**Laboratorio Nro. 1  
Recursión**

|  |  |
| --- | --- |
| **Martín Sánchez Reyes**  Universidad Eafit  Medellín, Colombia  msanchezr@eafit.edu.co | **Juan Martín Uribe Forero**  Universidad Eafit  Medellín, Colombia  jmuribef@eafit.edu.co |

***2)* Explicación Código en Línea**

***2.1***Triangle: el caso base es cuando el número de filas es cero. En esta situación, no hay bloques. En el resto de filas, cada fila agrega un número de bloques igual al de su posición (fila tres, tendrá tres bloques, por ejemplo), por lo que, para todos los otros casos distintos a cero, se devuelve el número de filas más la función con el parámetro reducido en uno.

***2.2*** Count7: el caso base es cuando el parámetro sea menor que diez, aquí se verifica si el número es igual a 7, y se devuelve 1 o 0. Para todos los demás casos, como evaluamos dígito a dígito, verificamos lo mismo para el último dígito y devolvemos 1 o 0, sumado al método con el parámetro sin el número ya evaluado.

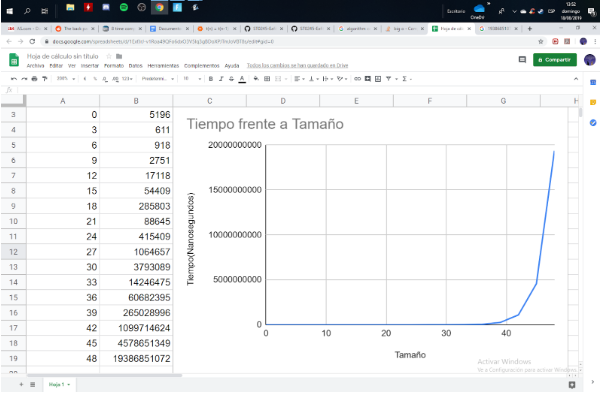
***2.3*** CountX: el caso base es cuando la longitud de la cadena es 0 o 1. Se verifica que cumpla la condición para contar como ocurrencia. En los demás casos, se verifica la condición para el último elemento de la cadena y la ocurrencia (si ocurre) se suma a la invocación del método con una cadena que va hasta el penúltimo caracter.

***2.4*** PowerN: por propiedad de los exponentes, es lo mismo una constante elevada a un entero n, que la misma constante elevada a n-1, multiplicada por sí misma. Este concepto se aplica al caso recursivo y tiene la condición de parada del exponente 0.

***2.5*** CountHi: el caso base es cuando la cadena tiene menos de dos elementos, lo que devuelve 0 ocurrencias adicionales. El caso recursivo evalúa que los dos últimos caracteres coincidan, y suma esto al método con la cadena sin el último caracter.

**3) Simulacro de preguntas de sustentación de Proyectos**

**3.1** La complejidad del algoritmo usado para calcular las posibles formas en las que se puede formar un conjunto de rectángulos de 2x1 en un tablero de 2xn es similar a la secuencia de fibonacci(recursivo). Siendo la función inicial T(n)=t(n-1)+t(n-2)+c1, donde la constante es el caso base (código en github). La solución, proporcionada por wolfram: t(n)=c1\*f(n)+c2\*L(n)-c1, la cual involucra secuencias de Fibonacci y Lucas. La notación asintótica seria O(2^n).

**3.2**En el caso específico de un tablero de 2\*50, se demoró 19 segundos en calcular las todas las posibles formas. 

**3.3**Este algoritmo no es adecuado para problemas a gran escala, como en el caso de Puerto Antioquia, ya que es de tipo exponencial y solo está por detrás de los factoriales respecto a su tiempo de ejecución. Es decir que a problemas de gran escala este algoritmo y su complejidad son ineficientes y no debería de aplicarse para dicha situación.

***4) Simulacro de Parcial***

4.5.1

linea 2: return n;

linea 3: n-1

linea 4: n-2

4.5.2

T(n)=c+T(n-1)+T(n-2)

4.6

linea 10: return n.charAt(i);

linea 12: n.charAt(i) - ’0’

4.8

linea 9: return 0;

linea 10: return nj;

4.12

linea 13: return 1:

linea 17: Math.max(fi,fj);

linea 18: return sat;