

## Laboratorio Nro. 1: Recursión

**Alejandro Arroyave Bedoya**

Universidad Eafit

Medellín, Colombia

[aarroyaveb@eafit.edu.co](mailto:aarroyaveb@eafit.edu.co)

**2.3)** El ejercicio groupSum5 calcula si es posible sacar un grupo de enteros de un arreglo tal que sus suma sea el objetivo introducido por el usuario, pero con la restricción de que todo número múltiplo de 5 tiene que agregarse al grupo de enteros, y si este está seguido inmediatamente de un 1, este no podrá estar incluido en el grupo.

### **2.4.1) Complejidad Recursión 1**

-factorial:

```
public int factorial(int n) {  
    if (n==0 || n==1)           //c  
        return 1;              //c  
    else  
        return n*factorial(n-1); //n*T(n-1)  
}
```

$T(n) = c' + n \cdot T(n-1)$

$O(c' + nT(n-1))$

-bunnyEars

```
public int bunnyEars(int bunnies) {  
    if (bunnies==0)           //c  
        return 0;             //c  
    else  
        return 2+bunnyEars(bunnies-1); //c + T(n-1)  
}
```

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

$O(cn + c') = O(n)$

**DOCENTE MAURICIO TORO BERMÚDEZ**

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

- bunnyEars2

```
public int bunnyEars2(int bunnies) {  
    if (bunnies==0) //c  
        return 0; //c  
    else  
        if (bunnies%2==1) //c  
            return 2+bunnyEars2(bunnies-1); //c+T(n-1)  
        else  
            return 3+bunnyEars2(bunnies-1); //c'+T(n-1)  
}
```

$$T(n)=c(2^n-1)+c_1.2^{n-1}$$

$$O(c(2^n-1)+c_1.2^{n-1})=O(2^n)$$

- triangle

```
public int triangle(int rows) {  
    if (rows==0) //c  
        return 0; //c  
    else  
        return triangle(rows-1)+rows; //T(n-1)+n  
}
```

$$T(n) = 1/2n(2c+n+1)+c_1$$

$$O(1/2n(2c+n+1)+c_1) = O(n)$$

- sumDigits

```
public int sumDigits(int n) {  
    int b=n%10; //c  
    if (n==0) //c  
        return 0; //c  
    else  
        return sumDigits(n/10)+b; //T(n/10)+c  
}
```

