R-312 Estructuras de Datos y Algoritmos II

Trabajo Práctico 2

Secuencias

Gianni Georg Weinand W-0528/2

20 de Junio, 2018

Índice

1 Introducción

A lo largo del trabajo se utiliza el siguiente modelo de costo:

$$W(c) = 1 \tag{1}$$
$$W(op\ e) = 1 + W(e) \tag{2}$$
$$W(e_1,\ e_2) = 1 + W(e_1) + W(e_2) \tag{3}$$
$$W(e_1 \| e_2) = 1 + W(e_1) + W(e_2) \tag{4}$$
$$W(\text{let } x = e_1 \text{in } e_2) = 1 + W(e_1) + W(e_2[Eval(e_1)/x]) \tag{5}$$
$$W(f(x) : x \in A) = 1 + \sum_{x \in A} W(f(x)) \tag{6}$$
$$S(c) = 1 \tag{7}$$
$$S(op\ e) = 1 + S(e) \tag{8}$$
$$S(e_1,\ e_2) = 1 + S(e_1) + S(e_2) \tag{9}$$
$$S(e1 \| e2) = \max(S(e_1),\ S(e_2)) \tag{10}$$
$$S(\text{let } x = e_1 \text{in } e_2) = 1 + S(e_1) + S(e_2[Eval(e_1)/x]) \tag{11}$$
$$S(f(x) : x \in A) = 1 + \max_{x \in A} S(f(x)) \tag{12}$$

- Por simplicidad se repetirán los nombres de las constantes, entendiendo que en cada análisis pueden ser diferentes. Es decir, la constante c_1 utilizada en el análisis del trabajo de filterS no necesariamente coincide con la constante c_1 utilizada en el análisis de la profundidad de la misma función.
- Se hará otro abuso de notación con k, que será una constante que englobe la suma de constantes que aparezcan en cada paso.
- Cada vez que en el margen de una ecuación aparezca un número entre paréntesis, quiere decir que en ese paso se utilizó la regla correspondiente (de las enumeradas arriba).
- Siempre que se hable de n será en referencia a la longitud de la secuencia s de entrada de cada función, es decir $n = |s|$ y cada vez que aparezca s con un subíndice, se hará referencia a un elemento de s. En particular s_i es el i-ésimo elemento de la secuencia.
- La función f y la operación $\oplus$ se utilizan indistintamente: $f(x, y) = x \oplus y$.

2 Primera implementación: Listas

2.1 filterS

```
filterS f [] = []
filterS f (x:xs)
    | valid     = x:ys
    | otherwise = ys
  where (valid, ys) = f x ||| filterS f xs
```

2.1.1 Trabajo

El primer argumento del trabajo es la función, y el segundo la longitud de la secuencia. En el caso general, ambas guardas tienen trabajo constante (considerando precalculados x e ys), por lo que se considera el trabajo de calcular el where más un k.

$$
\begin{aligned}
W_{filterS}(f,0) &= c_1 \\
W_{filterS}(f,n) &= W(\text{let (valid,ys) = f x}|||\text{filterS f xs in } e) \\
&= W(\text{f x}|||\text{filterS f xs}) + k && (5)\text{ y } e \text{ siempre es constante} \\
&= W_{filterS}(f,n-1) + W_f(s_0) + k && (4) \\
&\leq nk + \sum_{i=0}^{n-1} W_f(s_i)
\end{aligned}
$$

Se concluye

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

2.1.2 Profundidad

$$
\begin{aligned}
S_{filterS}(f,0) &= c_1 \\
S_{filterS}(f,n) &= S(\text{let (valid,ys) = f x}|||\text{filterS f xs in } e) \\
&= S(\text{f x}|||\text{filterS f xs}) + k && (5)\text{ y } e \text{ siempre es constante} \\
&= \max(S_{filterS}(f,n-1), S_f(s_0)) + k && (4) \\
&= \max(S_f(s_0), k + \max(S_f(s_1), ..., k + \max(S_f(s_n), c_1)...)
\end{aligned}
$$

Sea $S_f = \max\limits_{0 \leq i < n} S_f(s_i)$. Luego:

$$
\begin{aligned}
S_{filterS}(f,n) &\leq \max(S_f, k + \max(S_f, ..., k + \max(S_f, c_1)...) \\
&\leq nk + S_f && \text{Si suponemos } S_f > c_1
\end{aligned}
$$

