TP2 (recuperatorio) EDyAII

Facundo Emmanuel Messulam
Franco Ignacio Vallejos Vigier

Junio 2021

1 Introduction

Se presentan los costos obtenidos, para trabajo y profundidad, y sus respectivas demostraciones para las operaciones del TAD de secuencias implementados listas y con vectores, en Haskell.

2 Implementación con listas

El detalle más importante para los costos demostrados abajo es que el acceso del enésimo elemento es $O(n)$.

2.1 mapS

```
mapS f [] = []
mapS f (x:xs) = let (r, rs) = (f x) ||| mapS f xs
                in r:rs
```

Valga aclarar que a pesar de que `mapS` toma una función y una secuencia, nosotros definimos $W_{mapS}(f, n)$ y $S_{mapS}(f, n)$ con n el largo de la secuencia (y no la secuencia en si), esto es por conveniencia de notación en las demostraciones que se muestran debajo.

2.1.1 Trabajo

Calcularemos el trabajo en base al largo de la lista ingresada y sea s_i el i-esimo elemento de la lista, planteamos a partir del código:

$$W_{mapS}(f, 0) = c_{vacio}$$
$$W_{mapS}(f, n) = W_f(s_0) + W_{mapS}(f, n-1) + k_1$$

Después de realizar n interaciones, el trabajo adquiere la forma:

$$W_{mapS}(f, n) = \sum_{i=0}^{n-1} W_f(s_i) + \sum_{i=0}^{n-1} k_1$$

Al ver esto vamos a plantear como hipótesis inductiva lo siguiente:
HI)

$$W_{mapS}(f, n) \leq c_1 * \sum_{i=0}^{n-1} W_f(s_i) + k$$

Para llegar a demostrar lo siguiente:

$$W_{mapS}(f, n+1) \leq c_1 * \sum_{i=0}^{n} W_f(s_i) + k$$

Demostramos por inducción:
Casos base) Sea n = 0:

$$W_{mapS}(f, 0) = c_{vacio} \leq c_1$$

con $c_{vacio} \leq c_1$
Paso inductivo)

$$W_{mapS}(f, n+1) = W_f(s_0) + W_{mapS}(f, n) + k$$

Por Hipótesis Inductiva

$$W_{mapS}(f, n+1) \leq W_f(s_0) + c_1 * \sum_{i=1}^{n} W_f(s_i) + k$$

Siendo c_1 un factor mayor o igual a $\max(c_{vacio}, 1)$:

$$W_{mapS}(f, n+1) \leq c_1 * W_f(s_0) + c_1 * \sum_{i=1}^{n} W_f(s_i) + k$$

Factor común:

$$W_{mapS}(f, n+1) \leq c_1 * (W_f(s_0) + \sum_{i=1}^{n} W_f(s_i)) + k$$

Introduciendo el sumando a la sumatoria:

$$W_{mapS}(f, n+1) \leq c_1 * (\sum_{i=0}^{n} W_f(s_i)) + k$$

Llegando a lo que se quería demostrar, con c_1 mayor o igual a 1.
Por lo tanto:

$$W_{mapS}(f, n) \in O(\sum_{i=0}^{n-1} W_f(s_i))$$

2.1.2 Profundidad

$$S_{mapS}(f, 0) = c_1$$

$$S_{mapS}(f, n) = \max(S_f(s_0), S_{mapS}(f, n-1)) + c_1$$

$$S_{mapS}(f, n) = \max(S_f(s_0), c_1 + \max(S_f(s_1), S_{mapS}(f, n-2))) + c_1$$

$$S_{mapS}(f, n) = c_1 + \max(S_f(s_0), c_1 + \max(S_f(s_1), c_1 + \max(...(c_1 + \max(S_f(s_{(n-1)}), c_1))...)))$$

Por lo anterior vemos que:

$$S_{mapS}(f, n) \leq c_1 * n + \max_{i=0}^{n-1} S_f(s_i)$$

Entonces demostramos esto por inducción:
Caso base) $n = 0$:

$$S_{mapS}(f, 0) = c_{vacio} \leq c_1$$

con $c_{vacio} \leq c_1$
Planteamos la hipotesis inductiva)

$$S_{mapS}(f, n) \leq c_1 * n + \max_{i=0}^{n-1} S_f(s_i)$$

Paso Inductivo)
Queremos demostrar: $S_{mapS}(f, n+1) \leq c_1 * (n+1) + \max_{i=0}^{n}(S_f(s_i))$.
Tengo:

$$S_{mapS}(f, n+1) = c_1 + max(S_f(s_0), S_{mapS}(f, n))$$

Por hipotesis inductiva:

$$S_{mapS}(f, n+1) = c_1 + max(S_f(s_0), c_1 * n + \max_{i=1}^{n-1}(S_f(s_i)))$$

$$S_{mapS}(f, n+1) \leq c_1 * (n+1) + \max_{i=0}^{(n+1)-1}(S_f(s_i)))$$

Para entender el paso anterior se puede pensar que no importa si $S_f(s_0)$ es máximo o $c_1 * n + \max_{i=1}^{n-1}(S_f(s_i))$ es máximo, el nuevo termino $c_1 * (n+1) + \max_{i=0}^{(n+1)-1}(S_f(s_i))$ nos da un resultado que es mayor en cualquiera de los dos casos.
Quedando demostrada la hipótesis inductiva.
Por lo tanto:

$$S_{mapS}(f, n) \in O(n + max_{i=0}^{n-1} S_f(s_i))$$

3 Implementación con vectores

El detalle más importante para los costos demostrados abajo es que el acceso del enésimo elemento es $O(1)$, también crear subarreglos es $O(1)$.

3.1 contraccion

Esta es una funcion auxiliar, se demuestra su costo en esta sección para luego usarlo en la subsección de `scanS`.

```
contraccionAux :: (a -> a -> a) -> A.Arr a -> Int -> a
contraccionAux f s i | (lengthS s) - i == 0 = undefined
                     | (lengthS s) - i == 1 = let x = nthS s i
                                              in x
                     | otherwise            = let
                                                 x = nthS s i
                                                 y = nthS s (i+1)
                                              in f x y

contraccion :: (a -> a -> a) -> A.Arr a -> A.Arr a
contraccion f s = let
                     l = lengthS s
                     rlen =  if even l then (div l 2) else ((div l 2) +1)
                  in tabulateS (\i -> contraccionAux f s (i*2)) rlen
```

3.1.1 Trabajo

Primero que nada notese que `contraccionAux` no es recursiva. El peor de los casos es cuando $(lengthS\ s) - i > 1$, en estos casos se ejecutan dos `nthS` de costo (para trabajo) O(1) y una llamada a `f` de trabajo O(1) (el costo de `f` en este caso esta dado por la cátedra), formalmente:

$$\begin{aligned} W_{contraccionAux}(f, s, i) &= W_{nthS}(s, i) + W_{nthS}(s, i+1) + W_f(s_i, s_{(i+1)}) + c \\ &= c_2 \\ &\in O(1) \end{aligned}$$

Donde:

- `f` es la función a aplicar
- `s` es la secuencia a contraer
- `i` es el índice sobre el que contraer, nótese que la función es aplicada a s_i y $s_{(i+1)}$ en el caso en que se pueda

Luego es fácil ver que `contraccion` tampoco es recursiva, entonces el costo esta dado por:

$$\begin{aligned} W_{contraccion}(f, s) &= W_{lengthS}(s) + W_{tabulateS}(contraccionAux, |s|/2 + 1) + c_1 \\ &= W_{tabulateS}(contraccionAux, |s|/2 + 1) + c_2 \\ &= \sum_{i=0}^{|s|/2+1} W_{contraccionAux}(f, s, i * 2) + c_3 \\ &\in O(|s|) \end{aligned}$$

Notar que `lengthS` es de trabajo constante por ser vectores (costo dado por la catedra) y `tabulateS` es de trabajo lineal por ser vectores (costo dado por la catedra).
Donde:

- `f` es la función a aplicar
- `s` es la secuencia a contraer

3.1.2 Profundidad

El análisis de profundidad es análogo para `contraccionAux`, $S_{contraccionAux}() \in O(1)$.
Sin embargo, como `tabulateS` tiene costo dado por la cátedra $O(1)$ en profundidad, para `contraccion` el costo disminuye a O(1) en el caso de profundidad:

$$\begin{aligned} S_{contraccion}(f, s) &= S_{lengthS}(s) + S_{tabulateS}(contraccionAux, |s|/2 + 1) + c_1 \\ &= S_{tabulateS}(contraccionAux, |s|/2 + 1) + c_2 \\ &= c_3 \\ &\in O(1) \end{aligned}$$

Donde:

- `f` es la función a aplicar
- `s` es la secuencia a contraer

3.2 scanS

```
scanS f b s | lengthS s == 0 = (emptyS, b)
            | lengthS s == 1 = let x = nthS s 0
                               in (singletonS b, f b x)
            | otherwise      = let
                                    l = lengthS s
                                    contracted = (contraccion f s)
                                    (s', last) = scanS f b contracted

                                    r i | even i    = nthS s' (div i 2)
                                        | otherwise = let
                                                        a = nthS s' (div i 2)
                                                        b = nthS s (i-1)
                                                      in f a b -- se asume f \in O(1)

                                in (tabulateS (r) l, last)
```

