Laboratorio Nro. 1: Recursión

Agustin Restrepo Cadavid
Universidad Eafit
Medellín, Colombia
arestrepoc@eafit.edu.co

Nombre completo de integrante 2
Universidad Eafit
Medellín, Colombia
sgonzalez1@eafit.edu.co

2.3) GrupoSum

GrupoSum es un problema de recursión en el cual es necesario analizar todas las opciones. El código debe retornar verdadero si existe combinación de números del arreglo, los cuales sumen un numero especifico. Para esto se debe tomar todas las posibilidades. Usando recursión, el programa llama a el mismo método dos veces, en una le resta a el numero especifico un numero de el arreglo, y en el otro no, también afirma que números del arreglo ya fueron usados.

2.4 y 2.5) Online Exercises (CODINGBAT)

2.1 Recursion I

i.

```
public static int factorial(int n) {
    if(n==1) return 1;                          //C1
    return factorial(n-1)*n;                    //C2+T(n-1)
}
```

T(n) = C1, si n = 1
T(n) = C2+T(n-1), de lo contrario
T(n) = C*n + C'
T(n) = O(n)
La variable n significa el valor del parametro n en factorial(n).

ii.

```
public static int triangle(int rows) {
    if(rows == 0) return 0;                     //C1
    return rows+triangle(rows-1);               //C2+T(n-1)
}
```

T(n) = C1, si n = 0
T(n) = C2+T(n-1), de lo contrario
T(n) = C*n + C'
T(n) = O(n)
La variable n significa el valor del parametro n en triangle(rows).

iii.

```
public static int sumDigits(int n) {
      if(n/10<1) return n;                                    //C1
      return n%10+sumDigits(n/10);                            //C2+T(n/10)
   }
```

T(n) = C1, si n < 10
T(n) = C2+T(n/10), de lo contrario
T(n) = C*log(n) + C'
T(n) = O(log(n))
La variable n significa el valor del parametro n en sumDigits(n).

iv.

```
public static boolean array220(int[] nums, int index) {
      if(index>=nums.length-1) return false;                  //C1
      return nums[index]*10==nums[index+1]||array220(nums,index+1);   //C2+T(n-1)
   }
```

T(n) = C1, si array.length –n = 1
T(n) = C2+T(n-1), de lo contrario
T(n) = C*n + C'
T(n) = O(n)
La variable n significa el valor del parámetro index en array220(a,index), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

v.

```
public static boolean array6(int[] nums, int index) {
   if(index>nums.length-1) return false;                      //C1
   return nums[index]==6||array6(nums,index+1);               //C2+T(n-1)
}
```

T(n) = C1, si nums.length – n = 1
T(n) = C2+T(n-1), de lo contrario
T(n) = C*n + C'
T(n) = O(n)
La variable n significa el valor del parámetro index en array6(a,index), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

2.2 Recursion 2

i.

```
public static boolean groupSum6(int start, int[] nums, int target) {
      if(start>=nums.length) return target == 0;                          //C1
      if(nums[start]==6) return groupSum6(start+1,nums,target-6);         //C2+T(n-1)
      return groupSum6(start+1,nums,target-nums[start])||                 //C3+T(n-1)+T(n-1)
         groupSum6(start+1,nums,target);
   }
```

T(n) = C1, si n -nums.length = 0
T(n) = C3+T(n-1)+T(n-1), de lo contrario
T(n) = C*2^n+C'
T(n) = O(2^n)
La variable n significa el valor del parámetro index en groupSum6(start, nums, target), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

ii.

```
public static boolean groupNoAdj(int start, int[] nums, int target) {
      if(start>=nums.length) return target == 0;                          //C1
      return groupNoAdj(start+2,nums,target-nums[start])||                //C2+T(n-2)+T(n-1)
         groupNoAdj(start+1,nums,target);
   }
```