Se concluye:

$$S_{filterS} \in O\left(n + \max_{0 \leq i < n} S_f(s_i)\right)$$

2.2 showtS

```
showtS []    = EMPTY
showtS [x]   = ELT x
showtS xs    =
    let len     = quot (lengthS xs) 2
        (l, r) = takeS xs len ||| dropS xs len
    in  NODE l r
```

2.2.1 Trabajo

Los primeros dos casos consisten en pattern matching y constructores de TreeView, por lo que tienen trabajo constante. En el tercer caso, calcular len tiene un trabajo del orden de $O(n)$ producto de usar lengthS. Calcular (l, r) tiene un trabajo del mismo orden, por usar takeS y dropS (el trabajo de ambos está en dicho orden y al estar conectados por |||, éste se suma). La última línea sólo utiliza un constructor y resultados precalculados, por lo que su trabajo es constante.

Se toma el peor caso, es decir el tercero, y se concluye que $W_{showtS} \in O(n)$.

2.2.2 Profundidad

De forma análoga al trabajo, se ve que los primeros dos casos tienen profundidad constante y que en el tercero, calcular len tiene una profundidad del orden de $O(n)$ (pues $S_{lengthS} \in O(n)$). El cálculo de (l,r) tiene la misma profundidad pues es $\max(S_{takeS}, S_{dropS})$ y ambos son $O(n)$ en profundidad. Por las mismas razones que en el análisis del trabajo, la última línea presenta profundidad constante.

Se toma el peor caso para ver que $S_{showtS} \in O(n)$.

2.3 contractL

```
contractL f []          = []
contractL f [x]         = [x]
contractL f (x1:x2:xs) =
    let (y,ys) = f x1 x2 ||| contractL f xs
    in  y:ys
```

Si bien no se pide explícitamente el análisis de contract o expand, son pilares esenciales de las dos últimas funciones de cada sección. Es por eso necesario conocer su trabajo y profundidad.

2.3.1 Trabajo

El primer argumento del trabajo es la función, y el segundo la longitud de la secuencia.

$$
\begin{aligned}
&W_{contractL}(f,0) = c_1 \\
&W_{contractL}(f,1) = c_2 \\
&W_{contractL}(f,n) = W(\text{let (y,ys) = f x1 x2}|||\text{contractL f xs in y:ys}) \\
&\qquad = W(\text{f x1 x2}|||\text{contractL f xs}) + k && (5) \text{ y } W_{(:)} \in O(1) \\
&\qquad = W_{contractL}(f, n-2) + W_f(s_0,\ s_1) + k && (4) \\
&\qquad \leq nk + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} W(x \oplus y)
\end{aligned}
$$

La desigualdad ocurre porque el conjunto aplicaciones de $\oplus$ en una ejecución de contractL es un subconjunto de $\mathcal{O}_r(\oplus, b, s)$. Se concluye:

$$
W_{contractL} \in O\left(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} W(x \oplus y)\right)
$$

2.3.2 Profundidad

$$
\begin{aligned}
&S_{contractL}(f,0) = c_1 \\
&S_{contractL}(f,1) = c_2 \\
&S_{contractL}(f,n) = S(\text{let (y,ys) = f x1 x2}|||\text{contractL f xs in y:ys}) \\
&\qquad = S(\text{f x1 x2}|||\text{contractL f xs}) + k && (11) \text{ y } S_{(:)} \in O(1) \\
&\qquad = \max(S_{contractL}(f, n-2), S_f(s_0,\ s_1)) + k && (10) \\
&\qquad \leq \max(S_f(s_0,s_1), k + \max(S_f(s_2,s_3), ..., k + \max(S_f(s_{n-1}, s_n), c_3)...) && c_3 = \max(c_1, c_2)
\end{aligned}
$$

Sea $S_f = \max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S_f(x \oplus y)$. Como todas las aplicaciones de $\oplus$ en contractL son elementos de $\mathcal{O}_r(\oplus, b, s)$, se tiene:

$$
\begin{aligned}
&S_{contractL}(f,n) \leq \max(S_f, k + \max(S_f, ..., k + \max(S_f, c_3)...) \\
&\qquad \leq nk + S_f && \text{Si suponemos } S_f > c_3
\end{aligned}
$$

