

# Trabajo Práctico 2 EDyAII

## Análisis de Costos

Farizano, Juan Ignacio

Mellino, Natalia

## 1. Implementación de Secuencias con Listas

### 1.1. Función mapS

Sea  $xs = [x_{|xs|-1}, \dots, x_0]$  una lista,  $n = |xs|$  su longitud y sea  $f$  la función que toma `mapS` como argumento.

#### 1.1.1. Trabajo:

La recurrencia para el trabajo la podemos expresar de la siguiente manera:

$$W_{mapS}(n) = W_f(x_{n-1}) + W_{mapS}(n-1) + k$$

Donde  $k$  es una constante, y  $x$  es la cabeza de la lista.

Demostramos por inducción que  $W_{mapS} \in O\left(\sum_{i=0}^{n-1} W_f(x_i) - k\right)$

$$\begin{aligned} W_{mapS}(n) &= W_f(x_{n-1}) + W_{mapS}(n-1) + k \\ &\leq W_f(x_{n-1}) + c \cdot \left(\sum_{i=0}^{n-2} W_f(x_i) - k\right) + k \rightarrow \text{HI} \\ &\leq c \cdot W_f(x_{n-1}) + c \cdot \sum_{i=0}^{n-2} W_f(x_i) - c \cdot k + k \\ &\leq c \cdot \sum_{i=0}^{n-1} W_f(x_i) \iff c \geq 1 \end{aligned}$$

Por lo tanto como  $W_{mapS} \in O\left(\sum_{i=0}^{n-1} W_f(x_i) - k\right)$ , luego podemos decir que  $W_{mapS} \in O\left(\sum_{i=0}^{n-1} W_f(x_i)\right)$

#### 1.1.2. Profundidad:

Para la profundidad, la recurrencia la podemos expresar de la siguiente manera:

$$S_{mapS}(n) = \max(S_f(x_{n-1}), S_{mapS}(n-1)) + k$$

Podemos demostrar también que  $S_{mapS} \in O\left(\sum_{i=0}^{n-1} S_f(x_i)\right)$ :

$$\begin{aligned} S_{mapS}(n) &= \max(S_f(x_{n-1}), S_{mapS}(n-1)) + k \\ &\leq S_f(x_{n-1}) + S_{mapS}(n-1) + k \end{aligned}$$

Si observamos la ecuación anterior, podemos ver que se obtuvo una similar a la del trabajo, por lo tanto, el análisis de la profundidad se realiza de la misma manera que en el trabajo. Entonces podemos concluir que  $S_{mapS} \in O\left(\sum_{i=0}^{n-1} S_f(x_i)\right)$ .

## 1.2. Función appendS

Sean  $n$  la longitud de la primer lista que recibe como argumento la función **appendS**. Observemos, que tanto el trabajo como la profundidad se realizan con respecto  $n$  ya que es la lista que vamos consumiendo en cada llamada recursiva.

### 1.2.1. Trabajo:

Podemos ver que la recurrencia para el trabajo nos queda expresada como:

$$W_{appendS}(n) = W_{appendS}(n-1) + k$$

Donde  $k$  es una constante.

Podemos demostrar fácilmente por inducción que  $W \in O(n)$ :

$$\begin{aligned} W_{appendS}(n) &= W_{appendS}(n-1) + k \\ &\leq c(n-1) + k \rightarrow \text{HI} \\ &\leq c \cdot n - c + k \\ &\leq c \cdot n \iff c \geq k \end{aligned}$$

Por lo tanto  $W_{appendS} \in O(n)$

### 1.2.2. Profundidad:

Para la profundidad, la recurrencia nos queda expresada igual que la del trabajo:

$$S_{appendS}(n) = S_{appendS}(n-1) + k \in O(n)$$

Es decir, tanto el trabajo como la profundidad de la función **appendS** son del orden de la suma de la longitud de ambas listas.

## 1.3. Función reduceS

Sea  $n$  la longitud de la lista que **reduceS** recibe como argumento.