T(n) = C1, si n -nums.length = 0
T(n) = C3+T(n-2)+T(n-1), de lo contrario
T(n) = C*2^n+C'
T(n) = O(2^n)
La variable n significa el valor del parámetro index en groupNoAdj(start, nums, target), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

iii.

```
public static boolean splitArray(int[] nums) {
      return splitArrayHelper(0,nums,0,0);                                //C1
   }
 public static boolean splitArrayHelper(int index, int[] nums, int group1,int group2){
      if(index>=nums.length) return group1==group2;                       //C2
      return splitArrayHelper(index+1,nums,group1+nums[index],group2)||   //C3+T(n-1)+T(n-1)
         splitArrayHelper(index+1,nums,group1,group2+nums[index]);
   }
```

T(n) = C1+C2, si n -nums.length = 0
T(n) = C3+T(n-1)+T(n-1), de lo contrario
T(n) = C*2^n+C'
T(n) = O(2^n)
La variable n significa el valor del parámetro index en splitArrayHelper (index, nums, group1, group2), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

iv.

```
public static boolean splitOdd10(int[] nums) {
      return splitOdd10Helper(0,nums,0,0);                                    //C1
   }
   public static boolean splitOdd10Helper(int index, int[] nums, int group1,int group2){
      if(index>=nums.length) return group1%10==0&&group2%2==1;                //C2
      return splitOdd10Helper(index+1,nums,group1+nums[index],group2)||       //C3+T(n-1)+T(n-1)
         splitOdd10Helper(index+1,nums,group1,group2+nums[index]);
   }
```

T(n) = C1, si n -nums.length = 0
T(n) = C3+T(n-1)+T(n-1), de lo contrario
T(n) = C*2^n+C'
T(n) = O(2^n)

La variable n significa el valor del parámetro index en splitOdd10Helper (index, nums, group1, group2), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

v.

```
public static boolean split53(int[] nums) {
      return split53Helper(0,nums,0,0);                                                    //C1
   }
   public static boolean split53Helper(int index, int[] nums, int group1,int group2){
      if(index>=nums.length) return group1==group2;                                        //C2
      if(nums[index]%5==0) return split53Helper(index+1,nums,group1+nums[index],group2);   //C3+T(n-1)
      if(nums[index]%3==0) return split53Helper(index+1,nums,group1,group2+nums[index]);   //C4+T(n-1)
      return split53Helper(index+1,nums,group1+nums[index],group2)||                       //C5+T(n-1)+T(n-1)
         split53Helper(index+1,nums,group1,group2+nums[index]);
   }
```

T(n) = C1+C2, si n-nums.length = 0
T(n) = C5+T(n-1)+T(n-1), de lo contrario
T(n) = C*2^n+C'
T(n) = O(2^n)
La variable n significa el valor del parámetro index en split53Helper (index, nums, group1, group2), este parámetro está limitado por el tamaño de nums, por esto es que n es dada por el tamaño de nums.

3) Simulacro de preguntas de sustentación de Proyectos

1. **Aprendimos que hay programas que aunque sean simples, se vuelven difíciles para los computadores ya que el número de acciones que deben hacer crece exponencialmente a medida que cambia los parámetros.**
2. **El valor más grande que pude calcular para Fibonacci fue el 92, ya que después del 92 el numero era más grande que Long, para calcular este número use un método diferente a el común, en el cual se entraba como parámetro 3 números, los dos números Fibonacci pasados, y el número de**

veces restantes que se implementaría el método. Así fue más fácil para el computador. En el método tradicional, solo puedo calcular Fibonacci para números alrededor de 50.

3. **Para calcular valores más grandes de Fibonacci se necesita tipos de datos más grandes, y un método como el que implementé.**
4. **La complejidad de los problemas de Codingbat, era alta pero no tan alta como varios problemas que hemos hecho en clase, entonces eran posibles de resolver con esfuerzo.**

4) Simulacro de Parcial

1. Start+1,nums,target
2. d
3. 1=n-a,a,b,c 2 = solucionar(n-b,a,b,c)+1 3 = solucionar(n-c,a,b,c)+1
4. E
5. Linea 2 = return n Linea 3 = n-1 Linea 4 = n-2 5.2 = b
6. Linea 10 = sumaAux(n,i+2) Linea 12 = sumaAux(n.i+1)
7. Linea 9 = S,i+1,t-S[i] Linea 10 = S,i+1,t

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co