Se concluye:

$$S_{contractL} \in O\left(n + \max_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} S(x \oplus y)\right)$$

2.4 expandL

```
expandL f [] _              = []
expandL f [x] (y:ys)         = [y]
expandL f (x1:x2:xs) (y:ys) =
    let (z, zs) = f y x1 ||| expandL f xs ys
    in  y:z:zs
```

2.4.1 Trabajo

El primer argumento del trabajo es la función, el segundo la longitud de la secuencia de entrada y el tercero la longitud de la solución parcial. Las secuencias serán, respectivamente, x e y y sus longitudes n_1 y n_2.

$$\begin{aligned}
W_{expandL}(f, 0, n_2) &= c_1 \\
W_{expandL}(f, 1, n_2) &= c_2 \\
W_{expandL}(f, n_1, n_2) &= W(\text{let (z,zs)} = \text{f y x1}|||\text{expandL f xs ys in y:z:ys}) \\
&= W(\text{f y x1}|||\text{expandL f xs ys}) + k && (5) \text{ y } W_{(:)} \in O(1) \\
&= W_{expandL}(f, n_1 - 2, n_2 - 1) + W_f(x_0, y_0) + k && (4)
\end{aligned}$$

Al ser solución parcial, $n_2 < n_1$. Es correcto entonces decir que la profundidad de la recursión no es mayor a n_1. Además, al igual que ocurría con *contractL*, el conjunto aplicaciones de $\oplus$ en una ejecución de *expandL* es un subconjunto de $\mathcal{O}_r(\oplus, b, s)$. Entonces:

$$W_{expandL}(f, n_1, n_2) \leq n_1 k + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} W(x \oplus y)$$

De lo que se obtiene, considerando $n = n_1$ (pues $n1$ es el largo de la secuencia expandida):

$$W_{expandL} \in O\left(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} W(x \oplus y)\right)$$

2.4.2 Profundidad

$$
\begin{aligned}
S_{expandL}(f,0,n_2) &= c_1 \\
S_{expandL}(f,1,n_2) &= c_2 \\
S_{expandL}(f,n_1,n_2) &= S(\text{let (z,zs) = f y x1}|||\text{expandL f xs ys in y:z:ys}) \\
&= S(\text{f y x1}|||\text{expandL f xs ys}) + k && (5) \text{ y } S_{(:)} \in O(1) \\
&= \max(S_{expandL}(f,n_1-2,n_2-1), S_f(x_0,y_0)) + k && (4) \\
&\leq \max(S_f(x_0,y_0), k + \max(S_f(x_3,y_1), ..., k + \max(S_f(x_{n_1-1},y_{n_2}),c_3)...) && c_3 = \max(c_1,c_2)
\end{aligned}
$$

Sea $S_f = \max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S_f(x \oplus y)$. Como todas las aplicaciones de $\oplus$ en *expandL* son elementos de $\mathcal{O}_r(\oplus,b,s)$ y la profundidad de la recursión no supera n_1:

$$
\begin{aligned}
S_{expandL}(f,n_1,n_2) &\leq \max(S_f, k + \max(S_f, ..., k + \max(S_f,c_3)...) \\
&\leq n_1 k + S_f && \text{Si suponemos } S_f > c_3
\end{aligned}
$$

Nuevamente, considerando $n = n_1$:

$$
S_{expandL} \in O\left(n + \max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S(x \oplus y)\right)
$$

2.5 reduceS

```
reduceS f b []  = b
reduceS f b [x] = f b x
reduceS f b xs  = reduceS f b $ contractL f xs
```

2.5.1 Trabajo

El primer argumento del trabajo es la función, el segundo el valor base y el tercero la longitud de la secuencia. Se tiene en cuenta el hecho de que $contractL(f,n)$ retorna una secuencia de longitud $\lceil \frac{n}{2} \rceil$.

$$
\begin{aligned}
W_{reduceS}(f,b,0) &= c_1 \\
W_{reduceS}(f,b,1) &= W_f(b,s_0) \\
W_{reduceS}(f,b,n) &= W_{reduceS}(f,b,contractL(f,n)) \\
&= W_{reduceS}(f,b,\left\lceil \frac{n}{2} \right\rceil) + W_{contractL}(f,n)
\end{aligned}
$$

