UNR - FCEIA

Estructuras de Datos y Algoritmos II

Análisis de costos

Antuña Pablo

Garavano Lautaro

Índice

1. Costos de las implementaciones con listas

1.1. mapS

```
1 mapS _ [] = emptyS
2 mapS f (x : xs) =
3   let (y, ys) = f x ||| mapS f xs
4    in y : ys
```

Sea s la secuencia sobre la cual se aplica *mapS*
Sea f la función con la cual se aplica *mapS*
Sea n el largo de la secuencia *s*

1.1.1. Trabajo

Dada la definición de la función, podemos ver lo siguiente:

$$W_{mapS}(0) = c_0$$

$$W_{mapS}(n) = c_1 + W_f(s_0) + W_{mapS}(n-1)$$

Donde c_1 es el factor constante correspondiente al uso de `let` y `:`, $W_f(s_0)$ es el trabajo de la llamada izquierda y $W_{mapS}(n-1)$ es el trabajo de la llamada derecha. Adivinamos que

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

Demostración. Para probar esto primero probaremos que para algún $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ / $\forall n \geq n_0$, se cumple

$$0 \leq W_{mapS}(n) \leq c \cdot \sum_{i=0}^{n-1} W_f(s_i)$$

Realizaremos la prueba por inducción.

Caso base: n = 1

$$W_{mapS}(1) = c_1 + W_f(s_0) + c_0 \leq c * W_f(s_0)$$

Donde $c > 1 + c_0 + c_1$.
Suponemos que para algún $c \in \mathbb{R}^+$ se cumple

$$W_{mapS}(n) \leq c \cdot \sum_{i=0}^{n-1} W_f(s_i)$$

para $n > 1$. Esta será nuestra hipótesis inductiva.
Queremos probar ahora que vale para $n + 1$

$$W_{mapS}(n+1) \leq c \cdot \sum_{i=0}^{(n+1)-1} W_f(s_i)$$

Sabemos que

$$W_{mapS}(n+1) = c_1 + W_f(s_0) + W_{mapS}(n)$$

Aplicando nuestra HI, tenemos

$$W_{mapS}(n+1) \leq c_1 + W_f(s_0) + c \cdot \sum_{i=0}^{n-1} W_f(s_i)$$

Podemos encontrar una constante d tal que $c_1 + W_f(s_0) \leq d \cdot W_f(s_0)$.
Por ejemplo, podemos tomar cualquier $d \geq \frac{c_1}{W_f(s_0)} + 1$
Entonces, podemos acotar a la ecuación anterior con

$$W_{mapS}(n+1) \leq d \cdot W_f(s_0) + c \cdot \sum_{i=0}^{n-1} W_f(s_i)$$

Si $c' = \text{máx}(c, d)$, podemos decir que

$$d \cdot W_f(s_0) + c \cdot \sum_{i=0}^{n-1} W_f(s_i) \leq c' \cdot \sum_{i=0}^{n} W_f(s_i)$$

Llegamos de esta forma a una cota para $W_{mapS}(n+1)$

Teniendo este resultado, probaremos que $W_{mapS} \in O(\sum_{i=0}^{n-1} W_f(s_i))$.
Esta prueba consiste en aplicar la definición de O, estableciendo c igual al valor que satisface la cota probada anteriormente, y n_0 igual a cualquier $n > 1$.

□

1.1.2. Profundidad

Dada la definición de la función, podemos ver lo siguiente:

$$S_{mapS}(0) = c_0$$

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

Intentaremos probar que

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

Demostración. De manera similar a lo que hicimos en el trabajo, primero demostraremos que

$$0 \leq S_{mapS}(n) \leq c \cdot (\max_{i=0}^{n-1} S_f(s_i) + n)$$

para algún $c \in \mathbb{R}$ y $n_0 \in \mathbb{N}$ / $\forall n \geq n_0$.Esta prueba será por inducción, y esa será nuestra HI.
Caso base: $n = 1$

$$S_{mapS}(1) = c_1 + \text{máx}(S_f(s_0), c_0)$$

Vemos que tenemos dos casos, uno en el que máx devuelve el valor $S_f(s_0)$, que es igual a $\text{máx}_{i=0}^{1} S_f(s_i)$. Por otro lado, $n = 1$, por lo que $c_1 = c_1 \cdot n$. Tomando entonces a $c = \text{máx}(c_1, 1)$, tendríamos

$$S_{mapS}(1) \leq c_1 \cdot n + \max_{i=0}^{1} S_f(s_i) \leq c \cdot (\max_{i=0}^{1} S_f(s_i) + n)$$

En el segundo caso, tenemos a $S_{mapS}(1)$ acotado por una constante. Nuevamente, al ser $n = 1$, podemos tomar a $c = c_1 + c_0$ y concluir que

$$S_{mapS}(1) \leq c_1 + c_0 \leq c \cdot (\max_{i=0}^{1} S_f(s_i) + n)$$

Probaremos ahora el caso inductivo. Planteamos $S_{mapS}(n+1)$ como

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

Aplicando nuestra HI, tenemos

$$S_{mapS}(n+1) \leq c_1 + \max(S_f(s_0) + c \cdot (\max_{i=0}^{n-1} S_f(s_i) + n))$$

De manera similar al caso base, podemos hacer una división en casos de esta ecuación. En uno, el máximo devuelve el primer valor. En este caso, sabemos que

$$S_f(s_0) \leq \max_{i=0}^{n} S_f(s_i)$$

Puesto que $S_f(s_0)$ está incluido en el conjunto en el cual se calcula el máximo. De esta manera, tenemos

