ALGORITMIA

PRÁCTICAS – SESIÓN 2.2

VENTAJAS Y DESVENTAJAS DE LA RECURSIVIDAD

- Da solución a problemas de forma natural, sencilla, comprensible y con un esfuerzo de razonamiento menor.
- Da lugar a algoritmos más compactos
- Presenta facilidad para verificar formalmente que la solución es correcta
- En general, las soluciones recursivas son más ineficientes en tiempo y en espacio que las versiones iterativas, esto se debe al mecanismo de llamadas continuas a la función y al paso de parámetros, al uso de una pila donde cada uno de sus elementos reserva espacio para una activación de la función recursiva (direcciones de parámetros y variables locales).

En esta sección se muestran las **versiones iterativas** de los esquemas genéricos **recursivo simple no final** y **recursivo simple final** vistos en la sesión de prácticas anterior.

Recordemos que:

- **función recursiva simple** es cuando la función recursiva genera a lo sumo una llamada interna por cada llamada externa
- **función recursiva no final** es cuando la función c (función de combinación) es necesaria, esto es, $f(\bar{x}) = c(f(s(\bar{x})), \bar{x})$
- **función recursiva final** es cuando la función c (función de combinación) no es necesaria, esto es, $f(\bar{x}) = f(s(\bar{x}))$

TRANSFORMACIÓN A ITERATIVO DE UNA RECURSIÓN SIMPLE Y NO FINAL

- La transformación da lugar a **dos bucles**.

- El **primero** corresponde al **descenso en la cadena de llamadas recursivas**, transformando los parámetros $\bar{x}$ de la llamada en curso en los parámetros s($\bar{x}$) de la llamada sucesora, hasta encontrar el valor $\bar{x}$ correspondiente al caso trivial. Por ello se aplica repetidamente la función sucesor, función s.

- La asignación posterior al primer bucle calcula el primer resultado $\bar{y}$, que corresponde al caso trivial.

- El **segundo bucle** representa el **ascenso en la cadena de llamadas**, aplicando reiteradamente la función c (función de combinación) para calcular los resultados de la llamada en curso en función de los de la llamada sucesora. Antes de aplicar la función c es necesario recuperar los parámetros $\bar{x}$ de la llamada en curso a partir de los de la llamada sucesora. Para ello, se aplica la función $s^{-1}(\bar{x})$, eso es, la inversa de la función sucesor (función s).

A continuación se muestra el esquema general de una función recursiva no final y de su versión iterativa.

TRANSFORMACIÓN A ITERATIVO DE UNA RECURSIÓN SIMPLE Y NO FINAL

{ Q($\bar{x}$) }

función f($\bar{x}$: T_1) retorna ($\bar{y}$: T_2)

  caso

    Bt ($\bar{x}$) $\rightarrow$ triv($\bar{x}$)

    Bnt ($\bar{x}$) $\rightarrow$ c(f(s($\bar{x}$)), $\bar{x}$)

  fcaso

ffunción

{ R($\bar{x}$,$\bar{y}$) }

Función recursiva simple y no final

TRANSFORMACIÓN A ITERATIVO DE UNA RECURSIÓN SIMPLE Y NO FINAL

{ Q($\bar{x}$) }

función f($\bar{x}$: T_1) retorna ($\bar{y}$: T_2)

caso

Bt ($\bar{x}$) $\rightarrow$ triv($\bar{x}$)

Bnt ($\bar{x}$) $\rightarrow$ c(f(s($\bar{x}$)), $\bar{x}$)

fcaso

ffunción

{ R($\bar{x}$,$\bar{y}$) }

Función recursiva simple y no final

{ Q($\bar{x}_{inicial}$) }

función f($\bar{x}_{inicial}$: T_1) retorna ($\bar{y}$: T_2)

var $\bar{x}$: T_1 fvar

$\bar{x} = \bar{x}_{inicial}$

mientras Bnt($\bar{x}$) hacer

$\bar{x}$ = s($\bar{x}$)

fmientras

$\bar{y}$ = triv($\bar{x}$)

mientras $\bar{x} \neq \bar{x}_{inicial}$ hacer

$\bar{x}$ = s^{-1}($\bar{x}$)

