Laboratorio Nro. 1: Recursión y Notación O

Omar Alexis Becerra Sierra
Universidad Eafit
Medellín, Colombia
oabecerras@eafit.edu.co

Juan José Tamayo
Universidad Eafit
Medellín, Colombia
jjtamayo@eafit.edu.co

2.3 Expliquen con sus propias palabras cómo funciona el ejercicio GroupSum5

El problema nos pide que se compruebe utilizando grupos constituidos con los elementos de un arreglo pueda lograrse llegar a un número “target”, para eso utilizamos un apuntador que ira recorriendo el arreglo y una variable “target” que servirá como el numero deseado.

Para llegar a una solución de este problema debemos utilizar dos llamados recursivos, el primero avanza una posición del arreglo añadiéndole 1 al índice y restándole al número pedido el número en el índice actual para verificar que la suma sea el “tarjet”. La segunda también le suma 1 al índice, pero no le resta al “target”, de esta manera se forman varios grupos(conjuntos). La condición de parada es si el índice es igual al tamaño del arreglo retorna true si “target” llego a cero, porque si en alguno de los llamados el target llego a cero quiere decir que la resta de los elementos del grupo llamado fue igual al target.
El problema también nos pide que los múltiplos de 5 se tengan en cuenta entonces antes de los llamados recursivos debe colocarse una condición que revisa si la posición actual es divisible por 5, en caso de serlo, como el problema pide que si el elemento que sigue del múltiplo 5 es un 1 no se tenga en cuenta, se hace otra condición que revisa si el elemento que le sigue al actual es un 1, en caso de serlo se hace una llamada recursiva sumándole 2 al índice, y se le resta al target la posición actual ósea el 5, en caso de que la condición no se cumpla entonces se hace otra llamada recursiva que le suma 1 al índice y le resta el 5 al target.

2.4 Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y Y 2.2 agréguenla al informe PDF

factorial:

para la condicion base:

T(n) = C1 +C2 si n = 0
T(n) = C4 +C5 si n = 1

Para la recursión:

T(n) = C2 * T (n - 1) ecuación de recurrencia
T(n) = C2 * T(n!)

Notación O:

(n) es O (C2 * n!)
T(n) es O (n!)

bunny Earts :

para la condicion base:

T(n) = C1+C2 n=0
T(n) = C3+C4 n=1

Para la recurcion

T(n) = C5+T(n-1) de lo contrario
T (n) = C*n+C'

Notacion O:

T (n) es O(C*n+C')
T (n) es O(C*n)
T (n) es O(n)

fibonacci:

para la condicion base:

T(n) = C1 + C2 + C3, si n <= 1

Para la recursión:

T(n) = C4 + T (n − 1) + T (n − 2) ecuación de recurrencia
$T(n) = C * 2^n + C$

Notación O:

T(n) es O ($C * 2^n$)
T(n) es O (2^n)

Triangle

para la condicion base:

T(n) = C1+C2, si n=0
T(n) = C3+C4, si n=1

Para la recursión:

T(n) = C5+T(n-1), de lo contrario
T(n) = C*n+C'

Notación O:

T (n) es O(C*n+C')
T (n) es O(C*n)
T (n) es O(n)

Sumdigits

Para la condicion base

T(n) = C1+C2, si n < 10

Para la recursion

T(n) = C3+C4+T(n/10) de lo contrario
T (n) = (C log(n))/log(10)+c'

Notacion O:

T (n) es O (C log(n)/log(10))
T (n) es O (C log(n)
T (n) es O (log(n))

groupSum6

para la condicion base:

T (n) = C1+C2, si n ≥ nums.length
T (n) = C3+T(n-1) si n=6

Para la recursion:

T (n) = 2T(n-1)
T (n)= $C2^n$+c`

Notacion O:

T (n) es O($C2^n$ +C')
T (n) es O($C2^n$)
T (n) es O(2^n)

groupNoAdj

para la condición base:

T (n) = C1+C2, si n ≥ nums.length

Para la recursion:

T (n) = 2T(n-1) de lo contrario
T (n)= $C2^n$+c`

Notacion O:

T (n) es O($C2^n$ +C')
T (n) es O($C2^n$)
T (n) es O(2^n**)**

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

groupSum5

para la condicion base:

T(n) = C1+C2, si n >= nums.length

Para la recursión

T(n) = C3+C4+T(n-2) si n%5=0 n<nums.length-1 y n+1= 1
T(n) = C3+T(n-1) si n%5=0 n>= nums.length-1 o n+1 ≠1
T(n) = 2T(n-1) de lo contrario

Notacion O:

T (n) es O($C2^n$ +C')
T (n) es O($C2^n$)
T (n) es O(2^n**)**

splitArray

para la condición base:

T (n)= C1+C2, si n≥ nums.length

Para la Recursión:

T (n)= 2T(n+1) de lo contrario

Notacion O:

T (n) es O($C2^n$ +C')
T (n) es O($C2^n$)
T (n) es O(2^n)

splitOdd10

para la condición base:

T(n) = C1 + C2 start >= nums.length

Para la Recursión:

T (n)= 2T (n + 1) de lo contrario

Notacion O:

T (n) es O($C2^n$ +C')
T (n) es O($C2^n$)
T (n) es O(2^n)

2.5 Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del ejercicio 2.4

Las variables n, y m de los cálculos de complejidad representan
Son una variable que afecta el número de instrucciones que ejecuta entre mayor la variable mayor será el tiempo en que se ejecutara el programa dependiendo de cómo se comporte la función si es exponencial o logarítmica se demorara más que si es lineal

3) Simulacro de preguntas de sustentación de Proyectos

3.1) ¿Qué aprendieron sobre Stack Overflow?

Stack overflow es un error que ocurre cuando se sobrepasa la capacidad de la pila de la variable que se le asigno, entonces esta se llena y lanza la excepción esto puede suceder por que la función no es la óptima para solucionar el problema o la variable que se asignó para el valor es muy pequeña para almacenarlo, la solución más viable y rápida es cambiar la variable por otra de mayor capacidad.

3.2) ¿Cuál es el valor más grande qué pudo calcular para Fibonacci?

Fibonacci (1000) aunque se demoró mucho en calcularlo
¿Por qué? Fibonacci genera muchos llamados recursivos y hasta que no solucione cada uno y se devuelva no devuelve el resultado
¿Por qué no se puede ejecutar Fibonacci con 1 millón?
Ejecutaría un llamado recursivo demasiadas veces y ocurriría dos cosas que la se llenaría la memoria y demoraría mucho tiempo en calcularlo

3.3 ¿Cómo se puede hacer para calcular el Fibonacci de valores grandes?

Para calcular números muy grandes java posee una librería que se puede importar java.math.BigInteger usada para almacenar números muy grandes y con demasiada capacidad eso evita que se desborde la pila.

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

3.4¿Qué concluyen sobre la complejidad de los problemas de CodingBat Recursion 1 con respecto a los de Recursión 2?

Se puede concluir que la mayoría de problemas de condingbat 2 son de la forma 2^n y los ejercicios de codingbat 1 la mayoría nos da una complejidad de n o de n! concluidos que cuando se realizan más de 1 llamado recursivo el problema nos dé de una forma de 2^n mientras que si solo hacemos un llamado recursivo el problema se comporta de manera lineal en conclusión es mejor utilizar un solo llamado recursivo ya que este no crece tan rápido que los de 2 llamados recursivos

4) Simulacro de Parcial

1. *start +1, nums, target*
2. *d*
3. **3.1** n, a, b, c
   3.2 n, res
   3.3
4. *a*
5.
   5.1
   Linea 2: return 1
   Linea 3: n + 1
   Linea 4: n – 2
   5.2 = d

6.
   6.1 0
   6.2 n.charAt(i+1) - ‘0’
7.
   7.1 S, i+1, t
   7.2 S, i+1, t-S[i]

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