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

De esta manera, tenemos la cota deseada.
En el otro caso, tenemos

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

Es fácil ver que $\max_{i=0}^{n-1} S_f(s_i) \leq \max_{i=0}^{n} S_f(s_i)$, y que por lo tanto, nuestra expresión es acotada por

$$S_{mapS}(n+1) \leq c' \cdot (\max_{i=0}^{n} S_f(s_i) + (n+1))$$

siendo $c' \geq c_1 + c$.
Probamos de esta forma la cota que deseamos. Aplicando la definición de O

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

□

1.2. appendS

```
1  appendS xs [] = xs
2  appendS [] ys = ys
3  appendS (x:xs) ys = x : appendS xs ys
```

1.2.1. Trabajo

Dada la definición de la función, vemos que la recurrencia se realiza en relación a la longitud de la primer lista, y podemos ver lo siguiente:

$$W_{appendS}(0) = c_0$$

$$W_{appendS}(n) = \begin{cases} c_0 & |ys| = 0 \\ c_1 + W_{appendS}(n-1) & |ys| \geq 0 \end{cases}$$

Vamos a demostrar que

$$W_{appendS} \in O(n)$$

Demostración. Para probar esto primero probaremos que para algún $c \in \mathbb{R}$ y $n_0 \in \mathbb{N} \ / \ \forall n \geq n_0$ se cumple

$$0 \leq W_{appendS} \leq c \cdot n$$

Realizaremos la prueba por inducción.
Caso base: n = 1

$$W_{appendS}(1) = c_1 + c_0$$

Si consideramos $c = c_1 + c_0$ vemos que se cumple

$$W_{appendS}(1) \leq c \cdot 1$$

Suponemos que para algún $c \in \mathbb{R}^+$ se cumple

$$0 \leq W_{appendS} \leq c \cdot n$$

para $n > 1$. Esta será nuestra hipótesis inductiva.
Queremos probar ahora que se cumple para $n + 1$, tenemos entonces[1]

$$W_{appendS}(n+1) = c_1 + W_{appendS}(n)$$

Aplicando la hipótesis inductiva

$$c_1 + W_{appendS}(n) \leq c_1 + c \cdot n \leq c_1 + c \cdot (n+1) \leq d \cdot (n+1)$$

Donde $d \geq c + c_1$. Llegamos entonces a lo que queríamos por lo tanto podemos afirmar que

$$W_{appendS} \in O(n)$$

□

1.2.2. Profundidad

Viendo la implementación de appendS podemos ver

$$S_{appendS}(0) = c_0$$

$$S_{appendS}(n) = \begin{cases} c_0 & |ys| = 0 \\ c_1 + S_{appendS}(n-1) & |ys| \geq 0 \end{cases}$$

Vamos a demostrar que

$$S_{appendS} \in O(n)$$

Demostración. Para probar esto primero probaremos que para algún $c \in \mathbb{R}$ y $n_0 \in \mathbb{N}$ / $\forall n \geq n_0$ se cumple

[1]Ignoramos en este análisis el caso en el que $|ys| = 0$, ya que su costo c_0 es trivialmente acotado por $c_0 n$

$$0 \leq S_{appendS} \leq c \cdot n$$

Realizaremos la prueba por inducción.
Caso base: n = 1

$$S_{appendS}(1) = c_1 + c_0$$

Si consideramos $c = c_1 + c_0$ vemos que se cumple

$$S_{appendS}(1) \leq c \cdot 1$$

Suponemos que para algún $c \in \mathbb{R}^+$ se cumple

$$0 \leq S_{appendS} \leq c \cdot n$$

para $n > 1$. Esta será nuestra hipótesis inductiva.
Queremos probar ahora que se cumple para $n + 1$, tenemos entonces[2]

$$S_{appendS}(n + 1) = c_1 + S_{appendS}(n)$$

Aplicando la hipótesis inductiva

$$c_1 + S_{appendS}(n) \leq c_1 + c \cdot n \leq c_1 + c \cdot (n + 1) \leq d \cdot (n + 1)$$

Donde $d \geq c + c_1$. Llegamos entonces a lo que queríamos por lo tanto podemos afirmar que

$$S_{appendS} \in O(n)$$

□

[2]Nuevamente, el caso en el que $|ys| = 0$ es trivialmente acotado por $c_0 n$

1.3. reduceS

```
1  reduceS f b [] = b
2  reduceS f b s = f b (reduceS' s)
3    where
4      reduceS' [s1] = s1
5      reduceS' s = reduceS' (contraer s)
6      contraer [] = []
7      contraer [x1] = [x1]
8      contraer (x1:x2:xs) = let (y, ys) = f x1 x2 ||| contraer xs in y:ys
```

Para el análisis de costo de esta función se considerará $W_f \in O(1)$ y $Sf \in O(1)$

1.3.1. Trabajo

$$W_{reduceS}(0) = c_0$$

$$W_{reduceS}(n) = W_{reduceS'}(n) + W_f(b, n_0)$$

$$W_{reduceS}(n) = W_{reduceS'}(n) + c_1$$

Podemos ver que para calcular el costo de `reduceS` debemos calcular primero el de `reduceS'`.

$$W_{reduceS'}(1) = c_0$$

$$W_{reduceS'}(n) = W_{reduceS'}(\left\lceil \frac{n}{2} \right\rceil) + W_{contraer}(n)$$

Nuevamente podemos ver que para el calculo de este costo debemos calcular primero el de otra función, en este caso el de la función `contraer`

$$W_{contraer}(0) = c_0$$

$$W_{contraer}(1) = c_1$$

$$W_{contraer}(n) = c_2 + W_f(s_0, s_1) + W_{contraer}(n-2)$$

$$W_{contraer}(n) = c_3 + W_{contraer}(n-2)$$

Demostraremos que $W_{reduceS} \in O(n)$. Para eso, acotaremos $W_{contraer}$, $W_{reduceS'}$ y finalmente $W_{reduceS}$.

