Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Estructuras de datos y algoritmos 2

Trabajo Práctico 2

Autor:
Tomás Castro Rojas
Blas Barbagelata

13 de junio de 2021

Índice

1. Implementación de secuencias para listas

1.1. mapS

La definición de la función `mapS` es la siguiente:

```
mapList f [] = emptyList
mapList f (x:xs) = let (y, ys) = f x ||| mapList f xs
                   in y:ys
```

Trabajo

Dada la definición de la función, tenemos los siguientes trabajos:
$W_{mapS}(f, 0) = c_0$
$W_{mapS}(f, n) = c_1 + W_f(x_0) + W_{mapS}(f, n-1)$
donde c_1 es el factor constante de `let` y `cons`, $W_f(x_0)$ es el trabajo de la llamada a izquierda y $W_{mapS}(f, n-1)$ es el trabajo de la llamada a derecha.
Vamos a demostrar que

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

Demostración:
(HI) Suponemos que todo $k \leq n$, $\exists c' \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N}$ / $\forall n \geq n_0 : 0 \leq mapS(f, n) \leq c' * \sum_{i=0}^{n-1} W_f(i)$

(Caso inductivo) Vamos a probar para $n + 1$

$$W_{mapS}(f, n+1) = c_1 + W_f(x_0) + W_{mapS}(f, n) \leq^{(HI)} c_1 + W_f(x_0) + c' * \sum_{i=0}^{n-1} W_f(i) \leq^{(1)}$$

$$\leq c''W_f(x_0) + c' \sum_{i=0}^{n-1} W_f(i) \leq^{(2)} c * W_f(x_0) + c * \sum_{i=0}^{n-1} W_f(i) = c * \sum_{i=0}^{n} W_f(i)$$

(HI) Hipótesis inductiva
(1) $c''W_f(x_0) \geq c_1 + W_f(x_0) \Leftrightarrow c'' \geq \frac{c_1}{W_f(x_0)} + 1$ como $W_f(x_0) \geq 1$
(2) $c = max\{c', c''\}$

Profundidad

Dada la definición de la función, tenemos las siguientes profundidades:
$S_{mapS}(0) = k_0$
$S_{mapS}(n) = k_1 + max\{S_f(x_0), S_{mapS}(n-1)\}$
donde k_1 es la profundidad del uso de `let` y `cons`, y f es la funcion que se aplica a cada elemento de la lista.
Vamos a desmostrar que

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

(HI) Suponemos que todo $k \leq n$, $\exists c' \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N}$ / $\forall n \geq n_0 : 0 \leq mapS(f, n) \leq c' * (\max_{i=0}^{n-1} S_f(i) + n)$
(Caso Inductivo)

$$mapS(n+1) = k_1 + \max\{S_f(x_0), mapS(n)\} \leq^{(HI)} k_1 + \max\{S_f(x_0), c' * (\max_{i=0}^{n-1} S_f(i) + n)\} \leq^{(1)}$$

$$\leq k_1 + c' * (\max_{i=0}^{n-1} S_f(i) + n) \leq c + cn + c(\max_{i=0}^{n-1} S_f(i)) = c(\max_{i=0}^{n-1} S_f(i) + (n+1)) \leq^{(2)} c(\max_{i=0}^{n} S_f(i) + (n+1))$$

(1) Aca se podrian dar dos casos equivalentes:

- $\max\{S_f(x_0), c' * (\max_{i=0}^{n-1} S_f(i) + n)\} = S_f(x_0)$. Este caso esta incluido en el $\max_{i=0}^{n-1} S_f(i)$ y además acotado por el factor lineal que se le suma.
- $\max\{S_f(x_0), c' * (\max_{i=0}^{n-1} S_f(i) + n)\} = c' * (\max_{i=0}^{n-1} S_f(i) + n)$. Este caso estaria acotado por si mismo trivialmente.