$\bar{y}$ = c ($\bar{y}$, $\bar{x}$)

fmientras

retorna $\bar{y}$

ffunción

{ R($\bar{x}_{inicial}$, $\bar{y}$) }

Función iterativa equivalente

EJEMPLO FUNCIÓN RECURSIVA SIMPLE Y NO FINAL: POTENCIA

$Q \equiv \{ a \geq 0 \wedge n \geq 0 \}$

Funcion POTENCIA (a, n:entero) retorna (p:entero)

caso

$n = 0 \rightarrow 1$

$n > 0 \rightarrow$ POTENCIA(a,n-1) * a

fcaso

ffunción

$R \equiv \{ p = a^n \}$

Función recursiva simple y no final

$Q \equiv \{ a \geq 0 \wedge n_{inicial} \geq 0 \}$

Funcion POTENCIA (a, $n_{inicial}$:entero) retorna (p:entero)

var n: entero fvar

$n = n_{inicial}$

mientras $n > 0$ hacer

$n = n - 1$

fmientras

$p = 1$

mientras $n \neq n_{inicial}$ hacer

$n = n + 1$

$p = p * a$

fmientras

retorna p

ffunción

$R \equiv \{ p = a^{n_{inicial}} \}$

Función iterativa equivalente

EJEMPLO FUNCIÓN RECURSIVA SIMPLE Y NO FINAL: POTENCIA (mejora)

$Q \equiv \{ a \geq 0 \wedge n \geq 0 \}$

Funcion POTENCIA (a, n:entero) retorna (p:entero)
  caso
    $n = 0 \rightarrow 1$
    $n > 0 \rightarrow$ POTENCIA(a,n-1) * a
  fcaso
ffunción

$R \equiv \{ p = a^n \}$

Función recursiva simple y no final

$Q \equiv \{ a \geq 0 \wedge n_{inicial} \geq 0 \}$

Funcion POTENCIA (a, $n_{inicial}$:entero) retorna (p:entero)
  var n: entero fvar
  ~~$n = n_{inicial}$~~
  ~~mientras n > 0 hacer~~
    ~~$n = n - 1$~~
  ~~fmientras~~
  $p = 1$
  mientras $n \neq n_{inicial}$ hacer
    $n = n + 1$
    $p = p * a$
  fmientras
  retorna p
ffunción

$R \equiv \{ p = a^{n_{inicial}} \}$

n = 0;

Función iterativa equivalente

TRANSFORMACIÓN A ITERATIVO DE UNA RECURSIÓN SIMPLE Y NO FINAL CUANDO NO EXISTE s^{-1}

No siempre existe la inversa de la función s (sucesor).

En ese caso, la implementación de s^{-1} consistirá en recuperar $\overline{x}$ de una estructura de almacenamiento donde, durante el bucle de descenso, se han almacenado los parámetros $\overline{x}$ de todas las llamadas.

EJEMPLO FUNCIÓN RECURSIVA SIMPLE Y NO FINAL CUANDO NO EXISTE s^{-1}

$Q \equiv \{ (a \geq 0) \wedge (b \geq 0) \}$

```
Funcion PRODUCTO (a, b : entero) retorna (p:entero)
   var p´: entero fvar
   si a = 0 entonces retorna 0
   sino
      p´ = PRODUCTO(adiv2, b*2)
      si a % 2 = 0 entonces retorna p´
      sino retorna p´+ b
      fsi
   fsi
ffunción
```

$R \equiv \{ p = a * b \}$

Función recursiva simple y no final

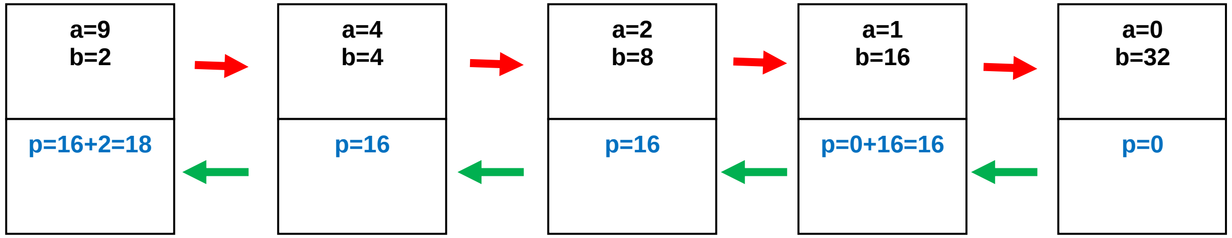
EJEMPLO FUNCIÓN RECURSIVA SIMPLE Y NO FINAL CUANDO NO EXISTE s^{-1}

$Q \equiv \{ (a \geq 0) \wedge (b \geq 0) \}$

```
Funcion PRODUCTO (a, b : entero) retorna (p:entero)
   var p´: entero fvar
   si a = 0 entonces retorna 0
   sino
      p´ = PRODUCTO(adiv2, b*2)
      si a % 2 = 0 entonces retorna p´
      sino retorna p´+ b
      fsi
   fsi
ffunción
```