Esta recurrencia se puede resolver por Teorema Maestro para obtener $W_{reduceS} \in O(W_{contractL}(f,n))$, es decir:

$$W_{reduceS} \in O\left(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} W(x \oplus y)\right)$$

2.5.2 Profundidad

$$\begin{aligned} S_{reduceS}(f,b,0) &= c_1 \\ S_{reduceS}(f,b,1) &= S_f(b,s_0) \\ S_{reduceS}(f,b,n) &= S_{reduceS}(f,b,contractL(f,n)) \\ &= S_{reduceS}(f,b,\left\lceil \frac{n}{2} \right\rceil) + S_{contractL}(f,n) \end{aligned}$$

Nuevamente se puede usar el Teorema Maestro para obtener $S_{reduceS} \in O(S_{contractL}(f,n))$, es decir:

$$S_{reduceS} \in O\left(n + \max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S(x \oplus y)\right)$$

2.6 scanS

```
scanS f b []  = ([], b)
scanS f b [x] = ([b], f b x)
scanS f b xs  =
    let cl        = contractL f xs
        (pl, res) = scanS f b cl
    in  (expandL f xs pl, res)
```

2.6.1 Trabajo

El primer argumento del trabajo es la función, el segundo el valor base y el tercero la longitud de la secuencia. Se tiene en cuenta el hecho de que $contractL(f,n)$ retorna una secuencia de longitud $\left\lceil \frac{n}{2} \right\rceil$ y $scanS(f,b,n)$ retorna un par cuyo primer elemento es una secuencia de longitud n.

$$
\begin{aligned}
W_{scanS}(f,b,0) &= c_1 \\
W_{scanS}(f,b,1) &= W_f(b,s_0) \\
W_{scanS}(f,b,n) &= W(\text{let } e_1, e_2 \text{ in } e_3) \\
&= W_{e_1} + W_{e_2} + We_3 + k \qquad (3),(5) \\
&= W_{scanS}(f,b,\left\lceil \frac{n}{2} \right\rceil) + W_{contractL}(f,n) + W_{expandL}(f,n,\left\lceil \frac{n}{2} \right\rceil) + k
\end{aligned}
$$

Esta recurrencia también se puede resolver por Teorema Maestro para obtener $W_{scanS} \in O(W_{contractL}(f,n) + W_{expandL}(f,n,\lceil \frac{n}{2} \rceil))$. Como ambas funciones tienen el mismo trabajo:

$$
W_{scanS} \in O\Big(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} W(x \oplus y)\Big)
$$

2.6.2 Profundidad

$$
\begin{aligned}
S_{scanS}(f,b,0) &= c_1 \\
S_{scanS}(f,b,1) &= W_f(b,s_0) \\
S_{scanS}(f,b,n) &= S(\text{let } e_1, e_2 \text{ in } e_3) \\
&= S_{e_1} + S_{e_2} + Se_3 + k \qquad (9),(10) \\
&= S_{scanS}(f,b,\left\lceil \frac{n}{2} \right\rceil) + S_{contractL}(f,n) + S_{expandL}(f,n,\left\lceil \frac{n}{2} \right\rceil) + k
\end{aligned}
$$

Por Teorema Maestro
$S_{scanS} \in O(S_{contractL}(f,n) + S_{expandL}(f,n,\lceil \frac{n}{2} \rceil))$ y como ambas funciones tienen igual profundidad:

$$
S_{scanS} \in O\Big(n + \max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S(x \oplus y)\Big)
$$

3 Segunda implementación: Arrays persistentes

3.1 mapS

```
mapS f s = tabulateS (f . nthS s) $ lengthS s
```

Tanto *lengthS* como *nthS* son $O(1)$ en trabajo y profundidad, luego *mapS* es igual a *tabulateS* en trabajo y profundidad.

3.1.1 Trabajo

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

3.1.2 Profundidad

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

3.2 filterS

```
filterS f s =
    let g x = if f x then singletonS x else emptyS
    in  joinS $ mapS g s
```

Por la definición de g y por *singletonS* y *emptyS* constantes en trabajo y profundidad, se tiene que *mapS* g s es $O(\sum_{i=0}^{n-1} W_f(s_i))$ en trabajo y $O(\max_{i=0}^{n-1} S_f(s_i))$ en profundidad. Sabiendo que *mapS* retorna una secuencia de largo n (del mismo largo que la que recibe), que el *mapS* y el *joinS* se ejecutan secuencialmente (pues el resultado del primero es el argumento del otro) y que *joinS* es $O(n)$ en trabajo (la suma de las longitudes de cada elemento de la secuencia arrojada por *mapS* es a lo sumo n) y $O(lg\ n)$ en profundidad, se puede obtener trabajo y profundidad de $filterS$ sumando respectivamente los de *mapS* g s y *joinS*.