(2) $\max_{i=0}^{n-1} S_f(i) \leq \max_{i=0}^{n} S_f(i)$ ya que, o bien, el $S_f(x_n)$ es el nuevo máximo y por lo tanto claramente mayor al máximo del conjunto sin él, o el máximo del conjunto no se altera agregando $S_f(x_n)$.
$\therefore S_{mapS} \in O(\max_{i=0}^{n-1} S_f(i) + n)$

1.2. appendS

Tenemos la siguiente implementación de la función `appendS`:

```
appendList [] sb     = sb
appendList sa []     = sa
appendList (a:sa) sb = a:(appendList sa sb)
```

Trabajo

Para el calculo del trabajo de la función, definimos la recurrencia respecto al largo de la primer lista. Entonces nos queda:
$W_{appendS}(0) = c_0$
$W_{appendS}(n) = c_1 + W_{appendS}(n-1)$

Vamos mostrar que

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

(HI) Suponemos que $k \leq n$, $\exists c' \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N} /\ \forall n \geq n_0 : 0 \leq W_{appendS}(n) \leq c'n$ (Caso Inductivo)

$$W_{appendS}(n+1) = W_{appendS}(n) + c_1 \leq^{(HI)} c'n + c_1 \leq^{(1)} c'(n+1)$$

(1) Tomando $c = \text{máx}\{c', c_1\}$
$\therefore W_{appendS} \in O(n)$.

Profundidad

Respecto a la profundidad de esta funcion, no se implementa ningun tipo de paralelizacion. Observando las ecuaciones de profundidad en relación al largo de la primer lista:
$S_{appendS}(0) = k_0$
$S_{appendS}(n) = k_1 + S_{appendS}(n-1)$

Observamos que son iguales a la recurrencia del Trabajo, donde ya se demostró que estaba acotado por $O(n)$. Por lo tanto, $S_{appendS} \in O(n)$.

1.3. reduceS

```
reduceList _ e []  = e
reduceList f e [x] = f e x
reduceList f e xs  = let ctr = contraerList f xs
                     in reduceList f e ctr
```

donde `contraerList` esta definido como:

```
contraerList _ []         = emptyList
contraerList _ l@[x]      = l
contraerList f (x:y:xs) = let (z,zs) = f x y ||| contraerList f xs
                          in z:zs
```

Asumimos que $W_f \in O(1)$ y $S_f \in O(1)$.

Trabajo

Ahora definimos la recurrencia de $W_{reduceS}$ respecto a la longitud de la lista:
$W_{reduceS}(0) = c_0$
$W_{reduceS}(1) = W_f(e, x_0) = c_f$
$W_{reduceS}(n) = W_{contraerList}(n) + W_{reduceS}(\lceil \frac{n}{2} \rceil) + c_3$

Primero veamos el Trabajo de `contraerList`. Dada la definición, tenemos las siguientes ecuaciones en relación a la longitud de la lista:
$W_{contraerList}(0) = c_3$
$W_{contraerList}(1) = c_4$
$W_{contraerList}(n) = c_5 + W_f(x_0, x_1) + W_{contraerList}(n-2) = c_5 + c_f + W_{contraerList}(n-2)$
Vamos a demostrar que

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

(HI) Suponemos que para todo $k \leq n$, $\exists c' \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N} / \forall n \geq n_0 : 0 \leq contraerList(n) \leq c'n$
(Caso Inductivo) Probamos para $n+1$

$$contraerList(n+1) = c_5 + c_f + contraerList(n-1) \leq^{(HI)} c_5 + c_f + c'(n-1) = c_5 + c_f + c'n - c' \leq^{(1)}$$
$$\leq c + c + cn - c = c(n+1)$$

(1) $c = \text{máx}\{c_5, c_f, c'\}$
$\therefore W_{contraerList} \in O(n)$.

Ahora, supongamos que $n = 2^k$.
Entonces `reduceS` quedaría: $W_{reduceS}(n) = W_{contraerList}(n) + W_{reduceS}(\lceil \frac{n}{2} \rceil) + c_3 = W_{reduceS}(n) = W_{contraerList}(n) + W_{reduceS}(\frac{n}{2}) + c_3$
Observando esta recurrencia, podríamos aplicar el 3^{er}caso del Teorema Maestro. Primero demostramos que $W_{contraerList} \in \Omega(n)$.
(HI) Suponemos que para todo $k \leq n$, $\exists c' \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N} /\ \forall n \geq n_0 : 0 \leq c'n \leq contraerList(n)$
(Caso Inductivo)

$$contraerList(n+1) = contraerList(n-1) + c_f + c_5 \geq^{(HI)} c'(n-1) + c_f + c_5 \geq^{(1)}$$

