Laboratorio Nro. 1
Escribir el tema del laboratorio

Manuela Herrera López
Universidad Eafit
Medellín, Colombia
mherreral@eafit.edu.co

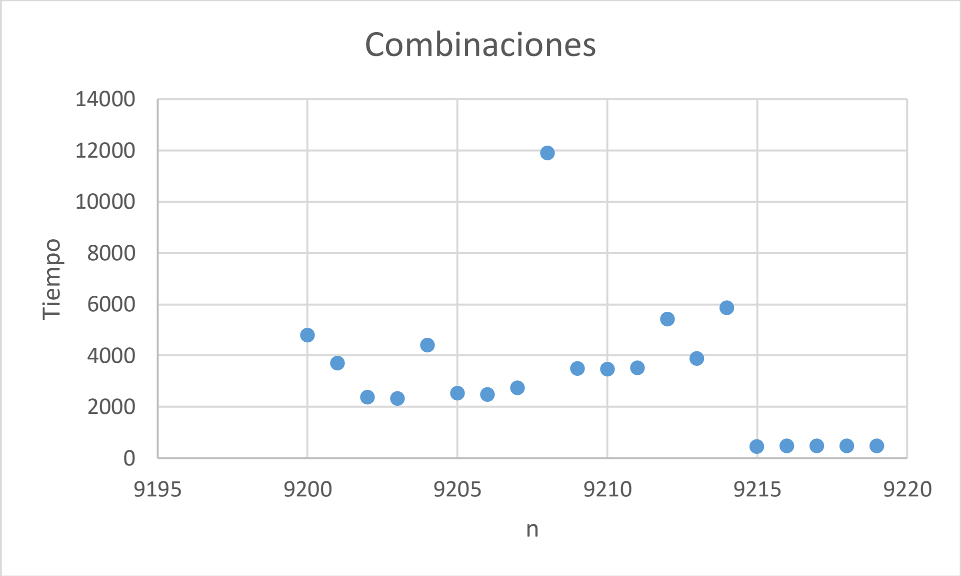
Samuel Palacios Bernate
Universidad Eafit
Medellín, Colombia
sdpalaciob@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1

T(n) = T(n-1) + c_2
T(n) = O((n-1) + c_2) por definición de big O
O(n) por propiedad de la suma

3.2



Para llenar el tablero de 50 x 2 se demoraría aproximadamente 1000 nanosegundos.

3.3 Claro que sería viable, porque es un algoritmo que se ejecuta en un tiempo lineal, dado por la multiplicación de una constante con nuestro n, aunque sería más eficaz con un algoritmo cuya complejidad asintótica sea O(logX) u O(c1), pero este también podría funcionar.

3.4 Nuestro caso base en GroupSum5 es si Start (donde se comienza a evaluar si hay elementos que sumados den target) es mayor o igual que la longitud del arreglo que tenemos como parámetro. Si esta condición se cumple, se retorna un falso; lo que en otras palabras sería que ya no hay nada más por evaluar.
Si nuestro caso base no se cumple, entramos a la recursión de la siguiente forma;

-La primera es que si el valor del arreglo en la posición start es divisible entre 5 y esto no deja ningún residuo, entonces evaluamos si la posición de inicio (start) es menor que la longitud del arreglo, y también, en este caso nos aseguramos de que el número que le sigue a la posición start del arreglo es 1, siendo así, llamamos de nuevo a la función, modificando el índice y el target de la siguiente forma: le enviamos el índice más dos, para que de esta forma nos saltemos el uno, y al target le restamos lo que haya en nuestro arreglo en la posición de inicio. Si lo anterior no pasa, simplemente retornamos llamando a la función sumándole un 1 al índice (ya que no se cumple que el número que hay después del número que es múltiplo de 5 es uno), y de la misma forma le enviamos nuestro target sin lo que hay en el arreglo en la posición de inicio.
Y al final se retorna verdadero o falso dependiendo de si se cumplen o no las llamadas a la función recursiva, ambas una posición adelante del estar, pero una sin el target modificado y la otra con una resta entre el target y el valor del arreglo en la posición start.
Esto se repite hasta que start sea igual a la longitud del arreglo, y devolvería verdadero si en el arreglo se encontraron los justos valores del target, y falso si aún el target está lleno.