3.2.1 Trabajo

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

$$W_{filterS} \in O(\sum_{i=0}^{n-1} W_f(s_i)) \qquad \text{(simplificando)}$$

3.2.2 Profundidad

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

3.3 showtS

```
showtS s =
    case lengthS s of
        0   -> EMPTY
        1   -> ELT $ nthS s 0
        len ->
            let half = quot len 2
            in  NODE (takeS s half) $ dropS s half
```

Todas las funciones que aparecen en esta definición de *showtS*, es decir, $lengthS, nthS, quot, takeS, dropS$ (y los constructores de TreeView) son $O(1)$ tanto en trabajo como en profundidad. Consecuentemente, *showtS* también lo es.

3.3.1 Trabajo

$$W_{showtS} \in O(1)$$

3.3.2 Profundidad

$$S_{showtS} \in O(1)$$

3.4 contractA

```
contractA f s
    | even len  = tabulateS g half
    | otherwise = tabulateS h $ half+1
  where len  = lengthS s
        half = quot len 2
        g i  = f (nthS s $ 2*i) (nthS s $ 2*i + 1)
        h i  = if i == half then (nthS s $ 2*i) else g i
```

El trabajo y la profundidad de la función h son iguales a los de g (pues el otro caso es constante). Además, por half= $\lfloor \frac{n}{2} \rfloor$ tanto half como half+1 son del orden de n. Basta entonces con analizar el caso de n par.

3.4.1 Trabajo

El primer argumento del trabajo es la función, y el segundo la longitud de la secuencia. c_0 es la constante que aparece al aplicar la definición de O sobre la especificación de *tabulateS* para lograr la desigualdad. k representa el trabajo del where (y las funciones de trabajo constante que contiene).

$$
\begin{aligned}
W_{contractA}(f,n) &= W_{tabulateS}(g, \left\lfloor \frac{n}{2} \right\rfloor) + k \\
&\leq k + c_0 \sum_{i=0}^{n/2-1} W_g(i) && \text{especificación de } tabulateS \\
&\leq k + c_0 \sum_{i=0}^{n/2-1} \big(1 + W_f(s_{2i}, s_{2i+1})\big) && \text{definición de } g \text{ y (2)} \\
&\leq k + c_0 n + c_0 \sum_{i=0}^{n/2-1} W_f(s_{2i}, s_{2i+1})
\end{aligned}
$$

El 1 aparece en la sumatoria por el costo de las operaciones aritméticas dentro de g. Al tener en cuenta que el conjunto de las aplicaciones de $\oplus$ en una ejecución de $contractA$ es un subconjunto de $\mathcal{O}_r(\oplus, b, s)$, se tiene:

$$
W_{contractA}(f,n) \leq k + c_0 n + c_0 \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} W(x \oplus y)
$$

$$
W_{contractA} \in O\Big(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} W(x \oplus y)\Big)
$$

3.4.2 Profundidad

$$
\begin{aligned}
S_{contractA}(f,n) &= S_{tabulateS}(g, \left\lfloor \frac{n}{2} \right\rfloor) + k \\
&\leq k + c_0 \max_{i=0}^{n/2-1} S_g(i) && \text{especificación de } tabulateS \\
&\leq k + c_0 \max_{i=0}^{n/2-1} \big(1 + S_f(s_{2i}, s_{2i+1})\big) && \text{definición de } g \text{ y (2)} \\
&\leq k + c_0 \max_{i=0}^{n/2-1} S_f(s_{2i}, s_{2i+1}) && k \text{ absorbe la constante del paso anterior}
\end{aligned}
$$

Nuevamente teniendo en cuenta que cada aplicación de $\oplus$ en una ejecución de $contractA$ está en $\mathcal{O}_r(\oplus, b, s)$:

$$
S_{contractA}(f,n) \leq k + c_0 \max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S(x \oplus y)
$$

$$
S_{contractA}(f,n) \in O\Big(\max_{(x \oplus y) \in \mathcal{O}_r(\oplus,b,s)} S(x \oplus y)\Big)
$$

3.5 expandA

```
expandA f s ps =
    let len = lengthS s
        g i = nthS ps $ quot i 2
        h i = if even i then g i else f (g i) (nthS s $ i-1)
    in  tabulateS h len
```

La función g es $O(1)$ en trabajo y profundidad pues está definida únicamente a partir de funciones con esta característica (esto será usado más adelante).