$R \equiv \{ p = a * b \}$

s(a,b) = (adiv2,b*2)



si a%2=0 entonces p = p’
sino p = p’ + b
fsi

Función recursiva simple y no final

VERSIÓN ITERATIVA DE PRODUCTO

$Q \equiv \{ (a_{inicial} \geq 0) \wedge (b_{inicial} \geq 0) \}$

función PRODUCTO ($a_{inicial}$, $b_{inicial}$: entero) retorna (p: entero)

var a, b, i, p: entero, vector_as[1..$a_{inicial}$]: vector de enteros fvar

i =1
vector_as[i] = $a_{inicial}$
a = $a_{inicial}$
b = $b_{inicial}$

$$\overline{x} = \overline{x}_{inicial}$$

mientras a ≠ 0 hacer
   a = a / 2
   b = b * 2
   i = i + 1
   vector_as[i] = a
fmientras

VERSIÓN ITERATIVA DE PRODUCTO

$Q \equiv \{ (a_{inicial} \geq 0) \wedge (b_{inicial} \geq 0) \}$

función PRODUCTO_iterativo ($a_{inicial}$, $b_{inicial}$: entero) retorna (p: entero)

var a, b, i, p: entero, vector_as[1..$a_{inicial}$]: vector de enteros fvar

i =1

vector_as[i] = $a_{inicial}$ $\qquad \bar{x} = \bar{x}_{inicial}$

a = $a_{inicial}$

b = $b_{inicial}$

mientras a ≠ 0 hacer $\qquad$ **mientras Bnt($\bar{x}$) hacer**

a = a / 2

b = b * 2 $\qquad$ **$\bar{x}$ = s($\bar{x}$)**

i = i + 1

vector_as[i]=a

fmientras $\qquad$ **fmientras**

VERSIÓN ITERATIVA DE PRODUCTO

$p = 0$ → $\bar{y} = triv(\bar{x})$

mientras $a \neq a_{inicial}$ hacer → mientras $\bar{x} \neq \bar{x}_{inicial}$ hacer

  $i = i - 1$

  $a = vector\_as[i]$ → $\bar{x} = s^{-1}(\bar{x})$

  $b = b / 2$

  si $a \% 2 \neq 0$ entonces $p = p + b$ fsi → $\bar{y} = c(\bar{y}, \bar{x})$

fmientras → fmientras

retorna p → retorna $\bar{y}$

ffuncion

$R \equiv \{ p = a_{inicial} * b_{inicial} \}$

VERSIÓN ITERATIVA DE PRODUCTO

p = 0

mientras a ≠ $a_{inicial}$ hacer
  i = i - 1
  a = vector_as[i]
  b = b / 2
  si a % 2 ≠ 0 entonces p = p + b fsi
fmientras

retorna p
ffuncion

R ≡ { p = $a_{inicial}$ * $b_{inicial}$ }

$\bar{y}$ = triv($\bar{x}$)

mientras $\bar{x} \neq \bar{x}_{inicial}$ hacer

  $\bar{x}$ = $s^{-1}(\bar{x})$

  $\bar{y}$ = c ($\bar{y}$, $\bar{x}$)
fmientras

retorna $\bar{y}$

VERSIÓN ITERATIVA DE PRODUCTO

p = 0

mientras a $\neq$ $a_{inicial}$ hacer

  i = i - 1

  a = vector_as[i]

  b = b / 2

  si a % 2 $\neq$ 0 entonces p = p + b fsi

fmientras

retorna p

ffuncion

R ≡ { p = $a_{inicial}$ * $b_{inicial}$ }

$\bar{y}$ = triv($\bar{x}$)

mientras $\bar{x} \neq \bar{x}_{inicial}$ hacer

$\bar{x}$ = $s^{-1}(\bar{x})$

$\bar{y}$ = c ($\bar{y}$, $\bar{x}$)

fmientras

retorna $\bar{y}$

TRANSFORMACIÓN A ITERATIVO DE UNA RECURSIÓN SIMPLE Y FINAL

- La transformación da lugar a **un solo bucle**.
- La variable $\bar{x}$ toma sucesivamente el valor de los parámetros de cada llamada recursiva. La versión iterativa solo necesita espacio para una copia de los mismos.
- A continuación se muestra el esquema general de una función recursiva final y de su versión iterativa.

