para resolver en parejas.

Considere los siguientes algoritmos.

ArrayMax, calcula el máximo valor de un arreglo de enteros. A es el arreglo y n es la última posición del arreglo en el primer llamado. ArrayMax se llama recursivamente hasta que n sea igual a 0.

Pista: Como un ejemplo ArrayMax([1,2,3,8,10,4], 5) retorna 10.
Pista 2: El tamaño del problema aquí es el número de elementos del arreglo

```
SubProceso ArrayMax( A, n )
      Definir i, max, temp Como Entero;
             max <- A[n]; //
Si n = 0, max <- A[0]
             Si n != 0 Entonces
                   temp <- ArrayMax(A, n-1);
                   Si temp > max Entonces
                       max <- temp;
      Retornar max;
```

El algoritmo **groupSum**, dado un arreglo de enteros, decide si es posible escoger un subconjunto de esos enteros, de tal forma que la suma de los elementos de ese subconjunto sea igual al parámetro ***target***. El parámetro ***start*** funciona como un contador y representa un índice en el arreglo de números ***nums***.

Pista: El tamaño del problema aquí es el número de elementos del arreglo

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

El algoritmo fibonacci calcula el valor enésimo de la serie de Fibonacci recursivamente.

Pista: El tamaño del problema es el término enésimo (int n).

```
public static long fibonacci(int n) {
    if (n <= 1) return n;
    else return fibonacci(n-1) + fibonacci(n-2);
}
```

1. Implementar el algoritmo de ArrayMax en Java y copiar los otros dos.

```
public static int ArrayMax( int []A, int n ){
   int max , temp =0;
   max =A[n] ;
     if(n==0){
          max=A[0];
      }else {
         temp = ArrayMax(A , n-1);
      }
      if(temp > max){
         max=temp;
      }

  return max;
}
```