### 1.3.1. Trabajo:

La recurrencia para **reduceS** la podemos expresar de la siguiente manera (recordemos que se asume que la función que recibe como argumento es de orden constante):

$$W_{reduceS}(n) = W(\frac{n}{2}) + W_{contract}(n) + k$$

Ahora necesitamos saber que orden tiene  $W_{contract}(n)$ , observemos que su recurrencia es de la forma:

$$W_{contract}(n) = W_{contract}(n-2) + k$$

Podemos demostrar que  $W_{contract} \in O(n)$ :

$$\begin{aligned} W_{contract}(n) &= W_{contract}(n-2) + k \\ &\leq c(n-2) + k \rightarrow \text{HI} \\ &\leq cn - 2c + k \\ &\leq cn \iff c \geq \frac{k}{2} \end{aligned}$$

Ahora, utilizando el tercer caso del **Teorema Maestro** podemos probar que  $W_{reduceS}(n) \in O(n)$ , debemos ver dos cosas:

Sean  $a = 1$ ,  $b = 2$  y  $f(n) = W_{contract}(n)$

- Existe  $\epsilon > 0$  tal que  $f(n) \in \Omega(n^{lg_2 1 + \epsilon})$ : de hecho, como  $f(n)$  es  $O(n)$  basta tomar  $\epsilon = 1$  y trivialmente se satisface la condición
- Existe  $c < 1$  y  $N \in \mathbb{N}$  tal que para todo  $n > N$ ,  $1 \cdot f(\frac{n}{2}) \leq c \cdot f(n)$ : nuevamente, como  $f(n)$  es  $O(n)$ , podemos tomar  $c = \frac{1}{2}$  y  $N = 1$  y se cumple:  $\frac{n}{2} \leq \frac{1}{2}n$ .

Entonces, como se cumplen las hipótesis del caso mencionado, podemos decir que  $W_{reduceS} \in O(f(n))$  y como  $f(n) \in O(n)$ , por transitividad, resulta  $W_{reduceS} \in O(n)$ .

### 1.3.2. Profundidad:

Para la profundidad tenemos la siguiente recurrencia:

$$S_{reduceS}(n) = \max(S_{contract}(n), S_{reduceS}(\frac{n}{2})) + k$$

Podemos ver que también,  $S \in O(n)$ :

$$\begin{aligned} S_{reduceS}(n) &= \max(S_{contract}(n), S_{reduceS}(\frac{n}{2})) + k \\ &\leq S_{contract}(n) + S_{reduceS}(\frac{n}{2}) + k \end{aligned}$$

Observemos que ahora la recurrencia nos quedo expresada de manera similar a la del trabajo, por lo tanto, viendo el análisis anterior podemos concluir que  $S_{reduceS} \in O(n)$ .

## 1.4. Función scanS

Sea  $n$  la longitud de la lista que recibe **scanS** como argumento, y asumiendo que la función  $f$  que toma es de orden constante, tenemos las siguientes recurrencias:

### 1.4.1. Trabajo:

Para el trabajo se tiene que:

$$W_{scanS}(n) = W_{scanS}\left(\frac{n}{2}\right) + W_{contract}(n) + W_{evenExpand}(n) + k$$

Donde  $k$  es una constante. Como vimos anteriormente en **reduceS**, tenemos que  $W_{contract}(n) \in O(n)$ . Podemos ver que pasa lo mismo para la función **evenExpand**. Sea  $n$  lo suficientemente grande, entonces:

$$W_{evenExpand}(n) = W_{oddExpand}(n-1) + k = W_{evenExpand}(n-2) + k'$$

Donde es trivial que viendo esta última igualdad,  $W_{evenExpand}(n) \in O(n)$ . Luego, tenemos que:

$$\begin{aligned} W_{scanS}(n) &= W_{scanS}\left(\frac{n}{2}\right) + W_{contract}(n) + W_{evenExpand}(n) + k \\ &\leq W_{scanS}\left(\frac{n}{2}\right) + cn + c'n + k \\ &= W_{scanS}\left(\frac{n}{2}\right) + (c + c')n + k \end{aligned}$$

Ahora observemos que esta recurrencia es similar a la obtenida para el trabajo en **reduceS**, entonces utilizando el tercer caso del **Teorema Maestro** tomando  $f(n) = (c + c')n$ ,  $a = 1$  y  $b = 2$ , podemos concluir que  $W_{scanS}(n) \in O(n)$ .

### 1.4.2. Profundidad:

La recurrencia para la profundidad nos queda expresada de la siguiente manera:

$$S_{scanS}(n) = S_{scanS}\left(\frac{n}{2}\right) + S_{contract}(n) + S_{evenExpand}(n) + k$$

Sabemos que  $S_{contract}(n) \in O(n)$  y podemos ver fácilmente que  $S_{evenExpand} \in O(n)$ :

$$S_{evenExpand}(n) = S_{oddExpand}(n-1) + k = S_{evenExpand}(n-2) + k'$$

Entonces, análogamente a la recurrencia del trabajo resuelta anteriormente se tiene que  $S_{evenExpand}(n) \in O(n)$ .