Demostración. Comenzaremos por que $W_{contraer} \in \Theta(n)$
Realizaremos la prueba por inducción. Primero veremos que $W_{contraer} \in \Omega(n)$
Caso base: $n = 1$

$$W_{contraer} = c_1 \geq c_1 \cdot n$$

Caso inductivo: Planteamos $W_{contraer}(n+1)$

$$W_{contraer}(n+1) = c_3 + W_{contraer}(n-1)$$

Aplicando HI

$$W_{contraer}(n+1) \geq c_3 + c \cdot (n-1) \geq c \cdot (n-1) \geq c'(n+1)$$

siendo $c' \leq c \cdot \frac{n-1}{n+1}$. Con esta cota del costo, podemos aplicar la definición de Ω y concluir que $W_{contraer} \in \Omega(n)$.
Ahora veamos que $W_{contraer} \in O(n)$
Caso base: $n = 1$

$$W_{contraer} = c_1 \leq c_1 \cdot n$$

Caso inductivo: Planteamos $W_{contraer}(n+1)$

$$W_{contraer}(n+1) = c_3 + W_{contraer}(n-1)$$

Aplicando HI

$$W_{contraer}(n+1) \leq c_3 + c \cdot (n-1) \leq c_3 + c \cdot (n+1) \leq c' \cdot (n+1)$$

siendo $c' \geq c_3 + c$. Con esta cota del costo, podemos aplicar la definición de O y concluir que $W_{contraer} \in O(n)$.

$$\therefore W_{contraer} \in \Theta(n)$$

Probaremos ahora que $W_{reduceS'} \in \Theta(n)$. Para lograr este resultado, primero asumiremos una entrada $n = 2^k$ con $k \in \mathbb{N}$.
Con esta entrada, podemos simplificar nuestra función de trabajo a la siguiente forma.

$$W_{reduceS'}(1) = c_0$$
$$W_{reduceS'}(n) = W_{reduceS'}(\frac{n}{2}) + W_{contraer}(n)$$

Esta función es de la forma $T(n) = aT(n/b) + f(n)$, con lo cual podemos aplicar el teorema maestro. En particular, nos encontramos con el tercer caso del mismo, puesto que $f(n) = W_{contraer}(n) \in \Omega(n) = \Omega(n^{\log_b(a+\epsilon)})$ con $a = 1, b = 2, \epsilon = 1$. Además, satisfacemos la segunda condición de este caso, ya que, para algún valor de c,

$$a \cdot f(n/b) \leq c \cdot f(n)$$

$$\iff W_{contraer}(n/2) \leq c \cdot W_{contraer}(n)$$

Sabemos que

$$W_{contraer}(n) = c_3 + W_{contraer}(n-2) = 2c_3 + W_{contraer}(n-4) = ... = \frac{n}{4}c_3 + W_{contraer}(\frac{n}{2})$$

por lo que cuando

$$c \geq \frac{1}{\frac{n}{4} \cdot c_3}$$

se cumple la condición que necesitamos para aplicar el teorema maestro. Este teorema nos dice entonces que $W_{reduceS'} \in \Theta(W_{contraer}) = \Theta(n)$. Dado que $f(n) = n$ es suave, podemos aplicar la regla de suavidad y extender la demostración de que $W_{reduceS'} \in \Theta(n)$ para todas las entradas posibles de n (en vez de sólo para potencias de 2). Demostramos de esta forma que $W_{reduceS'} \in \Theta(n)$.
Hecha esta demostración, solo debemos probar que $W_{reduceS} \in \Theta(n)$. Realizaremos

esta prueba por dos casos, ya que.
Caso $n = 1$
En este caso tenemos trivialmente

$$c_0 n \leq W_{reduceS} = c_0 \leq c_0 n \implies W_{reduceS} \in \Theta(n)$$

Caso $n > 1$

$$W_{reduceS'}(n) \in \Theta(n) \implies W_{reduceS}(n) = c_1 + W_{reduceS'}(n) \in \Theta(n)$$

De esta manera concluimos que $W_{reduceS}(n) \in \Theta(n)$.

□

1.3.2. Profundidad

Ahora vamos a realizar un análisis de la profundidad de esta función. Al igual que con el trabajo, veremos el como sería el costo de cada una de las funciones involucradas.
Primero veremos `reduceS`

$$S_{reduceS}(0) = c_0$$

$$S_{reduceS}(n) = S_{reduceS'}(n) + S_f(b, n_0)$$

$$S_{reduceS}(n) = S_{reduceS'}(n) + c_1$$

Luego veremos `reduceS'`

$$S_{reduceS'}(1) = c_0$$

$$S_{reduceS'}(n) = S_{reduceS'}(\left\lceil \frac{n}{2} \right\rceil) + S_{contraer}(n)$$