Por los costos dados por la cátedra podemos asegurar que los costos de `scanS` con una secuencia de largo 0 o 1 son constantes tanto en trabajo como profundidad:
$n = |s| = 0$:

$$W_{scanS}(f, b, s) = c_1$$
$$S_{scanS}(f, b, s) = c_2$$

$n = |s| = 1$:

$$W_{scanS}(f, b, s) = W_{nthS}(s, 0) + W_{singletonS}(b) + W_f(b, s_0) + c_3$$
$$S_{scanS}(f, b, s) = S_{nthS}(s, 0) + S_{singletonS}(b) + S_f(b, s_0) + c_4$$

Dado que los costos de `nthS` y `singletonS` son constantes (tanto en trabajo como en profundidad) y se asume que la función `f` tiene costo constante (tanto en trabajo como en profundidad), entonces:

$$W_{scanS}(f, b, s) \in O(1)$$
$$S_{scanS}(f, b, s) \in O(1)$$

Otra aclaración preambular es que la función:

```
r i | even i    = nthS s' (div i 2)
    | otherwise = let
                    a = nthS s' (div i 2)
                    b = nthS s (i-1)
                  in f a b -- se asume f \in O(1)
```

Que se puede ver en el `scanS` es una función no recursiva, que solo llama a funciones de costo en trabajo y profundidad que son constantes, esto es porque `nthS` y `f` son de costo constante. Lo anterior hace que la función `r` sea $W_r(i) \in O(1)$ y $S_r(i) \in O(1)$.
Demostramos ahora los costos para secuencias más largas.

3.2.1 Trabajo

Tomo una secuencia s de largo $|s| = n$.
Para un $n > 1$ cualquiera:

$$W_{scanS}(f, b, s) = W_{scanS}(f, b, s') + W_{contraccion}(f, s) + W_{tabulateS}(r, s) + c_1$$

Donde:

- s' tiene un largo de $|s|/2 + 1$ y se obtiene como resultado de aplicar `contraccion` sobre s
- c_1 es el trabajo de las operaciones adicionales (como `r`, `even`, `nthS`, y `div`) que se necesitan hacer para cada recursion de `scanS`, notar que todas estas tienen costo constante en trabajo.

Tomo

$$g(f, s, r) = W_{contraccion}(f, s) + W_{tabulate}(r, s) + c_1$$

Pasando en limpio la recurrencia:

$$W_{scanS}(f, b, s) \leq W_{scanS}(f, b, s') + g(f, s, r)$$

Para calcular el costo en trabajo de `g` podemos usa el hecho de que $W_{contraccion}(f, s) \in O(|s|)$ (probado antes) y $W_{tabulateS}(f, s) \in O(|s|)$ dado por la cátedra:

$$W_g(f, s, r) \in O(|s|)$$

Aplicando el Teorema Maestro (caso tres):

$$W_{scanS}(f, b, s) \in \Theta(W_g(f, s, r))$$

Entonces:

$$W_{scanS}(f, b, s) \in O(|s|)$$

$$W_{scanS} \in O\left(\sum_{(f\ x\ y) \in \mathcal{O}(f,b,s)} W_f(x, y)\right)$$

Quedando demostrado el costo requerido por la cátedra.

3.2.2 Profundidad

Para calcular la profundidad, la planteamos de la siguiente manera:

$$S_{scanS}(f, b, s) = S_{scanS}(f, b, s') + S_{contraccion}(f, s) + S_{tabulateS}(r, s) + c_1$$

Donde:

- s tiene un largo de $|s|$ y es la secuencia sobre la que se aplica el scanS.
- s' tiene un largo de $|s|/2 + 1$ y se obtiene como resultado de aplicar `contraccion` sobre s
- c_1 es la profundidad de las operaciones adicionales (como `r`, `even`, `nthS`, y `div`) que se necesitan hacer para cada recursion de `scanS`, notar que todas estas tienen costo constante en profundidad.

Como la cátedra provee un costo en profundidad para `tabulate` que es $S_{tabulate}(r, s) \in O(\max_{x \in s} S_r(x))$ y `r` es $S_r(i) \in O(1)$, podemos decir que $S_{tabulateS}(r, s) \in O(1)$.

Como ya probamos que $S_{contraccion}(f, s) \in O(1)$, podemos tomar una nueva constante que tenga sea la suma de estos costos y los de c_1:

$$S_{scanS}(f, b, s) \leq S_{scanS}\,(f, b, s') + c_2$$

Como $|s'| = |s|/2 + 1$, por el Teorema Maestro (caso dos) puedo decir que:

$$S_{scanS}\,(f, b, s') \in \Theta(\lg |s|)$$

Como está acotada $S_{scanS}(f, b, s)$ por $\lg |s|$, concluimos que:

$$S_{scanS} \in O\left(\lg |s| \max_{(f\ x\ y) \in \mathcal{O}(f,b,s)} S_f(x, y)\right)$$

Quedando demostrado el costo requerido por la cátedra.