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


}
```

```
public static long fibonacci(int n) {
   if (n <= 1) return n;
   else return fibonacci(n-1) + fibonacci(n-2);
}
```

2. Identificar qué representa el tamaño del problema para cada algoritmo

ArrayMax: el número de elementos del arreglo

GroupSum: el número de elementos del arreglo

Fibonacci: el término enésimo

3. Identificar valores apropiados para tamaños del problema

ArrayMax: 1 - 3.000.000

GroupSum: 1 - 45

Fibonacci: 1 - 45

4. Tomar los tiempos para los anteriores algoritmos con 10 tamaños del problema diferentes.

ArrayMax:

Números De Datos	Tiempo (nanosegundos)
10	2113
100	5132
1000	72142
10000	466056
100000	463641
1000000	563553
10000000	571401
100000000	581362
200000000	624828
300000000	619697
NOTA: la cantidad de los enteros dentro del arreglo son hasta 500000000	

GroupSum:

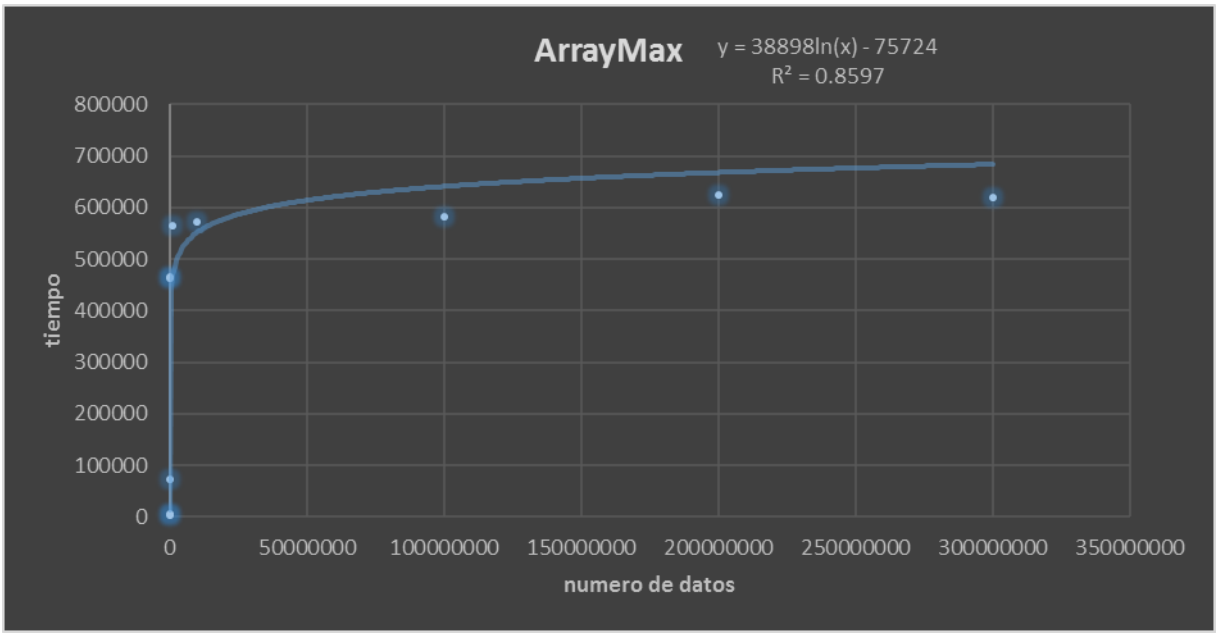
Números De Datos	Tiempo (nanosegundos)
1	157565
5	201937
10	230311
15	526425
20	1721145
25	10399920
30	970889641
35	24762935909
37	1.41793E+11
39	1.98768E+11
NOTA: la cantidad de los enteros dentro del arreglo son hasta 500000000	

Fibonacci:

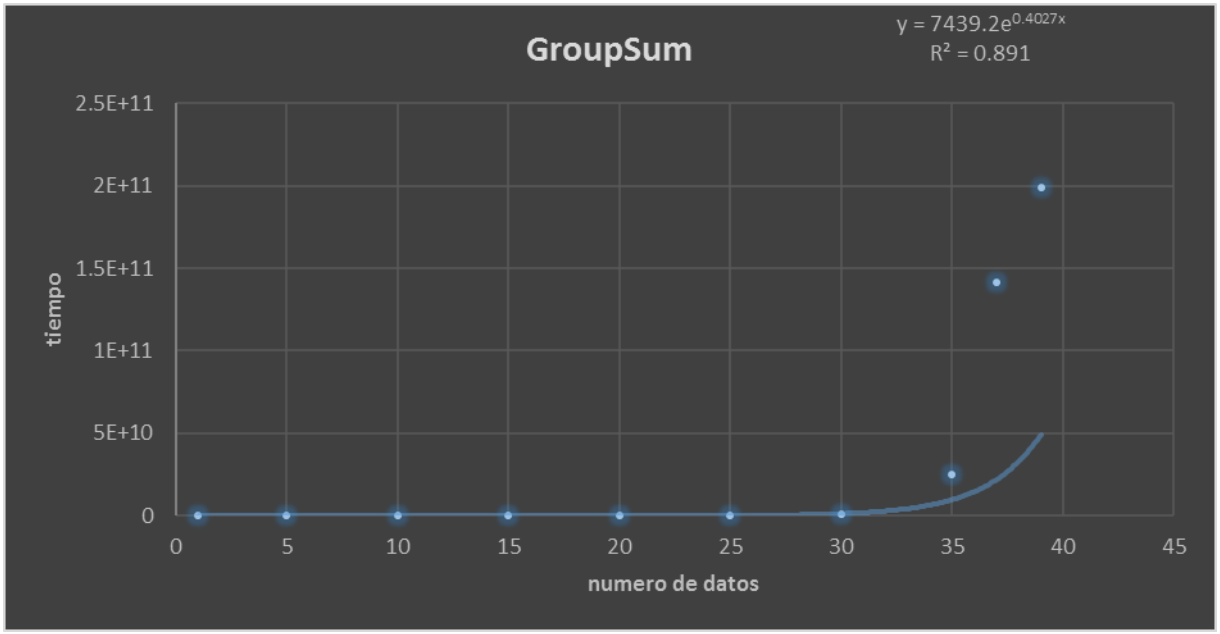
Números De Datos	Tiempo (nanosegundos)
1	1207
5	1811
10	7546
15	78783
20	315734
25	1061906
30	4825365
35	45767920
40	506131468
45	5528553447
NOTA: la cantidad de los enteros dentro del arreglo son hasta 500000000	

5. Hacer una gráfica en Excel para cada algoritmo y pasarla a Word luego

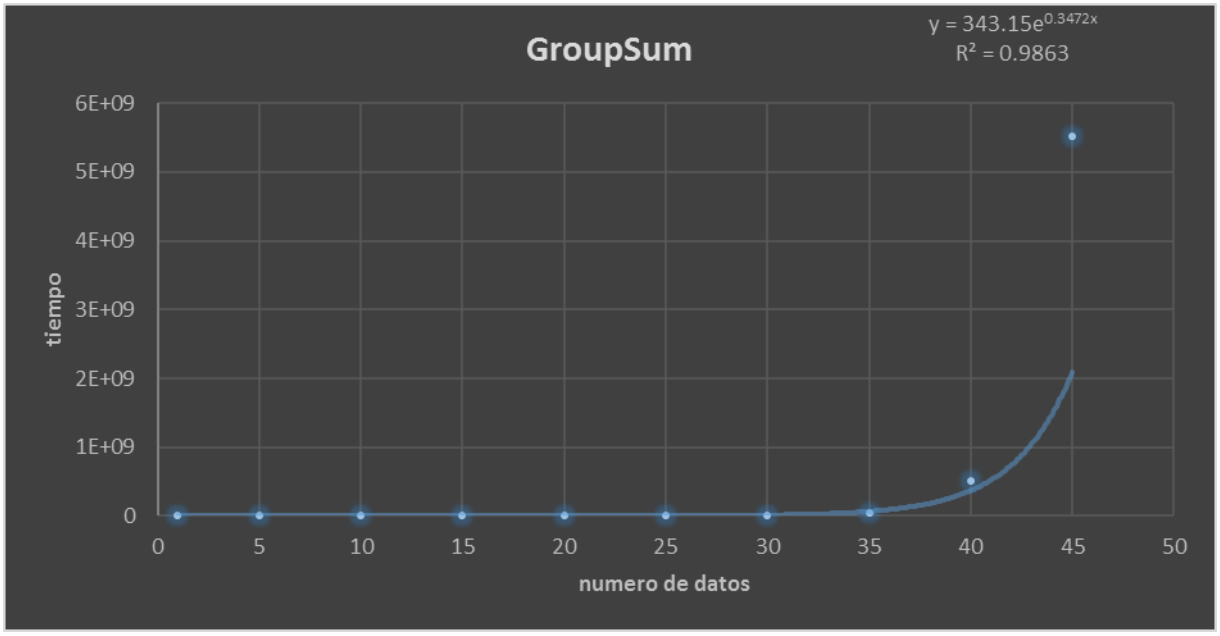
ArrayMax:



GroupSum:



Fibonacci:



En todas las gráficas las líneas de tendencias en log no coincidían con ninguno de los puntos graficados.

Velocidad del procesador: 3,4 GHz o 3.400.000.000 Hz