Por ultimo veremos la ultima función involucrada, la cual es `contraer`

$$S_{contraer}(0) = c_0$$

$$S_{contraer}(1) = c_1$$

$$S_{contraer}(n) = c_2 + \text{máx}(S_f s_0, s_1, S_{contraer}(n-2))$$

$$S_{contraer}(n) = c_2 + \text{máx}(c_3, S_{contraer}(n-2))$$

Veamos que $S_{reduceS} \in O(n)$

Demostración. Primero procedamos a demostrar que $S_{contraer} \in \Theta(n)$. Para eso, probaremos que pertenece a $O(n)$ y luego que pertenece a $\Omega(n)$. Esto lo haremos por inducción.
Caso base: $n = 1$

$$S_{contraer} = c_1 \leq c_1 \cdot n$$

Caso inductivo: Planteamos $S_{contraer}(n+1)$

$$S_{contraer}(n+1) = c_2 + \text{máx}(c_3, S_{contraer}(n-1))$$

Aplicando HI

$$S_{contraer}(n+1) \leq c_2 + \text{máx}(c_3, c \cdot (n-1))$$

Si $\text{máx}(c_3, c \cdot (n-1))$ devuelve el valor c_3 y tomamos $c' \geq c_2 + c_3$ resulta

$$c_2 + c_3 \leq c' \cdot n$$

Si $\text{máx}(c_3, c \cdot (n-1))$ devuelve el valor $c \cdot (n-1)$ y tomamos $c' \geq c_2 + c$ resulta:

$$S_{contraer}(n+1) \leq c_2 + c \cdot (n-1) \leq c_2 + c \cdot (n+1) \leq c' \cdot (n+1)$$

Con esta cota del costo, podemos aplicar la definición de O y concluir que

$$S_{contraer} \in O(n)$$

De manera similar probaremos que $S_{contraer} \in \Omega(n)$.
Caso base: $n = 1$

$$S_{contraer} = c_1 \geq c_1 \cdot n$$

Caso inductivo: Planteamos $S_{contraer}(n+1)$

$$S_{contraer}(n+1) = c_2 + \text{máx}(c_3, S_{contraer}(n-1))$$

Aplicando nuestra HI, tenemos

$$S_{contraer}(n+1) \geq c_2 + \text{máx}(c_3, c \cdot (n-1))$$

En el caso de que máx devuelva c_3, basta con elegir $c \leq \frac{\text{mín}(c_2,c_3)}{n}$ y tendremos

$$S_{contraer}(n+1) \geq c_2 + c_3 \geq c \cdot n$$

En el otro caso, tenemos

$$S_{contraer}(n+1) \geq c_2 + c \cdot (n-1) \geq c' \cdot (n+1)$$

donde

$$c' \leq \frac{c_2 + c \cdot (n-1)}{n+1}$$

De esta manera probamos que $S_{contraer} \in \Omega(n)$, y al estar también en $O(n)$, sabemos que pertenece a $\Theta(n)$
Ahora bien, observemos que `contraer` es la única función involucrada que utiliza operaciones en paralelo, sin embargo concluimos que la profundidad tiene el mismo orden que el trabajo. Resulta entonces, al no haber paralelización por parte de las otras funciones

$$S_{reduceS} \in O(n)$$

□

1.4. scanS

```
scanS _ b [] = ([], b)
scanS f b [s1] = ([b], f b s1)
scanS f b s = expandir s (scanS f b (contraer s))
  where
    contraer [] = []
    contraer [x1] = [x1]
    contraer (x1:x2:xs) = let (y, ys) = f x1 x2 ||| contraer xs in y:ys
    expandir _ s'@([], _) = s'
    expandir [] (_, t) = ([], t)
    expandir [s1] s' = s'
    expandir (s1:_:ss) (x1:xs, t) = let (interp, (exp, red)) =
      f x1 s1 ||| expandir ss (xs, t) in (x1 : interp : exp, red)
```

Para el análisis de costo de esta función se considerará $W_f \in O(1)$ y $Sf \in O(1)$

1.4.1. Trabajo

$$W_{scanS}(0) = c_0$$

$$W_{scanS}(1) = c_1 + W_f(s_0)$$

$$W_{scanS}(n) = c_2 + W_{contraer}(n) + W_{scanS}(\left\lceil \frac{n}{2} \right\rceil) + W_{expandir}(n)$$

Si vemos a la función contraer, notaremos que es la misma función contraer utilizada en reduce, la cual sabemos pertenece a $\Theta(n)$. La función expandir, a su vez, tiene los costos

$$W_{expandir}(0) = c_0$$

$$W_{expandir}(1) = c_1$$

$$W_{expandir}(n) = W_f + W_{expandir}(n-2) + c_2$$

(Consideramos a $W_{expandir}(n)$ como la aplicación de expandir cuyo primer argumento es de largo n).

Probaremos que $W_{expandir} \in O(n)$

$$W_{expandir}(0) = c_0 \in O(n)$$

$$W_{expandir}(1) = c_1 \in O(n)$$

Utilizaremos la siguiente HI

$$W_{expandir}(n) \in O(n)$$

Tenemos entonces

$$W_{expandir}(n+1) = W_f + W_{expandir}(n-1) + c_2$$

$$\leq W_f + c' \cdot (n-1) + c_2 \leq c(n-1) \leq c(n+1)$$

Donde $c \geq W_f + c' + c_2$. Entonces, $W_{expandir} \in O(n)$. Probaremos ahora que pertenece a $\Omega(n)$.

$$W_{expandir}(0) = c_0 \in \Omega(n)$$

$$W_{expandir}(1) = c_1 \in \Omega(n)$$

Nuestra HI será

$$W_{expandir}(n) \in \Omega(n)$$

Planteamos el caso inductivo

$$W_{expandir}(n+1) = W_f + W_{expandir}(n-1) + c_2$$

$$\geq W_f + c' \cdot (n-1) + c_2 \geq c' \cdot (n-1)$$

$$\geq c(n+1)$$

Donde

$$c \leq \frac{c'(n-1)}{(n+1)}$$

Demostramos entonces que $W_{expandir}$ pertenece a $\Omega(n)$, y también a $\Theta(n)$

