Laboratorio Nro. 1: Recursión

Andrés Felipe Tamayo Arango
Universidad Eafit
Medellín, Colombia
aftamayoa@eafit.edu.co

Jamer José Rebolledo Quiroz
Universidad Eafit
Medellín, Colombia
jjrebolleq@eafit.edu.co

2) EJERCICIOS ONLINE (CODINGBAT)

2.1 Recursion 1

1.

```
public String parenBit(String str) {
  if(str.startsWith("(") && str.endsWith(")"))return str;
  if(str.startsWith("(") && !str.endsWith(")"))return parenBit(str.substring(0,str.length() - 1));
  if(!str.startsWith("(") && str.endsWith(")"))return parenBit(str.substring(1));
  return parenBit(str.substring(1,str.length() - 1));
}

// Complejidad
T(N) = T(N - 2) + C
```

2.

```
public boolean nestParen(String str) {
   if(str.length() == 0)return true; //
   if(str.length() < 2)return false;   //
   if(str.length() == 2 && str.equals("()"))return true;
   if(str.length() == 2 && !str.equals("()"))return false;
   if(str.charAt(0) == '(' && str.charAt(str.length() - 1) == ')')return nestParen(str.substring(1,
str.length() - 1));
   return false;
}

// Complejidad
T(N) = T(N - 2) + C
```

3.

```
public int strCount(String str, String sub) {
        if(str.length() < sub.length())return 0;
        if(str.startsWith(sub))return 1 + strCount(str.substring(sub.length()), sub);
          else return strCount(str.substring(1), sub);
}

// Complejidad
T(N) = T(N - 1) + C
```

4.

```
public boolean strCopies(String str, String sub, int n) {
   if(str.length() < sub.length() && n > 0)return false;
   if(n == 0)return true;
   if(str.startsWith(sub))return strCopies(str.substring(1), sub, n - 1);
   if(!str.startsWith(sub))return strCopies(str.substring(1), sub, n);
   return false;
}

//Complejidad
T(N) =   T(N - 1) + C
```

5.

```
public int strDist(String str, String sub) {
        if(str.length() < sub.length())return 0;
        if(str.length() == sub.length() && str.equals(sub))return sub.length();
        if(str.length() == sub.length() && !str.equals(sub))return 0;
        if(str.startsWith(sub) && str.endsWith(sub))return str.length();
        if(!str.startsWith(sub) && str.endsWith(sub))return strDist(str.substring(1), sub);
        if(str.startsWith(sub) && !str.endsWith(sub))
        return strDist(str.substring(0,str.length() - 1), sub);
        else return strDist(str.substring(1, str.length() - 1), sub);
}

// Complejidad
T(N) = T(N - 2) + C
```

2.2

1.

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

```
// Complejidad
T(N) = T(N - 1) + T(N - 2) + C
```

2.

```
public boolean groupSum5(int start, int[] nums, int target) {
        if(start == nums.length)return target == 0;
        if(nums[start] % 5 == 0){
        if(start == nums.length - 1)return groupSum5(start + 1, nums, target - nums[start]);
        if(nums[start + 1] == 1)return groupSum5(start + 2, nums, target - nums[start]);
        if(nums[start + 1] != 1)return groupSum5(start + 1, nums, target - nums[start]); }
        if(groupSum5(start + 1, nums, target - nums[start]))return true;
         if(groupSum5(start + 1, nums, target))return true;
         return false;
}
```

```
// Complejidad
T(N) = 2 * T(N - 1) + C
```

3.

```
public boolean splitArray(int[] nums) {
         return sumArray(nums, 0, 0);
}

boolean sumArray(int[] nums, int index, int target){
        if(index == nums.length)return target == 0;
        if(sumArray(nums, index + 1, target - nums[index]))return true;
        if(sumArray(nums, index + 1, target + nums[index]))return true;
        return false;
}
```

```
// Complejidad
T(N) = 2 * T(N - 1) + C
```

4.

```
public boolean splitOdd10(int[] nums) {
         return odd(nums, 0, 0, 0);
}

boolean odd(int[] nums, int index, int target1, int target2){
        if(index == nums.length)return target1 % 10 == 0 && target2 % 2 == 1;
        if(odd(nums, index + 1, target1 + nums[index], target2))return true;
         if(odd(nums, index + 1, target1, target2 + nums[index]))return true;
         return false;
}
```

```
// Complejidad
```

T(N) = 2 * T(N - 1) + C

5.

```
public boolean split53(int[] nums) {
        return sp(nums, 0, 0);
}

public boolean sp(int[] nums, int index, int target){
        if(index == nums.length)return target == 0;
        if(nums[index] % 5 == 0)return sp(nums, index + 1, target + nums[index]);
        if(nums[index] % 3 == 0)return sp(nums, index + 1, target - nums[index]);
        return sp(nums, index + 1, target + nums[index]) || sp(nums, index + 1, target - nums[index]);
}
```

// Complejidad
T(N) = 2 * T(N - 1) + C

2.3 Explicación ejercicio GroupSum5

R.

2.5 Explicación de variables

En recursión 1, se está trabajando con Strings. Así, la variable N representa la longitud de la cadena sobre la que se está trabajando. Por otro lado, "C" representa una constante.

En recursión 2, la variable "N" representa el arreglo que consideraremos y "C" una constante.

3) Simulacro de preguntas de sustentación de Proyectos

1. **¿Qué aprendiste sobre Stack Overflow?**

   R. Aprendimos que es un error muy común en programación que sucede cuando en un llamado recursivo se excede la pila para los llamados en los métodos.

2. **¿Cuál es el valor más grande que pudo calcular para Fibonacci? ¿Por qué? ¿Por qué no se puede ejecutar Fibonacci para 1 millón?**

   R. ±50. Lo que sucede es que para calcular por ejemplo Fibonacci(8), se necesita calcular 1 vez Fibonacci(7), 2 veces Fibonacci(6), 3 veces Fibonacci(5), 5 veces Fibonacci(4), 8 veces Fibonacci(3), 13 veces Fibonacci(2) y así sucesivamente. Se están repitiendo muchas operaciones. Así para Fibonacci de 1 millón, se necesitan hacer una cantidad astronómica de operaciones antes de calcular la de 1 millón. ¡No hay suficiente tiempo en el Universo!

3. **¿Cómo se puede hacer para calcular el Fibonacci de valores grandes?**

   R. Con ciclos o almacenando los fibonacci ya calculados en un array.

4. **¿Qué concluyen sobre la complejidad de los problemas de CodingBat Recursion 1 con respecto a los de Recursion 2?**

   R. Pudimos observar que en la mayoría de ocasiones es el doble de complejidad para Recursión 2 que para recursión 1

4) Simulacro de Parcial

1. *Start + 1, nums, target*
2. *b*
3. *3.1 n - a, a, b, c*
   3.2 solucionar(n - a, a, b, c),solucionar(n - b, a, b, c)
   3.3 solucionar(n - a, a, b, c),solucionar(n - c, a, b, c)
4. *e*
5. 5.1 return n, n - 1, n - 2
   5.2 b
6. *sumaAux(n, i + 1), sumaAux(n, i + 1)*
7. *7.1 (S, i, t - S[i])*
   7.2 (S, i, t)

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