3.5
Para los ejercicios 2.1:

powerN: T(n) = c_2*n + c1
T(n)= O(c_2*n + c1) por definición de big O
O(c_2*n) por regla de la suma
O(n) por regla del producto

countX: T(n) = T(n) + c_1
O(n + c_1) por definición de big O
O(n) por regla de la suma

bunnyEars: T(n) = c_2 + T(n-1)
O(c_2 + n-1) por definición de big O
O(n) por regla de la suma

Count7: T(n) = T(n/10) + c_1
O(n/10 + c_1) por definición de big O
O(n/10) por regla de la suma
O(log n) por regla de los logaritmos naturales

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

Fibonacci: T(n) = T(n-1) + T(n-2) + c_2
O(2^n)

Para los ejercicios de 2.2:

GroupSum6: T(n) = 2 * T(n-1) + c_2
O(n) = c_1 * 2^(n-1)
O(n) = c_1 * 2^n * 2^{-1} --> por algebra
O(n) = 2^n --> por regla del producto
O(2^n)

GroupSum5: T(n)= 2 * T(n-1) + c_2
O(n) = c_1 * 2^(n-1)
O(n) = c_1 * 2^n * 2^{-1} --> por algebra
O(n) = 2^n --> por regla del producto
O(2^n)

groupNoAdj: T(n)= 2 * T(n-1) + c_2
O(n) = c_1 * 2^(n-1)
O(n) = c_1 * 2^n * 2^{-1} --> por algebra
O(n) = 2^n --> por regla del producto
O(2^n)

splitArray: T(n) = 2 * T(n-1) + c_2
O(n) = c_1 * 2^(n-1)
O(n) = c_1 * 2^n * 2^{-1} --> por algebra
O(n) = 2^n --> por regla del producto
O(2^n)

splitOdd10: T(n) = 2 * T(n-1) + c_2
O(n) = c_1 * 2^(n-1)
O(n) = c_1 * 2^n * 2^{-1} --> por algebra
O(n) = 2^n --> por regla del producto
O(2^n)

3.6

Para 2.1:
- En el caso de powerN, n significa la potencia a la que se quiere elevar la base
- En el caso de countX, la n significa una cadena de texto en la cual se va a evaluar si un elemento es una equis minúscula
- En el caso de bunnyEars, la n significa lo que falta para que se termine la recursión (cantidad de conejos)
- En el caso de count7, la n significa el dígito más a la izquierda de un número

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

- En el caso de Fibonacci, n significa el número al cual se le va a hallar su equivalente en la sucesión

Para 2.2:
- En todos los casos “n” hace referencia a lo que falta para terminar de recorrer el arreglo dependiendo de las condiciones.

4) Simulacro de Parcial

4.1 *Start + 1, nums, target*
4.2 A.
4.3

4.4 *E.*

4.5

1.5.1
Línea 2: return n;
Líneas 3 y 4: return formas(n - 1) + formas(n - 2);

1.5.2. B

4.6 **Línea 10**: sumaAux(n.substring(I + 2), 0);
Línea 12: return (n.charAt(i) - ‘ 0 ’ + sumaAux(I + 1, 0));

4.7 **Líneas 9 y 10**: return comb(S, i+1, t) || comb(S, i+1, t - S[i])

4.8 **Línea 9**: return 0;
Línea 13: suma = ni + nj;

4.9 La función retorna 22 (c)

4.10 La función retorna 6 (b)

4.11 **Línea 4**: n - 1 + lucas(n - 2);
Complejidad es C.
4.12 **Línea 13**: return sat;
Línea 17: Math.max(fi, fj);
Línea 18: return sat;

5) Lectura recomendada (opcional)

Mapa conceptual

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