A continuación probaremos que $W_{scanS} \in \Theta(n)$ cuando la entrada es una potencia de dos. En este caso, tenemos

$$W_{scanS} = c_2 + W_{contraer}(n) + W_{scanS}(\frac{n}{2}) + W_{expandir}(n)$$

Vemos que W_{scanS} tiene la forma $T(n) = a \cdot T(n/b) + f(n)$. En este caso,

$$f(n) = c_2 + W_{contraer}(n) + W_{expandir}(n)$$

Notamos que

$$f(n) = \Theta(1) + \Theta(n) + \Theta(n) = \Theta(n)$$

En nuestro caso, si quisiéramos aplicar el teorema maestro, tendríamos $a = 1$, $b = 2$, y por lo tanto $\log_b a = 0$. Si tomamos $\epsilon = 1$, vemos que efectivamente $f(n) \in \Omega(n^{\log_b(a+\epsilon)})$. Ahora debemos probar que, para algún $c < 1$,

$$a \cdot f(n/b) \leq c \cdot f(n)$$

En nuestro caso, esta fórmula se ve como

$$f(n/2) \leq c \cdot f(n)$$

Si $c = \frac{1}{2}$, tenemos

$$f(n/2) = W_{contraer}(n/2) + W_{expandir}(n/2) + c_2 \leq \frac{W_{contraer}(n) + W_{expandir}(n) + c_2}{2} = \frac{f(n)}{2}$$

Como $W_{contraer}$ y $W_{expandir}$ pertenecen a $\Theta(n)$, esto vale si

$$\frac{n}{2} + \frac{n}{2} \leq \frac{n+n}{2}$$

lo cual vale. Entonces, aplicamos el teorema maestro para obtener que $W_{scanS} \in \Theta(n)$, al menos para valores de la forma 2^k, $k \in \mathbb{N}$. Como sabemos que la función $g(n) = n$ es suave, podemos aplicar la regla de suavidad y extender este resultado para todas las entradas de W_{scanS}.

Llegamos así a la conclusión de que $W_{scanS} \in \Theta(n)$

1.4.2. Profundidad

La profundidad de scanS está dada por

$$S_{scanS}(0) = c_0$$

$$S_{scanS}(1) = c_1$$

$$S_{scanS}(n) = S_{contraer}(n) + S_{scanS}(\left\lceil \frac{n}{2} \right\rceil) + S_{expandir}(n) + c_2$$

A su vez,

$$S_{expandir}(0) = c_0$$

$$S_{expandir}(1) = c_1$$

$$S_{expandir}(n) = \text{máx}(S_f, S_{expandir}(n-2)) + c_2$$

Sabemos del ejercicio anterior que $S_{contraer} \in \Theta(n)$
Veremos también que $S_{expandir} \in \Theta(n)$.
Primero, probaremos que pertenece a $O(n)$

$$S_{expandir}(0) = c_0 \in O(n)$$

$$S_{expandir}(1) = c_1 \in O(n)$$

HI: $S_{expandir}(n) \in O(n)$
Caso inductivo:

$$S_{expandir}(n+1) = \text{máx}(S_f, S_{expandir}(n-1)) + c_2$$

Aplicando HI

$$S_{expandir}(n+1) \leq \text{máx}(S_f, c' \cdot (n-1)) + c_2$$

Tenemos dos casos, según el resultado del máx. En el primer caso

$$S_{expandir}(n+1) \leq S_f + c_2 \leq c(n+1)$$

vale para cualquier $c \geq S_f + c_2$. En el otro caso, tenemos

$$S_{expandir}(n+1) \leq c' \cdot (n-1) + c_2 \leq c(n-1) \leq c(n+1)$$

Siendo $c \geq c' + c_2$. Probamos entonces que $S_{expandir} \in O(n)$. Probaremos ahora, de manera similar, que pertenece a $\Omega(n)$.

$$S_{expandir}(0) = c_0 \in \Omega(n)$$

$$S_{expandir}(1) = c_1 \in \Omega(n)$$

$$S_{expandir}(n+1) = \text{máx}(S_f, S_{expandir}(n-1)) + c_2 \overset{HI}{\geq} \text{máx}(S_f, c'(n-1)) + c_2$$

$$\geq c'(n-1) + c_2 \geq c(n+1)$$

Donde $c \leq \frac{c'(n-1)+c_2}{n+1}$.
Queda demostrado entonces que

$$S_{expandir} \in \Theta(n)$$

.
De la misma manera que en reduce, notamos que scanS no posee ninguna operación capaz de realizarse en paralelo. Por lo tanto, podemos hacer un análisis análogo al del trabajo para demostrar que $S_{scanS} \in \Theta(n)$. Esto es posible debido a que tanto

$$S_{expandir}$$

como

$$S_{contraer}$$

son del mismo orden que

$$W_{expandir}$$

y

$$W_{contraer}$$

respectivamente.

2. Costos de las implementaciones con arrays

2.1. mapS

```
1  mapS f seq = tabulateS (f . nthS seq) (lengthS seq)
```

2.1.1. Trabajo

Viendo la función podemos decir que el trabajo es:

$$W_{mapS}(n) = W_{tabulateS}(g, n) + W_{lengthS}(n)$$

Donde $g = f.nthSseq$
Queremos probar que $W_{mapS} \in O(\sum_{i=0}^{n-1} W_f(s_i))$