$$T(n)=(c.\log(n)/\log(10)+c)$$

$$O((c.\log(n)/\log(10)+c))=O(\log(n))$$

```
- groupSum6
public boolean groupSum6(int start, int[] nums, int target) {
    if (start >= nums.length)
        return target == 0;
    else {
        if (nums[start] == 6)
            return groupSum6(start+1, nums, target-nums[start]);
        else
            return groupSum6(start+1, nums, target-nums[start]) ||
                groupSum6(start+1, nums, target);
    }
}

- groupNoAdj
public boolean groupNoAdj(int start, int[] nums, int target) {
    if (start >= nums.length)
        return target == 0;
    else
        return groupNoAdj (start+2, nums, target-nums[start]) ||
            groupNoAdj (start+1, nums, target);
}

- groupSum5
public boolean groupSum5(int start, int[] nums, int target) {
    if (start >= nums.length)
        return target == 0;
    else {
        if (nums[start] % 5 == 0)
            return groupSum5(start+1, nums, target-nums[start]);
        if (start != 0 && nums[start] == 1 && nums[start-1] % 5 == 0)
            return groupSum5(start+1, nums, target);
    }
    return groupSum5(start+1, nums, target-nums[start]) ||
        groupSum5(start+1, nums, target);
}

- groupSumClump
public boolean groupSumClump(int start, int[] nums, int target) {
    if (start >= nums.length)
        return target == 0;
    else {
        int n = 1;
        while (start <= nums.length-2) {
            if (nums[start] != nums[start+1])
```

```

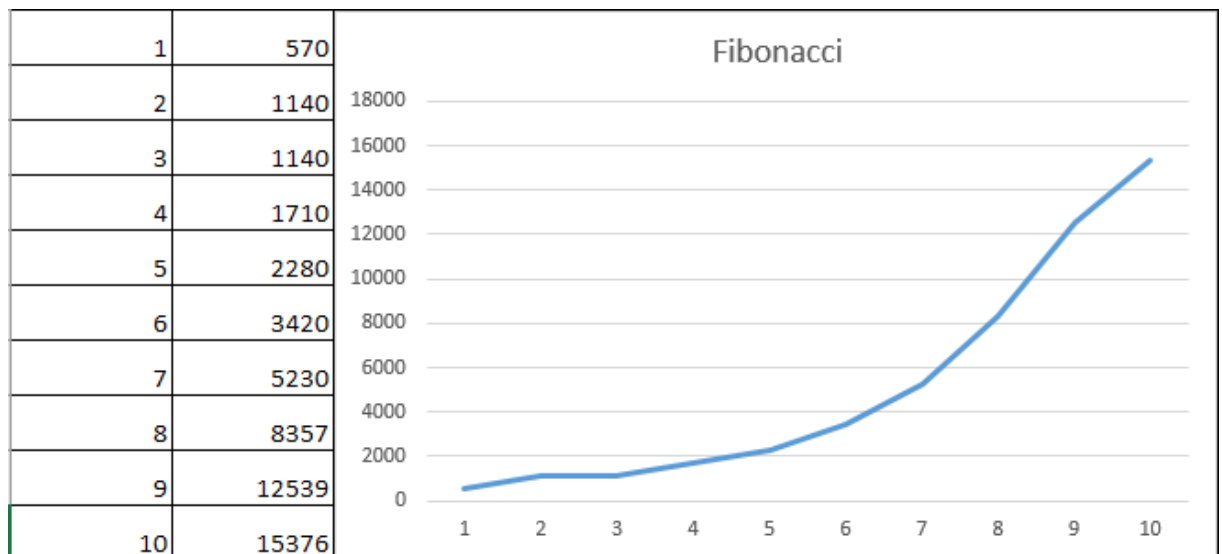
        break;
        start++;
        n++;
    }
    n=n*nums[start];
    return groupSumClump(start+1, nums, target-n) ||
    groupSumClump(start+1, nums, target);
}

}
- splitArray
public boolean splitArray(int[] nums) {
    return splitArrayAux(nums, 0, 0, 0);
}
- Método auxiliar para el problema splitArray
public boolean splitArrayAux(int []nums,int start,int g1,int g2){
    if (start >= nums.length)
        return g1 == g2;
    else
        return splitArrayAux(nums, start+1, g1+nums[start], g2) ||
        splitArrayAux(nums, start+1, g1, g2+nums[start]);
}
}