Veamos que la recurrencia  $S_{scanS}$  nos quedó igual a la del trabajo, entonces el razonamiento es análogo y podemos concluir que  $S_{scanS} \in O(n)$ .

## 2. Implementación de Secuencias con Arreglos

Por especificación tenemos que

$$\begin{aligned} W_{tabulate}(f \ n) &\in O\left(\sum_{i=0}^{n-1} W_f(i)\right) \\ S_{tabulate}(f \ n) &\in O\left(\max_{i=0}^{n-1} S_f(i)\right) \end{aligned}$$

### 2.1. Función mapS

Sea  $f$  la función que se recibe como argumento, y  $n$  la longitud del arreglo sobre el cual se evaluará  $f$  sobre sus elementos.

### 2.1.1. Trabajo:

La recurrencia para el trabajo de la función **mapS** la podemos expresar como sigue:

$$W_{mapS}(f\ n) = W_{tabulate}(f\ n) + \underbrace{W_{length}(n)}_{\in O(1)} + k$$

Por lo tanto  $W_{mapS}(f\ n) \in O\left(\sum_{i=0}^{n-1} W_f(i)\right)$

### 2.1.2. Profundidad:

Luego, para la profundidad tenemos la siguiente recurrencia:

$$S_{mapS}(f\ n) = S_{tabulate}(f\ n) + \underbrace{S_{length}(n)}_{\in O(1)} + k$$

Por lo tanto  $S_{mapS}(f\ n) \in O\left(\max_{i=0}^{n-1} S_f(i)\right)$

## 2.2. Función **appendS**

Sea  $n$  y  $m$  la longitudes de los arreglos que reciben **appendS** como argumento.

### 2.2.1. Trabajo:

Podemos expresar la recurrencia para el trabajo de **appendS** de la siguiente manera:

$$W_{appendS}(n + m) = W_{tabulate}(f\ (n + m)) + \underbrace{W_{length}(n)}_{\in O(1)} + \underbrace{W_{length}(m)}_{\in O(1)} + k$$

Podemos ver fácilmente que la función que recibe **tabulateS** como argumento, es  $O(1)$  ya que simplemente toma un índice y devuelve el elemento de la lista que corresponde. Por lo tanto:

$$W_{appendS}(n + m) \in O\left(\sum_{i=0}^{n+m-1} W_f(i)\right) \Rightarrow W_{appendS}(n + m) \in O(n + m)$$

### 2.2.2. Profundidad:

Podemos expresar la recurrencia para la profundidad de **appendS** de la siguiente manera:

$$S_{appendS}(n + m) = S_{tabulate}(f\ (n + m)) + \underbrace{S_{length}(n)}_{\in O(1)} + \underbrace{S_{length}(m)}_{\in O(1)} + k$$

Igualmente al trabajo  $f \in O(1)$ , por lo tanto:

$$S_{appendS}(n + m) \in O\left(\max_{i=0}^{n+m-1} S_f(i)\right) \Rightarrow S_{appendS}(n + m) \in O(1)$$

## 2.3. Función **reduceS**

Sea  $n$  la longitud de la lista que recibe **reduceS** como argumento, y  $f$  una función de orden constante que toma la función como argumento.

### 2.3.1. Trabajo:

La recurrencia para el trabajo de **reduceS** la podemos expresar de la siguiente manera:

$$W_{reduceS}(n) = W_{reduceS}\left(\frac{n}{2}\right) + W_{contract}(n) + \underbrace{W_{length}(n) + k}_{\in O(1)}$$

Primero debemos saber que orden tiene la función **contract**, su recurrencia tiene la forma:

$$W_{contract}(n) = W_{tabulateS}\left(f \frac{n}{2}\right) + \underbrace{W_{length}(n) + k}_{\in O(1)}$$

Por lo tanto,  $W_{contract}(n) \in O\left(\sum_{i=0}^{\frac{n}{2}} W_f(i)\right)$  y como  $f \in O(1)$  resulta  $W_{contract}(n) \in O(n)$

Ahora utilizando el tercer caso del **Teorema Maestro**, podemos demostrar de forma análoga a la función equivalente en listas que  $W_{contract}(n) \in O(n)$ .