Demostración. Comenzaremos viendo que $W_g(i) \in O(W_f(s_i))$

$$w_g(i) = W_f(s_i) + W_{nthS}(i)$$

Sabemos por la tabla de complejidades proporcionada que $W_{nthS} \in O(1)$. Resulta entonces considerando $c \geq c_0 + 1$

$$W_g(i) \leq W_f(s_i) + c_0 \leq c \cdot W_f(s_i)$$

Por lo tanto por definición de O concluimos $W_g(i) \in O(W_f(s_i))$.
Por la tabla de complejidades también sabemos que $W_{tabulateS}(g, n) \in O(\sum_{i=0}^{n-1} W_g(i))$ y además $W_{lengthS} \in O(1)$.
Dicho esto procedamos a analizar `mapS`

$$W_{mapS}(n) \leq c' \cdot \sum_{i=0}^{n-1} W_g(i) + c''$$

Por lo demostrado anteriormente tenemos

$$c' \cdot \sum_{i=0}^{n-1} W_g(i) + c'' \leq c' \cdot \sum_{i=0}^{n-1} (c_i''' \cdot W_f(s_i)) + c'' \leq c' \cdot c''' \sum_{i=0}^{n-1} W_f(s_i) + c'' \leq c \cdot \sum_{i=0}^{n-1} W_f(s_i)$$

donde tomamos

$$c''' \geq \max_{i=0}^{n-1} c_i'''$$
$$c \geq c' \cdot c''' + c''$$

De esta forma utilizando la definición de O obtenemos

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

□

2.1.2. Profundidad

Sabemos que tenemos los siguientes costos $S_{nthS} \in O(1)$, $S_{lengthS} \in O(1)$ y $S_{tabulateS}(f, n) \in O(\max_{i=0}^{n} S_f(i))$
Luego viendo la función podemos decir que la profundidad es:

$$S_{mapS}(n) = S_{tabulateS}(g, n) + S_{lengthS}(n)$$

Donde $g = f.nthSseq$
Queremos probar que $S_{mapS} \in O(\max_{i=0}^{n-1} S_f(s_i))$

Demostración. Para esto comenzaremos observando la profundidad de g la cual resulta

$$S_g(i) = S_f(s_i) + S_{nthS} = S_f(s_i)$$

Entonces resulta $S_g(i) \in O(S_f(s_i))$.
Continuando con el costo de *mapS* tenemos

$$O(\max_{i=0}^{n} S_g(i)) + O(1) = O(\max_{i=0}^{n} S_f(s_i)) + O(1) = O(\max_{i=0}^{n} S_f(s_i))$$

$$\therefore S_{mapS} \in O(\max_{i=0}^{n} S_f(s_i))$$

□

2.2. appendS

```
1  appendS xs ys = tabulateS aux (largoXS + largoYS)
2    where
3      largoXS = lengthS xs
4      largoYS = lengthS ys
5      aux n
6        | n < largoXS = nthS xs n
7        | otherwise = nthS ys (n - largoXS)
```

2.2.1. Trabajo

Sean `n,m` el largo de las secuencias a unir y sabemos que tenemos los siguientes costos $W_{nthS} \in O(1)$, $W_{lengthS} \in O(1)$ y $W_{tabulateS}(f, n) \in O(\sum_{i=0}^{n} W_f(i))$. Veamos como queda el trabajo de la función

$$W_{appendS}(n, m) = W_{tabulateS}(aux, n + m) + 2 \cdot W_{lengthS}$$

Queremos demostrar que $W_{appendS} \in O(n + m)$

Demostración. Para esto comenzaremos observando la función `aux` la cual tiene el siguiente trabajo

$$W_{aux} = W_{nthS} + c_0 = O(1)$$

Continuando con el costo de *appendS* tenemos

$$O(\sum_{i=0}^{n+m} W_{aux}(i))+2\cdot W_{lengthS} = O(\sum_{i=0}^{n+m} O(1))+2\cdot O(1) = O(n+m)+O(1) = O(n+m)$$

$$\therefore W_{appendS} \in O(n+m)$$

□

2.2.2. Profundidad

Sean `n,m` el largo de las secuencias a unir y sabemos que tenemos los siguientes costos $S_{nthS} \in O(1)$, $S_{lengthS} \in O(1)$ y $S_{tabulateS}(f,n) \in O(\text{máx}_{i=0}^{n} S_f(i))$. Veamos como queda la profundidad de la función

$$S_{appendS}(n,m) = S_{tabulateS}(aux, n+m) + 2 \cdot S_{lengthS}$$

Queremos demostrar que $S_{appendS} \in O(1)$

Demostración. Para esto comenzaremos observando la función `aux` la cual tiene la siguiente profundidad

$$S_{aux} = S_{nthS} + c_0 = O(1)$$

Continuando con el costo de *appendS* tenemos

$$O(\max_{i=0}^{n+m} S_{aux}(i)) + 2 \cdot S_{lengthS} = O(\max_{i=0}^{n+m} O(1)) + 2 \cdot O(1) = 3 \cdot O(1) = O(1)$$

$$\therefore S_{appendS} \in O(1)$$

□

2.3. reduceS

```haskell
reduceS f b seq
  | largo == 0 = b
  | otherwise = f b (reduceS' seq)
  where
    largo = lengthS seq
    reduceS' seq
      | largo == 1 = nthS seq 0
      | otherwise = reduceS' (contraer seq)
      where
        largo = lengthS seq
    contraer seq
      | largo < 2 = seq
      | otherwise = tabulateS (\x -> if (2 * x + 1) == largo
        then nthS seq (2 * x)
        else f (nthS seq (2 * x)) (nthS seq (2 * x + 1))) mitad
      where
        largo = lengthS seq
        mitad = ceiling (fromIntegral largo / 2)
```

Para el análisis de costo de esta función se considerará $W_f \in O(1)$ y $S_f \in O(1)$

