Laboratorio Nro. 1: Recursión

Santiago Arredondo Quintero
Universidad Eafit
Medellín, Colombia
sarredondq@eafit.edu.co

Juan Esteban Alzate Uribe
Universidad Eafit
Medellín, Colombia
jealzateu@eafit.edu.co

2.3)

Lo que el problema pide es que se comprueba si con grupos formados con los elementos de un arreglo se puede llegar a un número "target", entonces para esto usamos un índice que ira recorriendo el arreglo y una variable "target" que servirá como el numero solicitado. La estructura principal de la solución de este problema consiste en dos llamadas recursivas, la primera avanza una posición del arreglo sumándole 1 al índice y le resta al número pedido el número en el índice actual (de esta manera se comprueba que la suma de los elementos sea el target), la segunda también le suma 1 al índice, pero no le resta al target, de esta manera se forman varios grupos.
La condición de parada es si el índice es igual al tamaño del arreglo, ósea llego al final, retorna true si el target llego a cero ya que si en alguno de los llamados el target llego a cero quiere decir que la resta de todos elementos del grupo formado en ese llamado fue igual al target. Como la recursión está dominada por una disyunción entonces solo se necesita que una llamado sea true para que todo sea true.
Como el problema pide que los múltiplos de 5 siempre se tengan en cuenta entonces antes de los llamados recursivos se pone un condicional que revisa si la posición actual es un múltiplo de 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)

Factorial: T(n) es O(n)
bunnyEars: T(n) es O(n)
bunnyEars2: T(n) es O(n)
triangle: T(n) es O(n)
sumDigits: T(n) es O(log(n))
groupSum: T(n) es O(2^n)
groupSum6: T(n) es O(2^n)
groupNoAdj: T(n) es O(2^n)
groupSum5: T(n) es O(2^n)
splitArray: T(n) es O(2^n)

2.4)

Representa una variable que puede afectar el número de instrucciones que ejecuta el programa, ósea es una variable que mientras más grande sea más lento se ejecutara el programa.

3) Simulacro de preguntas de sustentación de Proyectos

1. Lo que concluimos de Stack overflow es que es un error que ocurre cuando se hacen muchos llamados recursivos llenando la pila, esto puede suceder por que se trabaja con valores y funciones complejas, o porque el algoritmo está mal optimizado, se puede bajar la probabilidad de que aparezca este error aumentando el tamaño de la pila.
2. El mayor valor para el que pudimos calcular Fibonacci fue para 45 (demora 22s). Para valores mayores que este el programa ya se demoraba demasiado, por ejemplo, Fibonacci de 50 demoró 3 minutos y 44 segundos y además dio un valor negativo (-298632863). Al intentar calcular Fibonacci de un millón, se produce un error de StackOverflow.
3. una opción es hacer el programa con ciclos ya que esta no se va acumulando en el stack si no que seria paso por paso.
4. Podemos concluir que dada la complejidad de recursión2 y la recursión1, los de recursión2 toman más tiempo en ejecución.

4) Simulacro de Parcial

1. *a*
2. *b*
3. *length-1*
4. *x+1, a[i]*

5) Lectura recomendada (opcional)

a) Título
b) Ideas principales
c) Mapa de Conceptos

6) Trabajo en Equipo y Progreso Gradual (Opcional)

a) Actas de reunión
b) El reporte de cambios en el código
c) El reporte de cambios del informe de laboratorio

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