```

### 3) Simulacro de preguntas de sustentación de Proyectos

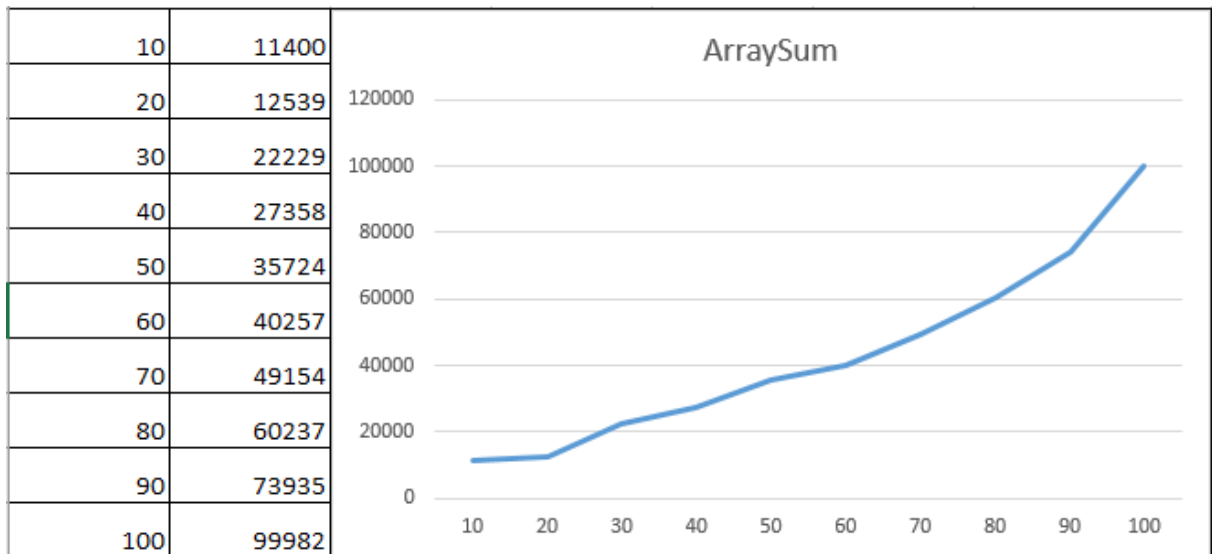
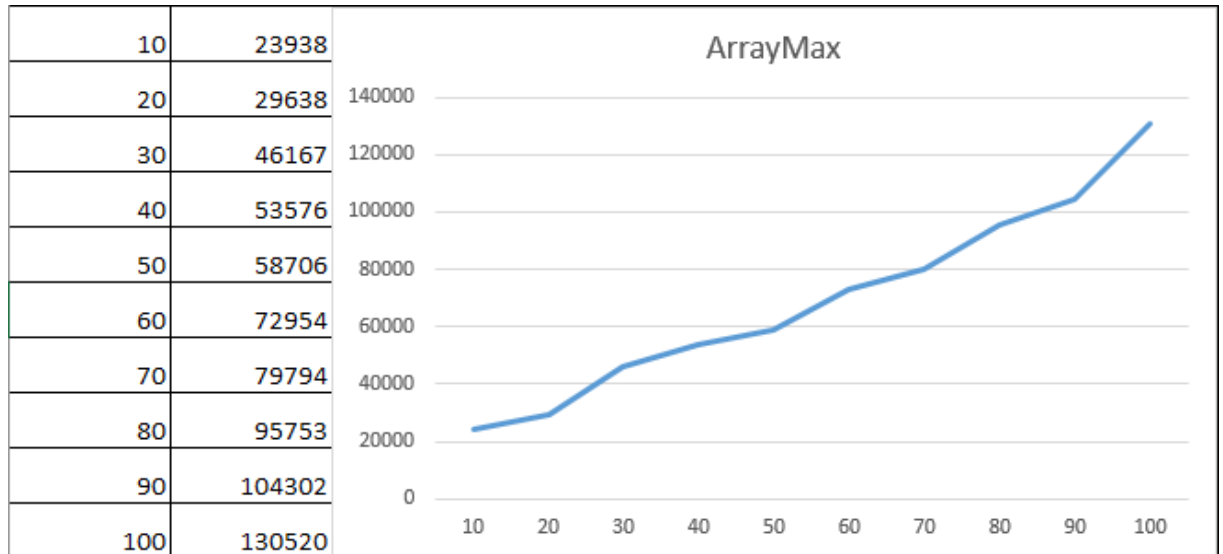
2.



**DOCENTE MAURICIO TORO BERMÚDEZ**

**Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627**

**Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)**




3. Que puede variar mucho los tiempos dependiendo de las especificaciones de la máquina, pero en general son acertados.
4. El stack overflow es cuando la pila de la máquina se llena, esto se debe a que el programa ejecutado requiere mas pila de lo que esta configurado predeterminadamente, son comunes cuando el programador crea por error un bucle infinito o por programas profundamente anidados.
5. 50, porque primero se prolonga mucho el tiempo de ejecución, y también por que salta la excepción stack overflow, que significa que se desbordó la pila
- 6.

**DOCENTE MAURICIO TORO BERMÚDEZ**

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

	<p>UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</p>	<p>Código: ST245</p> <p>Estructura de Datos 1</p>
---	---	---

7. La complejidad de los problemas de recursión 2 es mucho más alta que la de recursión 1, ya que hay mas llamadas recursivas en cada método.

#### **4) Simulacro de Parcial**

1. *Start+1, nums, target*