2.3.1. Trabajo

$$W_{reduceS}(0) = c_0$$

$$W_{reduceS}(n) = c_1 + W_{reduceS'}(n)$$

Donde `reduceS'` es

$$W_{reduceS'}(1) = c_0$$

$$W_{reduceS'}(n) = c_1 + W_{reduceS'}(\left\lceil \frac{n}{2} \right\rceil) + W_{contraer}(n)$$

Y contraer es

$$W_{contraer}(0) = W_{contraer}(1) = c_0$$

$$W_{contraer}(n) = W_{tabulateS}(g, \left\lceil \frac{n}{2} \right\rceil) + c_1$$

Donde la función `g` es la lambda que vemos en el código.
Como consideramos $W_f \in \Theta(1)$ queremos demostrar que $W_{reduceS} \in O(n)$

Demostración. Para esto comencemos por analizar la función `g`. Para ver su trabajo analizaremos los dos casos posibles del `if`.
Si cumple la condición resulta

$$W_g = W_{nthS} + c_0 = W_{nthS} = \Theta(1) + c_0 = \Theta(1)$$

Si no cumple la condición resulta

$$W_g = W_f + 2 \cdot W_{nthS} = \Theta(1) + 2 \cdot \Theta(1) = \Theta(1)$$

Llegando entonces a

$$W_g \in \Theta(1)$$

De esta manera, el costo de `tabulate` se simplifica a $\Theta(n)$.
Siguiendo con el costo de `contraer` tenemos

$$W_{contraer}(n) = W_{tabulate}(\left\lceil \frac{n}{2} \right\rceil) + c_1 = \Theta(\left\lceil \frac{n}{2} \right\rceil) + \Theta(1) = \Theta(n)$$

$$\therefore W_{contraer} \in \Theta(n)$$

Luego volviendo al costo de `reduceS'` tenemos

$$W_{reduceS'}(n) = c_1 + W_{reduceS'}(\left\lceil \frac{n}{2} \right\rceil) + W_{contraer}(n) = W_{reduceS'}(\left\lceil \frac{n}{2} \right\rceil) + \Theta(n)$$

Queremos probar que esta función pertenece a $O(n)$, que es suave. Por lo tanto, podemos restringirnos a entradas que sean potencias de dos y remover el ceiling. Notamos entonces que en la función resultante podemos aplicar el teorema maestro. Nuestra f, en este caso, sería $c_1 + W_{contraer}(n) \in \Theta(n)$. Por lo tanto, sabemos que

pertenece a $\Omega(n^{\log_2(1+\epsilon)})$ con $\epsilon = 1$. Probaremos ahora la segunda condición para aplicar el tercer caso del teorema maestro.

$$f(n/2) \leq c \cdot f(n)$$

$$c_1 + W_{contraer}(n/2) \leq c \cdot (c_1 + W_{contraer}(n))$$

Como $W_{contraer} \in \Theta(n)$, sabemos que esto se cumplirá cuando $c \geq \frac{1}{2}$.
Aplicando entonces el teorema maestro, podemos concluir que $W_{reduceS'} \in \Theta(n)$. Como dijimos antes, este resultado que acabamos de demostrar funciona para todas las entradas ya que la función $f(n) = n$ es suave.
Volviendo a `reduceS` tenemos $W_{reduceS}(n) = c_1 + W_{reduceS'}(n)$

$$\therefore W_{reduceS} \in \Theta(n)$$

□

2.3.2. Profundidad

$$S_{reduceS}(0) = c_0$$

$$S_{reduceS}(n) = c_1 + S_{reduceS'}(n)$$

$$S_{reduceS'}(1) = c_0$$

$$S_{reduceS'}(n) = c_1 + S_{reduceS'}(\left\lceil \frac{n}{2} \right\rceil) + S_{contraer}(n)$$

$$S_{contraer}(0) = S_{contraer}(1) = c_0$$

$$S_{contraer}(n) = S_{tabulateS}(g, \left\lceil \frac{n}{2} \right\rceil) + c_1$$

Donde g es la lambda function definida.
Queremos demostrar que $S_{reduceS} \in O(\log_2 n)$

Demostración. Comencemos analizando qué sucede con la profundidad de `g`.

Dado que $f(n) \in \Theta(1)$ y que en g solo se realizan operaciones constantes, sabemos que $S_g \in \Theta(1)$. De esta forma, tenemos

$$S_{tabulateS}(g, n) = \max_{i=0}^{n-1} S_g(i) = \max_{i=0}^{n-1} \Theta(1) = \Theta(1)$$

Entonces, en $S_{contraer}$

$$S_{contraer}(n) = S_{tabulateS}(g, \left\lceil \frac{n}{2} \right\rceil) + c_1 = \Theta(1) + c_1 = \Theta(1)$$

Para `reduceS'`, primero restringiremos nuestras entradas a potencias de dos. Entonces,

$$S_{reduceS'}(n) = c_1 + S_{reduceS'}(\frac{n}{2}) + S_{contraer}(n) = S_{reduceS'}(\frac{n}{2}) + \Theta(1)$$