TRANSFORMACIÓN A ITERATIVO DE UNA RECURSIÓN SIMPLE Y FINAL

{ Q($\bar{x}$) }

función f($\bar{x}$: T_1) retorna ($\bar{y}$: T_2)

caso

Bt ($\bar{x}$) $\rightarrow$ triv($\bar{x}$)

Bnt ($\bar{x}$) $\rightarrow$ f(s($\bar{x}$))

fcaso

ffunción

{ R($\bar{x}$,$\bar{y}$) }

Función recursiva simple y final

{ Q($\bar{x}$) }

función f($\bar{x}$: T_1) retorna ($\bar{y}$: T_2)

  caso

    Bt ($\bar{x}$) $\rightarrow$ triv($\bar{x}$)

    Bnt ($\bar{x}$) $\rightarrow$ f(s($\bar{x}$))

  fcaso

ffunción

{ R($\bar{x}$,$\bar{y}$) }

Función recursiva simple y final

{ Q($\bar{x}_{inicial}$) }

función f($\bar{x}_{inicial}$: T_1) retorna ($\bar{y}$: T_2)

  var $\bar{x}$: T_1 fvar

  $\bar{x} = \bar{x}_{inicial}$

  mientras Bnt($\bar{x}$) hacer

    $\bar{x}$ = s($\bar{x}$)

  fmientras

  retorna triv($\bar{x}$)

ffunción

{ R($\bar{x}_{inicial}$, $\bar{y}$) }

Función iterativa equivalente

EJEMPLO FUNCIÓN RECURSIVA SIMPLE Y FINAL: TABLEROS

$Q \equiv \{ (n \geq m) \wedge (n > 0) \wedge (m > 0) \}$

Funcion TABLEROS (n, m : entero) retorna (h:entero)

caso

$m = 1 \rightarrow n^2$

$m > 1 \rightarrow$ TABLEROS(n-1, m-1)

fcaso

ffunción

$R \equiv \{ h = (n - m + 1)^2 \}$

Función recursiva simple y final

$Q \equiv \{ (n_{inicial} \geq m_{inicial}) \wedge (n_{inicial} > 0) \wedge (m_{inicial} > 0) \}$

Funcion TABLEROS ($n_{inicial}$, $m_{inicial}$: entero) retorna (h:entero)

var n, m: entero fvar

$n = n_{inicial}$

$m = m_{inicial}$

mientras $m > 1$ hacer

$n = n - 1$

$m = m - 1$

fmientras

retorna n^2

ffunción

$R \equiv \{ h = (n_{inicial} - m_{inicial} + 1)^2 \}$

Función iterativa equivalente

TAREAS PARA EL ALUMNO

Realizar las tareas en el fichero fuente entregado en la sesión de prácticas anterior:

- Implementar las transformaciones a iterativo de las siguientes funciones:
  - Función POTENCIA (la función iterativa figura en la diapositiva 8 de este documento).
  - Función MCD (transformación a iterativo a realizar por el alumno)

  comprobar que el resultado ofrecido por la versión iterativa coincide con el de la versión recursiva

- Implementar la función recursiva Tableros y su correspondiente versión iterativa. Ambas funciones están en la diapositiva 20 del presente documento. Comprobar que el resultado ofrecido por la versión iterativa coincide con el de la versión recursiva

TAREAS PARA EL ALUMNO

- Implementar las funciones recursivas: suma elementos de un vector (diapositivas 67 y 72), capicúa (diapositiva 77) y simetría de una matriz (diapositivas 83 y 84).

- Implementar sus correspondientes versiones iterativas. Comprobar que el resultado ofrecido por la versión iterativa coincide con el de la versión recursiva

- Al finalizar la sesión de prácticas, entregar a través del Campus Virtual el fichero fuente que recoja las tareas realizadas. El nombre de dicho fichero debe contener el nombre y apellidos del alumno.