ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 1
Escribir el tema del laboratorio

David Vergara Patiño
Universidad Eafit
Medellín, Colombia
dvergarap13@eafit.edu.co

1. **Simulacro de proyecto**
   - **1.1.** En el punto 1.1 la entrada de la función son dos strings, el parámetro de parada es cuando el tamaño de las strings llegue a cero, lo que hace la recursión primero es que compara si los últimos valores de las string son iguales si acumula 1 y llama otra vez la función pero sin el ultimo valor de las dos string y por ultimo este método empieza a abrirse como árbol, y que retorna el maximo entre la recursión de quitarle el último elemento al string 1 y de quitarle el último elemento al string 2, así se asegura que el substring que se escoja es el máximo
   - **1.2.** En el problema ,del tablero el n pertenece al área del table de 2*n cuadrados, la condición de para es cuando n<=2 ya que en ese casi retorna n, ya que en ese momento solo cabria ese n, después se ve que la recursión es de n-1 y de n-2, ya que se le van quitando baldosas , se puede ver que quitar 3 baldosas es quitar 1 y después quitar 2, así que en eso se puede resumir y retorna la suma de las 2.
2. **Ejercicios en línea:**
   - **2.1. Recursion1**
     - **2.1.1.** Factorial, en la factorial el n, es el numero al que se le quiere sacar la factorial, la para es cuando n<=1 ya que la factorial de 0 y 1 es, y en la recursión sabemos que la factorial es el número actual por la factorial del n-1
     - **2.1.2.** Bunnyears, aquí n es el numero de conejos, sabemos que cuando tenemos 0 conejos hay 0 orejas, y el llamo recursivo, es 2+ el número de conejos -1
     - **2.1.3.** Fibonacci, sabemos que Fibonacci de 0 es 0 y Fibonacci de 1 es 1, así la llamada recursiva es el Fibonacci actual es el Fibonacci de n-1 + el Fibonacci de n-.2
     - **2.1.4.** Bunnyears2, aquí la diferencia es que si el numero del conejo es par tiene 3 orejas y sino tiene 2, la condición de para es lo mismo, pero ahora cada llamado hay que mirar si es par en vez de sumarle 2 se le suma 3 y sino es par entonces se le suma 2
     - **2.1.5.** En el triángulo, sabemos que cuando la fila es 0 tiene 0 bloques y si la fila es 1 tiene 1, por lo que cada fila tiene n bloques, así el llamado recursivo es desde la fila mas grande donde seria n+ en numero de bloques de n-1 y así sucesivamente hasta 0
   - **2.2. Recursión 2**
     - **2.2.1.** Groupsum6, start siempre empieza en 0, que es la posición, nums el arreglo y, target es lo que debe sumar, la condición de parada es cuando start llegue al

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

tamaño del arreglo ahí mira si target 0 retorna true, después mira si el arreglo en la posición start es 6, entonces al target le resta 6 osea lo incluye y sigue, y por ultimo tenemos un ó que lo que hace es que nos abre el árbol en dos, que son en el que se incluye el numero de la posición start y otro que no lo incluye.

2.2.2. Groupsumadj Es casi igual que el group sum, solo que pone la condición se agrega uno, el siguiente no se puede agregar, así que cuando llegamos a la recursión, en la parte de el ó que se incluye en vez de sumarle 1 a start se le suma 2.

2.2.3. Groupsum5, aquí la condición es que si en la posición start es múltiplo 5, entonces se agrega y si el que sigue es uno el 1 no se agrega el 1, así que la primera condición que se pone es que si es el arreglo en la posición start es múltiplo de 5, si se pasa a la siguiente condición, hay que comprobar 2 cosa que si exista siguiente numero y que ese número es igual a 1, así que si cumple todo, se agrega el múltiplo de 5 y además el start incremente en 2, sino el siguiente no es 1 entonces solo se agrega el múltiplo de 5 y start se le suma 1, y sino es ninguno, se hace lo quiso que groupsum.

2.2.4. Groupsumclump, este la condición que tiene es que si hay repetidos consecutivos, en arreglo o se agregan todo o ninguno, así que necesitamos un ciclo, que empieza start y que compruebe que no se pase del tamaño del arreglo y que el arreglo en posición i=arreglo en posición start, si los suma, y la ultima parte es igual que groupsum, solo que en vez de seguir en start, sigue donde termina el ciclo en i.

2.2.5. Splitarray, tiene start que es la posición en el arreglo, nums, que es el arreglo y arreglo 1 y arreglo 2 que son los arreglos que debo llenar y comparar, todo excepto nums, es 0. Así el objetivo de parada es cuando start legue el tamaño del arreglo, ahí veo si arreglo1 y arreglo 2 son iguales devuelvo true, sino hago el llamado recursivo que seria, aumentar start y tiene dos ramas, en una le sumo al arreglo 2, la posición del arreglo en start y la otra lo mismo solo que al otro arreglo.

3. Simulacro de preguntas de sustentación de Proyectos

3.1. La complejidad asintótica esta dada por T(N)=T(n-2)+T(n-2)+c
3.2. grafica y datos en Excel adjunto, para n=50 el t es 2361
3.3. NO es valido en puerto Antioquia ya que su complejidad es demasiado grande de 2 elevado a n
3.4. No
3.5. Recursion1
  3.5.1. Fibonaci T(n)=T(n-1)+T(n-2) +c
  3.5.2. Bunnyears T(n)=c+T(n-1)
  3.5.3. Factorial T(n)=T(n-1) +c
  3.5.4. Bunnyears2 T(n)=T(n-1)+c
  3.5.5. Triangulo T(n)=T(n-1)+c
  3.5.6. Recursion 2
    3.5.6.1. Groupsum6 T(n)=T(n-1) +T(n-2)+c
    3.5.6.2. Groupnoadj T(n)= T(n-1) +T(n-2)+c
    3.5.6.3. Groupsum5 T(n)=T(n-1) +T(n-2)+c
    3.5.6.4. Groupsumclump T(n)=t(n-1) +T(n-2)+n+c
    3.5.6.5. Split array T(n)=T(n-1) +T(n-2)+c

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

3.6. Del calculo anterior n, es numero de repeticiones que el algoritmo tiene que hacer para el peor de los casos.

4. *Simulacro de Parcial*

4.1 *Linea3: return True Linea4:s.charAt(0)==s.charAt(s.length()-1)*
4.2 d
4.3 *solucionar*
- *4.3.1* *solucionar(n-b,a,b,c)*
- *4.3.2* *math.max(res,solucionar,n-a,a,b,c))*
- *4.3.3* *math.max(res,solucionar,(n-c,a,b,c)*

4.4 *no*
4.5 *ways*
- *4.5.1* *linea 3: if T==0 return 1 if t<0 return 0*
- *4.5.2* *b*

4.6 suma
- *4.6.1* suma(n,i+2)
- *4.6.2* suma(n,i+1)

4.7 no
4.8 cuantas
- *4.8.1* return 0
- *4.8.2* ni+nj

4.9 no
4.10 6
4.11 LUCAS
- *4.11.1* Lucas(n-1)+lucas(n-2)
- *4.11.2* C

4.12 Conejo
- *4.12.1* Return sat
- *4.12.2* Math.max(fi,fj)
- *4.12.3* Sat

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473