### 2.3.2. Profundidad:

La recurrencia para el trabajo de **reduceS** podemos expresarla de la siguiente manera:

$$S_{reduceS}(n) = S_{reduceS}\left(\frac{n}{2}\right) + S_{contract}(n) + \underbrace{W_{length}(n) + k}_{\in O(1)}$$

$$S_{contract}(n) = S_{tabulateS}\left(f \frac{n}{2}\right) + \underbrace{S_{length}(n) + k}_{\in O(1)}$$

$S_{contract}(n) \in O\left(\max_{i=0}^{\frac{n}{2}} S_f(i)\right)$  y como  $f \in O(1)$  resulta  $S_{contract}(n) \in O(1)$

Sabiendo esto, podemos ver que:

$$S_{reduceS}(n) = S_{reduceS}\left(\frac{n}{2}\right) + \underbrace{S_{contract}(n)}_{\in O(1)} + \underbrace{S_{length}(n) + k}_{\in O(1)} = S_{reduceS}\left(\frac{n}{2}\right) + k'$$

Demostramos por inducción que  $S_{reduceS}(n) \in O(\log n)$

$$\begin{aligned} S_{reduceS}(n) &= S_{reduceS}\left(\frac{n}{2}\right) + k' \\ &\leq c \cdot \log \frac{n}{2} + k' \rightarrow HI \\ &= c \cdot \log n - c \cdot \underbrace{\log 2}_{=1} + k' \\ &= c \cdot \log n - c + k' \\ &\leq c \cdot \log n \iff c \geq k' \end{aligned}$$

Por lo tanto,  $S_{reduceS}(n) \in O(\log n)$ .

## 2.4. Función scanS

Sea  $n$  la longitud de la lista que recibe **scanS** como argumento. El análisis de costos para esta función será parecido al de **reduceS**. Veamos cómo nos quedan las recurrencias:

### 2.4.1. Trabajo:

Para el trabajo la recurrencia nos queda expresada de la siguiente manera:

$$W_{scan}(n) = W_{scan}\left(\frac{n}{2}\right) + W_{contract}(n) + W_{expand}(n) + k$$

Donde  $k$  es una constante. Ahora, razonando de forma análoga a **reduceS** sabemos que  $W_{contract}(n) \in O(n)$ . Luego, es fácil de ver que  $W_{expand}(n) \in O(n)$ . Veamos cómo es su recurrencia:

$$W_{expand}(n) = W_{tabulateS}(f\ n) + k$$

Como la función que se le pasa como argumento a **tabulateS** es de orden constante, sabiendo ya el costo de esta podemos decir que  $W_{expand}(n) \in O(n)$ .

Entonces, sabiendo todo esto se tiene:

$$\begin{aligned} W_{scan}(n) &= W_{scan}\left(\frac{n}{2}\right) + W_{contract}(n) + W_{expand}(n) + k \\ &\leq W_{scan}\left(\frac{n}{2}\right) + cn + c'n + k \\ &= W_{scan}\left(\frac{n}{2}\right) + (c + c')n + k \end{aligned}$$

Entonces, usando el tercer caso del **Teorema Maestro** tomando  $f(n) = (c + c') * n$ ,  $a = 1$ ,  $b = 2$ , estamos nuevamente en un caso similar a la función **reduceS** y podemos concluir que  $W_{scanS} \in O(n)$ . Observemos que para **scanS** también asumimos que la función que toma como argumento es de orden constante.

### 2.4.2. Profundidad:

Para la profundidad tenemos la siguiente recurrencia:

$$S_{scanS} = S_{scanS}\left(\frac{n}{2}\right) + S_{contract}(n) + S_{expand}(n) + k$$

Observemos que tanto  $S_{contract}(n)$  como  $S_{expand}(n)$  son  $O(1)$ , pues para la función **contract** ya analizamos su costo en **reduceS** y en el caso de **expand**, como está definida en términos de **tabulateS**, que es  $O(1)$ , también resulta  $\mathbf{expand} \in O(1)$ . Entonces, sabiendo todo esto, y observando de nuevo la recurrencia  $S_{scanS}$ , podemos ver que nos termina quedando similar a la de **reduceS**, entonces resulta  $S_{scanS} \in O(\log(n))$ .