Laboratorio Nro. 1: Recursión y cálculo de complejidad.

Jose Joab Romero Humba
Universidad Eafit
Medellín, Colombia
jjromeroh@eafit.edu.co

Kevin Alexander Herrera Garces
Universidad Eafit
Medellín, Colombia
kaherrerag@eafit.edu.co

Manuel Alejandro Gutierrez Mejía
Universidad Eafit
Medellín, Colombia
magutierrm@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. El ejercicio sumGroup5 funciona de la siguiente manera: como el ejercicio pide retornar verdadero o falso si de alguna manera sumando un subgrupo de elementos del arreglo pueda dar el mismo valor que target, como nosotros resolvimos el ejercicio se hacen 3 llamados recursivos, el primero sucede si entra a un if el cual decide si el número el cual estemos recorriendo el arreglo es múltiplo de 5, si esto es verdadero entrara al llamado recursivo el cual restará el target por ese múltiplo de 5.
El segundo llamado recursivo hace que cualquiera número del arreglo sea restado con el target, para poder verificar si algún número del subgrupo podrá sumarse con otra y realizar la suma. El último llamado recursivo funciona como iterador, una comparación de este llamado recursivo sería la i++ de un ciclo for.
Finalmente la condición de parada de nuestro algoritmo será si la variable start es mayor o igual que el tamaño del arreglo, si esta condición se cumple retornara nuestro target con un valor de 0 para así retroceder en el árbol y retornar el booleano el cual usará.

3. Explicación de términos:
-T(n): La complejidad temporal en la cual se desarrolla el algoritmo.
- T(n-1): La complejidad temporal se ve reducida un término menos.
- c_n: Una constante que se le suma a nuestra complejidad algorítmica.
- O: notación O, la cual nos ayuda a encontrar el comportamiento asintótico de un algoritmo.

4. El error de StackOverFlow es muy común cuando se está aprendiendo recursión, ya que si no se hace una condición correcta de parada o un iterador en nuestra función recursiva, se convierte en un bucle infinito el cual hace que nuestro tamaño de pila sea más grande de lo que debería ser; en conclusión el programa nunca terminara.
5. El valor más grande que se calculó fue el 50, debido a que valores más grandes con llevaban mucho mas tiempo y debido a las capacidades del computador esto podría tardar ma.
abria problema al ejecutar 1 millón debido inicialmente al tiempo que esto conlleva y a que superaría las capacidades de la memoria
7. Se puede notar que los ejercicios de recursión 1 tienen una menor complejidad lo que los lleva a gastar menos recursos y ejecutarse más rápido que los que se proponen en recursión 2

4) Simulacro de Parcial

1. return sumaGrupo(star+1, nums, target-nums[start] || sumaGrupo(start+1, nums, target);
2. d

3. linea 4: int res = solucionar(n-a, b, c, a) + 1;
   linea5: res = Math.max(res , Math.max(solucionar(n-a, b, a, c)+1, Math.max(solucionar(n-a, c, a, c)+1, solucionar(n-b, a, c, b)+1)));
   linea 6: res = Math.max(res , Math.max(solucionar(n-a, a, b, c)+1, Math.max(solucionar(n-b, a, b, c)+1, solucionar(n-c, a, b, c)+1)));

5. Línea 2. return n
   Línea 3. n-1
   Línea 4. n.2
6. Linea 10 return Suma(n,i+2);
   Linea 12 sumaAux(n,i+1)
7. Línea 9. return comb (S, i+1, t-s[i])
   Línea 10. comb (S, i+1, t);

8 . Linea 9 : return 0;
Linea 13: return ni - nj;