INFORME DE CORRECCIÓN

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN, UNIVERSIDAD DEL VALLE
13/10/2022

SEBASTIÁN IDROBO AVIRAMA

Informe de procesos

Función mcdTFA

Para esta función se tiene un proceso recursivo lineal. Esto debido a que acumula en memoria el primer elemento de una determinada lista para multiplicarlo con el resto de las demás listas introducidas. Esto queda más que claro al observar la ejecución del programa de la siguiente manera:

$mcdTFA(List(3,1,1),\ List(2,0,3),\ List(2,3,5))$
$-> 2^{2} * mcdTFA(List(1,1),\ List(0,3),\ List(3,5))$
$-> 4 * 3^{0} * mcdTFA(List(1),\ List(3),\ List(5)$
$-> 4 * 1 * 5^{1} * mcdTFA(List(),\ List(),\ List())$
$-> 3 * 1 * 5 * 1$
$->\ 20$ [1]

Función mcdEBez

Para esta función se tiene un proceso recursivo iterativo, específicamente en la función encontrarCoeficientes. Esto debido a que, al no ser *r1* igual a 0, la función vuelve a llamarse iterativamente sin guardar en la memoria un valor determinado. Esto queda más claro al observar la ejecución del programa de la siguiente manera:

$mcdEBez(45,\ 36)$
$-> encontrarCoeficientes(45,\ 36,\ 1,\ 0,\ 0,\ 1)$
$-> encontrarCoeficientes(36,\ 9,\ 0,\ 1,\ 1,\ -1)$
$-> encontrarCoeficientes(9,\ 0,\ 1,\ -4,\ -1,\ 5)$
$-> (9, 1, -1)$

[1] Este proceso se encuentra simplificado, esto con el objetivo de demostrar que la llamada recursiva de la función guarda en memoria los resultados previos obtenidos, siguiendo una forma Expansión-Contracción propia del proceso recursivo lineal. Las siguientes demostraciones de los procesos también se encuentran simplificados

Función fibonacciA

Para esta función se tiene un proceso recursivo lineal de árbol. Esto debido a que sigue el modelo de forma de Expansión-Contracción, guardando en memoria la suma de del llamado recursivo de la misma función con *n-1* y *n-2*

Una manera ilustrada de demostrar la recursividad de árbol se puede ver en el siguiente diagrama:

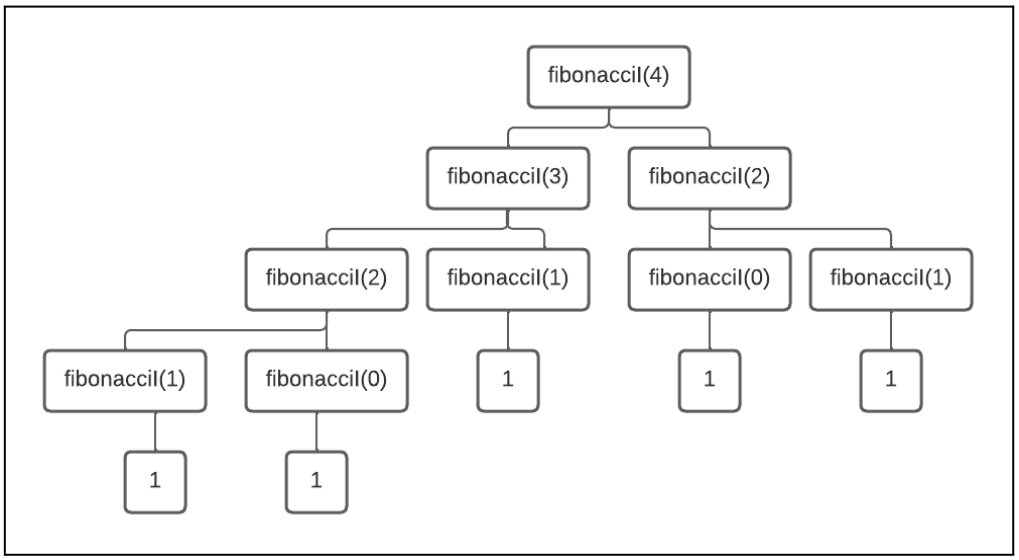

Diagrama que se encuentra reflejado en la ejecución del código:

$fibonacciA(4)$
$-> fibonacciA(3) + fibonacciA(2)$
$-> fibonacciA(2) + fibonacci(1) + fibonacci(0) + fibonacci(1)$
$-> fibonacciA(1) + fibonacci(0) + fibonacci(1) + fibonacci(0) + fibonacci(1)$
$-> 1 + 1 + 1 + 1 + 1$
$-> 5$

Función fibonacciI

Para esta función se tiene un proceso recursivo iterativo, específicamente la función fibIterativo. Esto debido a que, al no ser *contador* igual a 0, la función vuelve a llamarse iterativamente sin guardar en memoria un valor determinado, más que su propio llamado a sí misma otra vez. Esto queda más claro al observar la ejecución del programa de la siguiente manera:

$fibonacciI(4)$
$-> fibIterativo(0, 1, 0, 5)$
$-> fibIterativo(1, 0, 1, 4)$
$-> fibIterativo(1, 1, 1, 3)$
$-> fibIterativo(2, 1, 2, 2)$
$-> fibIterativo(3, 2, 3, 1)$

$-> \ fibIterativo(5,\ 3,\ 5,\ 0)$
$-> \ 5$

Informe de corrección

Argumentación sobre la corrección

Función mcdTFA

Vamos a demostrar que: Siendo $n = p_1^{\,i_1} * p_2^{\,i_2} * p_3^{\,i_3}$, y $m = p_1^{\,j_1} * p_2^{\,j_2} * p_3^{\,j_3}$:

$$\forall n,\ m\ \in N \setminus \{0\}\colon\ mcdTFA((i_1,\ i_2,\ i_3),\ (j_1,\ j_2,\ j_3),\ (p_1,\ p_2,\ p_3))\ ==\ mcd(n, m)$$

Caso base: $ln\ =\ \{\}$. Vamos a calcular que devuelve $mcd(n, m)$ usando el modelo de substitución

$$mcdTFA(\{\},\ lm,\ primos)\ \rightarrow\ if\ (ln\ ==\ \{\})\ 1\ else\ if\ (0\ <\ lm.head)\ \{...\}\ \rightarrow 1$$

Por otro lado:

$$mcd(1, m)\ = p_1^{\,0} * p_2^{\,0} * p_3^{\,0} = 1$$

Por lo que podemos concluir que:

$$mcdTFA(\{\},\ lm,\ primos)\ ==\ mcd(1, m)$$

Caso de inducción: $n = p_1^{\,i_1} * p_2^{\,i_2} * p_3^{\,i_3},\ m = p_1^{\,j_1} * p_2^{\,j_2} * p_3^{\,j_3}$, siendo $i < j$. Hay que demostrar que:

$$mcdTFA((i_1,\ i_2,\ i_3),\ (j_1,\ j_2,\ j_3),\ (p_1,\ p_2,\ p_3))\ ==\ p_1^{\,min(i_1,j_1)} * p_2^{\,min(i_2,j_2)} * p_3^{\,min(i_3,j_3)} \rightarrow$$
$$mcdTFA((i_1 + 1,\ i_2 + 1,\ i_3 + 1),\ (j_1 + 1,\ j_2 + 1,\ j_3 + 1),\ (p_1,\ p_2,\ p_3))\ ==$$
$$p_1^{\,min(i+1_1,j_1+1)} * p_2^{\,min(i_2+1,j_2+1)} * p_3^{\,min(i_3+1,j_3+1)}$$

Empecemos por calcular que devuelve $mcdTFA$ usando el modelo de substitución:

$$mcdTFA((i_1 + 1,\ i_2 + 1,\ i_3 + 1),\ (j_1 + 1,\ j_2 + 1,\ j_3 + 1),\ (p_1,\ p_2,\ p_3)) \rightarrow$$
$$if\ ((i_1 + 1,\ i_2 + 1,\ i_3 + 1)\ ==\ \{\})\ else\ if\ (i_1 + 1 < j_1 + 1)\ p_1^{\,i_1+1} * mcdTFA((i_2 + 1,\ i_3 + 1),$$
$$(j_2 + 1,\ j_3 + 1),\ (p_2,\ p_3))\ \rightarrow$$
$$p_1^{\,i_1+1} * p_2^{\,i_2+1} * mcdTFA((i_3 + 1),\ (j_3 + 1),\ (p_3)) \rightarrow$$
$$p_1^{\,i_1+1} * p_2^{\,i_2+1} * p_3^{\,i_3+1} * mcdTFA(\{\},\ \{\},\ \{\}) \rightarrow p_1^{\,i_1+1} * p_2^{\,i_2+1} * p_3^{\,i_3+1} * 1$$

Por otro lado:

$$mcd((p_1^{i_1+1} * p_2^{i_2+1} * p_3^{i_3+1}),\ (p_1^{j_1+1} * p_2^{j_2+1} * p_3^{j_3+1})) \rightarrow\ p_1^{i_1+1} *\ p_2^{i_2+1} *\ p_3^{i_3+1} * 1$$

Por lo que podemos concluir que:

$$\forall n,\ m\ \in N \setminus \{0\}\colon mcdTFA((i_1,\ i_2,\ i_3),\ (j_1,\ j_2,\ j_3),\ (p_1,\ p_2,\ p_3))\ ==\ mcd(n,m)$$