3.5.1 Trabajo

El primer argumento del trabajo es la función, el segundo la longitud de la secuencia de entrada y el tercero la longitud de la solución parcial. Las secuencias son, respectivamente, s y ps y sus longitudes n_1 y n_2. c_0 es la constante que aparece al aplicar la definición de O sobre la especificación de *tabulateS* para lograr la desigualdad. k representa el trabajo del let (y las funciones de trabajo constante que contiene).

$$
\begin{aligned}
W_{expandA}(f, n_1, n_2) &= W_{tabulateS}(h, n) + k \\
&\leq k + c_0 \sum_{i=0}^{n-1} W_h(i) && \text{espec. de } tabulateS \\
&\leq k + c_0 \sum_{i=0}^{n/2-1} \big(1 + W_g(2i)\big) + c_0 \sum_{i=0}^{n/2-1} \big(1 + W_f(g(2i+1), s_{2i})\big) && \text{definición de } h, (2) \\
&\leq k + c_0 \sum_{i=0}^{n/2-1} c_1 + c_0 \sum_{i=0}^{n/2-1} \big(1 + W_f(g(2i+1), s_{2i})\big) && \text{definición y costos de } g \\
&\leq k + c_2 n + c_0 \sum_{i=0}^{n/2-1} W_f(ps_i, s_{2i}) && c3 \geq (c_0 c_1 + c_0)
\end{aligned}
$$

El 1 aparece en las sumatorias por el costo de las operaciones aritméticas/control de flujo dentro de h. Al tener en cuenta que el conjunto de las aplicaciones de $\oplus$ en una ejecución de *expandA* es un subconjunto de $\mathcal{O}_r(\oplus, b, s)$, se tiene:

$$
W_{expandA}(f, n_1, n_2) \leq k + c_2 n + c_0 \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} W(x \oplus y)
$$

$$
W_{expandA} \in O\left(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} W(x \oplus y)\right)
$$

3.5.2 Profundidad

$$
\begin{aligned}
S_{expandA}(f, n_1, n_2) &= S_{tabulateS}(h, n) + k \\
&\leq k + c_0 \max_{i=0}^{n-1} S_h(i) && \text{espec. de } tabulateS \\
&\leq k + c_0 \max_{i=0}^{n/2-1} \big(1 + S_g(2i), 1 + S_f(g(2i+1), s_{2i})\big) && \text{definición de } h, (2) \\
&\leq k + c_0 + c_0 \max_{i=0}^{n/2-1} \big(S_g(2i), S_f(g(2i+1), s_{2i})\big) \\
&\leq k + c_0 \max_{i=0}^{n/2-1} \big(S_g(2i), S_f(g(2i+1), s_{2i})\big) && \text{k absorbe la constante} \\
&\leq k + c_0 \max_{i=0}^{n/2-1} \big(1, S_f(ps_i, s_{2i})\big) && \text{def. y costo de } g \\
&\leq k + c_0 \max_{i=0}^{n/2-1} S_f(ps_i, s_{2i})
\end{aligned}
$$

Nuevamente teniendo en cuenta que cada aplicación de $\oplus$ en una ejecución de *expandA* está en $\mathcal{O}_r(\oplus, b, s)$:

$$S_{expandA}(f, n_1, n_2) \leq k + c_0 \max_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} S(x \oplus y)$$

$$S_{expandA}(f, n) \in O\left(\max_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} S(x \oplus y)\right)$$

3.6 reduceS

```
reduceS f b s =
    case lengthS s of
        0 -> b
        1 -> f b $ nthS s 0
        _ -> reduceS f b $ contractA f s
```

3.6.1 Trabajo

El primer argumento del trabajo es la función, el segundo el valor base y el tercero la longitud de la secuencia. Se tiene en cuenta el hecho de que $contractA(f, n)$ retorna una secuencia de longitud $\lceil \frac{n}{2} \rceil$.