En este caso podemos aplicar el teorema maestro con $a = 1$, $b = 2$ y $f(n) = c_1 + S_{contraer}(n)$. Como $f(n) \in \Theta(1) = \Theta(n^{\log_b(a)})$, podemos aplicar el segundo caso y concluir que

$$S_{reduceS'}(n) \in \Theta(n^{\log_b(a)} \log_2 n) = \Theta(\log_2 n)$$

Para entradas potencia de dos. Como $\log_2 n$ es suave, podemos extender este resultado a todas las entradas.
Volviendo a `reduceS`, vemos que su profundidad es $S_{reduceS'}(n)$ más una constante, por lo que también podemos concluir

$$S_{reduceS}(n) \in \Theta(\log_2 n)$$

□

2.4. scanS

```
scanS f b s
  | largo == 0 = (emptyS, b)
  | largo == 1 = (singletonS b, f b (nthS s 0))
  | otherwise = expandir s (scanS f b (contraer s))
  where
    largo = lengthS s
    contraer seq
      | largo < 2 = seq
      | otherwise = tabulateS (\x -> if (2 * x + 1) == largo
        then nthS seq (2 * x)
        else f (nthS seq (2 * x)) (nthS seq (2 * x + 1))) mitad
      where
        largo = lengthS seq
        mitad = ceiling (fromIntegral largo / 2)
    expandir s (s', t) = (tabulateS
        (\x -> if even x
        then nthS s' (x `div` 2)
        else f (nthS s' (x `div` 2)) (nthS s (x - 1)))
      largo, t)
      where
        largo = lengthS s
```

Para el análisis de costo de esta función se considerará $W_f \in O(1)$ y $S_f \in O(1)$

2.4.1. Trabajo

$$W_{scanS}(0) = W_{emptyS} = c_0$$

$$W_{scanS}(1) = W_{singletonS} + W_f + W_{nthS} = c_1$$

$$W_{scanS}(n) = W_{expandir}(n) + W_{scanS}(\left\lceil \frac{n}{2} \right\rceil) + W_{contraer}(n)$$

$$W_{expandir}(n) = W_{tabulateS}(g, n) + W_{lengthS}$$

Donde g es la función anónima definida
Queremos demostrar que $W_{scanS} \in O(n)$

Demostración. Comenzamos analizando el costo de la función `g`. Donde tendremos dos casos.
Si se cumple la condición del `if`

$$W_g = W_{lengthS} + c_0 = \Theta(1)$$

Si no se cumple la condición del `if`

$$W_g = 2 \cdot W_{lengthS} + c_0 = \Theta(1)$$

Podemos ver entonces que resulta

$$W_g \in \Theta(1)$$

Retomando el costo de `expandir`

$$W_{expandir}(n) = W_{tabulateS}(g, n) + W_{lengthS} = \Theta(n) + \Theta(1) = \Theta(n)$$

Recordando que $W_{contraer} \in \Theta(n)$ podemos retomar el analisis de `scanS`

$$W_{scanS}(n) = W_{expandir}(n) + W_{scanS}(\left\lceil \frac{n}{2} \right\rceil) + W_{contraer}(n) =$$

$$= \Theta(n) + W_{scanS}(\left\lceil \frac{n}{2} \right\rceil) + \Theta(n) = W_{scanS}(\left\lceil \frac{n}{2} \right\rceil) + \Theta(n)$$

Restringiendo la entrada a 2^k, tenemos

$$W_{scanS}(n) = W_{scanS}(\frac{n}{2}) + W_{contraer}(n) + W_{expandir}(n)$$

Podemos aplicar aquí el teorema maestro, con $a = 1$, $b = 2$, y $f(n) = W_{contraer}(n) + W_{expandir}(n)$. Vemos que cumplimos la primera condición del tercer caso ya que $f(n) \in \Omega(n) = \Omega(n^{\log_b(a+\epsilon)})$ con $\epsilon = 1$. Veamos ahora la segunda condición.

$$a \cdot f(n/b) \leq c \cdot f(n)$$

$$W_{contraer}(n/2) + W_{expandir}(n/2) \leq c \cdot (W_{contraer}(n) + W_{expandir}(n))$$

Como $f(n) \in \Theta(n)$, esto se cumple cuando $c \geq \frac{1}{2}$.
De esta manera, el teorema maestro nos dice que para entradas potencia de dos, $W_{scanS} \in \Theta(n)$. Como esta función es suave, se extiende para todas las entradas.

$$\therefore W_{scanS} \in O(n)$$

□

2.4.2. Profundidad

$$S_{scanS}(0) = c_0$$

$$S_{scanS}(1) = c_1$$

$$S_{scanS}(n) = S_{contraer}(n) + S_{expandir}(n) + S_{scanS}(\left\lceil \frac{n}{2} \right\rceil)$$

Probaremos que $S_{scanS} \in \Theta(\log_2 n)$

Demostración. Primero, analizaremos la profundidad de `expandir`.

$$S_{expandir}(n) = S_{tabulateS}(g, n) + c_0$$

Donde g es la lambda function definida. Ya que solo utiliza operaciones constantes, como `nthS` y f, podemos concluir que $g \in \Theta(1)$.
Entonces tenemos en `tabulateS`

$$S_{tabulateS}(g, n) = \max_{i=0}^{n-1} S_g(i) = \max_{i=0}^{n-1} \Theta(1) = \Theta(1)$$

Y como expandir solo llama a `tabulateS` y otras operaciones constantes, $S_{expandir} \in \Theta(1)$.
Volviendo a `scanS`, si restringimos la entrada a potencias de dos, tenemos

$$S_{scanS}(n) = S_{scanS}(\frac{n}{2}) + S_{contraer}(n) + S_{expandir}(n) = S_{scanS}(\frac{n}{2}) + \Theta(1)$$

Podemos aplicar el segundo caso del teorema maestro, con $a = 1$, $b = 2$, y

$$f(n) = S_{contraer}(n) + S_{expandir}(n) \in \Theta(1) = \Theta(n^{\log_b a})$$

Aplicando este teorema sabemos que $S_{scanS}(n) \in \Theta(n^{\log_b a} \log_b n) = \Theta(\log_2 n)$ para entradas potencias de dos. Como $\log_2 n$ es suave, podemos extender este resultado a todas las entradas.

$$\therefore S_{scanS}(n) \in \Theta(\log_2 n)$$

□