Función mcdEBez

—

Función fibonacciA

Vamos a demostrar que:

$$\forall n \in N:\ fibA(n)\ ==\ f_n$$

Casos base: n <= 1. Vamos a calcular qué devuelve f_n usando el modelo de substitución

$$fibA(0) \rightarrow if(0\ ==\ 0||0\ ==\ 1)\ 1\ else\ fibA(0\ -\ 1)\ +\ fibA(0\ -\ 2) \rightarrow 1$$
$$fibA(1) \rightarrow if(1\ ==\ 0||1\ ==\ 1)\ 1\ else\ fibA(1\ -\ 1)\ +\ fibA(1\ -\ 2) \rightarrow 1$$

Por otro lado:

$$f_0 = 1$$
$$f_1 = 1$$

Por lo que podemos concluir que:

$$f_{n\leq 1} == fibA(n \leq 1)$$

Caso de inducción: $n\ =\ k\ +\ 1,\ k\ \geq\ 1$. Hay que demostrar que:

$$fibA(k)\ ==\ f_k \rightarrow fibA(k\ +\ 1)\ ==\ f_k + f_{k-1}$$

Empecemos por calcular qué devuelve $fibA$ usando el modelo de sustitución:

$$fibA(k\ +\ 1) \rightarrow if(k\ +\ 1\ ==\ 0\ ||\ k\ +\ 1\ ==\ 1)\ 1\ else\ fibA(k\ +\ 1\ -\ 1)\ +\ fibA(k\ +\ 1\ -\ 2) \rightarrow$$
$$fibA(k)\ +\ fib(k\ -\ 1) \rightarrow\ f_k + f_{k-1}$$
$$fibA(k\ +\ 1) \rightarrow f_k + f_{k-1}$$

Por otro lado

$$f_k = f_{k-1} + f_{k-2} \rightarrow f_{k+1} = f_k + f_{k-1}$$

Por lo que podemos concluir por inducción que:

$$\forall n \in N : fibA(n) == f_n$$

Función fibonacciI

—

Casos de prueba

Función mcdTFA

Para los casos de pruebas, la función fue puesta a prueba con listas de tamaño 1 hasta tamaño 4, obteniendo de todos aquellos los resultados esperados. Adicional a esto, también fue puesto a prueba el intercambio de los parámetros *ln* y *lm* respecto a la prueba tres, en la prueba cuatro[2].

Para esta función, los casos de prueba son suficientes para confiar en la corrección de programa para esta función, debido a que al exponer los parámetros de la función a listas con diferentes tamaños, demuestra un correcto funcionamiento al retornar el resultado esperado. De la misma manera, no importa si intercambiamos entradas en los parámetros *ln* y *lm*, la función retorna el resultado esperado.

Función mcdEBez

Para los casos de pruebas, la función fue puesta a prueba con parámetros que me arrojaran al caso base, y respecto a las cuatro pruebas restantes, son pruebas con parámetros aleatorios.

Para esta función, los casos de prueba son suficientes para confiar en la corrección de programa para esta función, debido a que las pruebas demuestran un correcto funcionamiento al retornar el resultado esperado. Además de que los coeficientes se encuentran en las posiciones correctas.

Función fibonacciA

Para los casos de pruebas, la función fue puesta a prueba con los parámetros que me arrojaran al caso base, y respecto a las cuatro pruebas restantes, son pruebas con parámetros aleatorios.

Para esta función, los casos de prueba son suficientes para confiar en la corrección de programa para esta función, debido a que las pruebas demuestran un correcto funcionamiento tanto para las funciones parámetros de caso base, como también a aquellos que me generan un proceso recursivo. Al tener las funciones con un proceso recursivo un funcionamiento correcto, son suficientes para obtener un resultado confiable de la función.

Función fibonacciI

Para los casos de prueba, la función fue puesta a prueba con los parámetros que me arrojaran al caso base, y respecto a las cuatro pruebas restantes, son pruebas con parámetros aleatorios.

---

[2] mcdTFA(List(3,1,1), List(2,0,3), List(2,3,5)) –> mcdTFA(List(2,0,3), List(3,1,1), List(2,3,5))

Para esta función, los casos de prueba son suficientes para confiar en la corrección de programa para esta función, debido a que las pruebas demuestran un correcto funcionamiento tanto para las funciones con parámetros de caso base, como también para aquellos que me generan un proceso recursivo. Al tener las funciones con un proceso recursivo un funcionamiento correcto, son suficientes para obtener un resultado confiable de la función.

Conclusiones

Se concluye de este taller que:

- La recursividad es una herramienta potente para resolver problemas al subdividir el problema en cuestión a sus términos más pequeños, por lo que es posible obtener un código consistente frente a los casos base y su llamada recursiva.
- Aunque existen 2 tipos de proceso para realizar una función en programación funcional, las cuales son la recursión lineal y recursión iterativa, es posible afirmar que una recursión iterativa va a ser mejor en la mayoría de los casos para resolver un problema, debido a que optimiza el código de tal manera que acorta los tiempos de procesamiento y uso de memoria.