$$
\begin{aligned}
W_{reduceS}(f, b, 0) &= c_1 \\
W_{reduceS}(f, b, 1) &= W_f(b, s_0) \\
W_{reduceS}(f, b, n) &= W_{reduceS}(f, b, contractA(f, n)) \\
&= W_{reduceS}(f, b, \left\lceil \frac{n}{2} \right\rceil) + W_{contractA}(f, n)
\end{aligned}
$$

Esta recurrencia se puede resolver por Teorema Maestro para obtener $W_{reduceS} \in O(W_{contractA}(f, n))$, es decir:

$$
W_{reduceS} \in O\left(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} W(x \oplus y)\right)
$$

3.6.2 Profundidad

$$
\begin{aligned}
S_{reduceS}(f, b, 0) &= c_1 \\
S_{reduceS}(f, b, 1) &= S_f(b, s_0) \\
S_{reduceS}(f, b, n) &= S_{reduceS}(f, b, contractA(f, n)) \\
&= S_{reduceS}(f, b, \left\lceil \frac{n}{2} \right\rceil) + S_{contractA}(f, n)
\end{aligned}
$$

Como en cada paso la longitud se reduce a la mitad, la profundidad de recursión es del orden de $lg(n)$. Y como la profundidad de cada *contractA* es a lo sumo $\max_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} S(x \oplus y)$, se concluye:

$$
S_{reduceS} \in O\left(lg(n) \max_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} S(x \oplus y)\right)
$$

3.7 scanS

```
scanS f b s =
    case lengthS s of
        0 -> (emptyS, b)
        1 -> (singletonS b, f b $ nthS s 0)
        _ ->
            let cs = contractA f s
                (ps, res) = scanS f b cs
            in (expandA f s ps, res)
```

3.7.1 Trabajo

El primer argumento del trabajo es la función, el segundo el valor base y el tercero la longitud de la secuencia. Se tiene en cuenta el hecho de que $contractA(f, n)$ retorna una secuencia de longitud $\lceil \frac{n}{2} \rceil$ y $scanS(f, b, n)$ retorna un par cuyo primer elemento es una secuencia de longitud n.

$$
\begin{aligned}
W_{scanS}(f, b, 0) &= c_1 \\
W_{scanS}(f, b, 1) &= W_f(b, s_0) \\
W_{scanS}(f, b, n) &= W(\text{let } e_1, e_2 \text{ in } e_3) \\
&= W_{e_1} + W_{e_2} + We_3 + k \qquad (3), (5) \\
&= W_{scanS}(f, b, \left\lceil \frac{n}{2} \right\rceil) + W_{contractL}(f, n) + W_{expandL}(f, n, \left\lceil \frac{n}{2} \right\rceil) + k
\end{aligned}
$$

Esta recurrencia también se puede resolver por Teorema Maestro para obtener $W_{scanS} \in O(W_{contractL}(f, n) + W_{expandL}(f, n, \lceil \frac{n}{2} \rceil))$. Como ambas funciones tienen el mismo trabajo:

$$
W_{scanS} \in O\left(n + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} W(x \oplus y)\right)
$$

3.7.2 Profundidad

$$
\begin{aligned}
S_{scanS}(f, b, 0) &= c_1 \\
S_{scanS}(f, b, 1) &= W_f(b, s_0) \\
S_{scanS}(f, b, n) &= S(\text{let } e_1, e_2 \text{ in } e_3) \\
&= S_{e_1} + S_{e_2} + Se_3 + k \qquad (9), (10) \\
&= S_{scanS}(f, b, \left\lceil \frac{n}{2} \right\rceil) + S_{contractL}(f, n) + S_{expandL}(f, n, \left\lceil \frac{n}{2} \right\rceil) + k
\end{aligned}
$$

En cada paso se reduce la longitud a la mitad, luego la profundidad de recursión es $lg(n)$. La profundidad de $contractL$ es igual a la de $expandL$, luego en cada nivel la profundidad es igual a la de estas funciones. Resulta:

$$
S_{scanS} \in O\left(lg(n) \max_{(x \oplus y) \in \mathcal{O}_r(\oplus, b, s)} S(x \oplus y)\right)
$$