Lenguajes de programación

Oscar Eduardo Galaviz Cuen

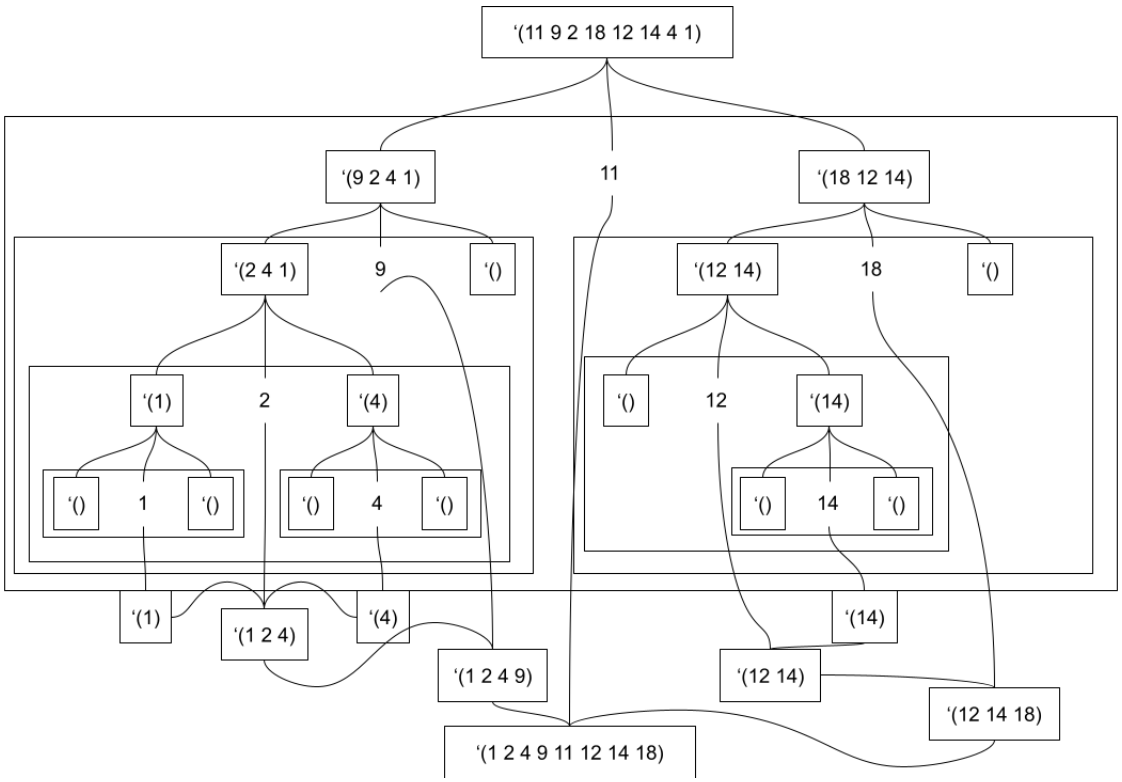
7 de Septiembre de 2022

1 Problema 5: La llamada (bundle '("a" "b" "c") 0) es un buen uso de bundle? ¿qué produce? ¿por qué?

No es un buén uso de bundle, este técnicamente no produce nada ya que se cicla, esto debido a que (drop '("a" "b" "c") 0) regresa la misma lista y, al usar recursividad en (bundle (drop '("a" "b" "c") 0) 0), provoca que esta función se cicle.

```
(define (bundle s n)
  (cond
    [(null? s) null]
    [else
     (if (unit-string-list? s)
         (cons (implode (take s n))
          (bundle (drop s n) n))
         (list->chunks s n))]))
```

2 Problema 9: Dibuja un diagrama como el de la figura anterior pero para la lista '(11 9 2 18 12 14 4 1).



3 Problema 11: Si la entrada a quicksort contiene varias repeticiones de un número, va a regresar una lista estrictamente más corta que la entrada. Responde el por qué y arregla el problema.

Esto sucede debido a que, cuando el pivote es un número que está repetido en la lista, los comparadores comprueban si el número es estrictamente mayor o menor al pivote, lo que hace que se ignore al ser igual. Mi manera de arreglar esto fue concatenando una tercera lista, la cual contiene a todos los elementos de la lista que sean

iguales al pivote.

```
;;quicksortfix : (listof number?) -> (listof number?)
(define (quicksortfix ls)
  (cond
    [(empty? ls) null]
    [else
     (define pivot (first ls))
     (append (quicksortfix (smallers ls pivot))
             (equallers ls pivot)
             (quicksortfix (largers ls pivot)))]))
```

4 Problema 13: Implementa una versión de quicksort que utilice isort si la longitud de la entrada está por debajo de un umbral. Determina este umbral utilizando la función time, escribe el procedimiento que seguiste para encontrar este umbral.

Lo primero que hice en este caso es definir una función la cual toma como primer argumento al número de elementos que tendrá la lista y como segundo argumento el mayor número que alcanza esta lista.

```
;función para crear una lista de numeros aleatorios para saber el umbral
(define (randomlist n mx)
  (for/list ((i n))
    (add1 (random mx))))
```

Lo que hice fue que, una vez generada la lista con n elementos, llamé la función time y puse como argumento a quicksortfix, el cual, tiene como argumento la lista generada, después utilicé el mismo proceso, pero ahora utiizando la función isort.
Si llega el caso en el que los tiempos son iguales, aumentaba el número de elementos de la lista hasta que fueran distintos, de esta forma, llegué a la conclusión de que, si la lista no supera los 100 elementos, es mejor usar isort, en caso contrario es mejor usar quicksortfix.

5 Problema 18: Considera la siguiente definición de smallers, uno de los procedimientos utilizados en quicksort, responde en qué puede fallar al utilizar esta versión modificada en el procedimiento de ordenamiento.

Se cicla, veamos con este caso:
(define (smallers '(2 1 1) 2))

Hagamoslo paso a paso, veamos que pasa.
'(2 1 1)
(empty? '(2 1 1))
(<=? 2 2)
(cons 2 (smallers '(1 1) 2))
(cons 2 ((emtpy? '(1 1))))
(cons 2 ((<=? 2 1)))
(cons 2 (cons 1 (smallers '(1) 2))
(cons 2 (cons 1 ((empty? '(1)))))
(cons 2 (cons 1 ((<=? 2 1)))
(cons 2 (cons 1 (cons 1 (smallers '() 2)))
(cons 2 (cons 1 (cons 1 ((empty? '()))))))
(cons 2 (cons 1 (cons 1 '())))
(cons 2 (cons 1 '(1)))
(cons 2 '(1 1))
'(2 1 1)
Entonces, como podemos observar, da el mismo resultado.