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

(1) $c'' = \text{mín}\{c', c_f, c_5\}$
Ahora si, tomando $a = 1$, $b = 2$, $\epsilon = 1$, $c = \frac{1}{2}$ y $N > 1$ tenemos:

$$W_{contraerList} \in \Omega(n^{\log_2(1+1)}) = \Omega(n^{\log_2 2}) = \Omega(n)$$

$$W_{contraerList}(\frac{n}{2}) = \frac{n}{2} = \frac{1}{2} * n = c * W_{contraerList}(n)$$

Luego, por el tercer caso del Teorema Maestro, podemos afirmar que $W_{reduceS}(2^k) \in \Theta(W_{contraerList}(2^k)) = \Theta(2^k)$.

Al ser la $f(n) = n$ suave y $W_{reduceS}$ no decreciente, por Regla de suavidad, concluimos que $W_{reduceS} \in \Theta(n)$

Profundidad

Para calcular la profundidad, tenemos las siguientes ecuaciones:
$S_{reduceS}(0) = k_0$
$S_{reduceS}(1) = S_f(e, x_0) = k_1$
$S_{reduceS}(n) = S_{contraerList}(n) + S_{reduceS}(\lceil \frac{n}{2} \rceil) + k_2$

Del mismo modo que procedimos con el Trabajo, primero analizamos la profundidad de `contraerList`. Tenemos las siguientes recurrencias para esta función:
$S_{contraerList}(0) = k_3$
$S_{contraerList}(1) = k_4$
$S_{contraerList}(n) = k_5 + \text{máx}\{S_f(x_0, x_1), S_{contraerList}(n-2)\} = k_5 + \text{máx}\{k_f, S_{contraerList}(n-2)\}$
Vamos a demostrar que

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

$S_{contraerList} \in O(n)$
(HI) Suponemos que para todo $k \leq n$, $\exists c_1 \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N}/$
$\forall n \geq n_0 : 0 \leq contraerList(n) \leq c_1 n$
(Caso Inductivo)
$contraerList(n+1) = k_5 + \text{máx}\{c_f, contraerList(n-1)\} \leq^{(HI)} k_5 + \text{máx}\{c_f, c_1(n-1)\} \leq^{(1)} k_5 + c_1(n-1) \leq^{(2)} c + c(n-1) = cn \leq c(n+1)$
(1) Siempre se puede tomar $c_1 > c_f$
(2) $c = \text{máx}\{c_1, k_5\}$
$n \in O(S_{contraerList})$
(HI) Suponemos que para todo $k \leq n$, $\exists c_2 \in \mathbb{R}^+$ y $\exists n_0 \in \mathbb{N}/$
$\forall n \geq n_0 : 0 \leq n \leq c_2 S_{contraerList}(n)$
(Caso Inductivo)
$(n+1) = 1 + n \leq^{(HI)} 1 + c_2 S_{contraerList}(n) = 1 + c_2(k_5 + \text{máx}\{k_f, S_{contraerList}(n-2)\}) = 1 + c_2.k_5 + c_2 \text{máx}\{k_f, S_{contraerList}(n-2)\} \leq 1 + c_2.k_5 + c_2 S_{contraerList}(n-2) \leq^{(3)} c' + c_2 S_{contraerList}(n-2) \leq^{(4)} c'' + c'' S_{contraerList}(n-2) = c''(S_{contraerList}(n-2)+1) \leq^{(5)} c''(S_{contraerList}(n+1)+1) \leq^{(6)} cS_{contraerList}(n+1)$
(3) $c' = 1 + c_2.k_5$
(4) $c'' = max\{c_2, c'\}$
(5) $S_{contraerList}$ es una función de profundidad entonces es no decreciente
(6) Vamos a probar el siguiente resultado. $\forall n \in \mathbb{N}, \exists c \in \mathbb{R}^+ / c(n+1) \leq 2cn$.

(Caso Base)$n = 1$
$c(n+1) = c(1+1) = c2 = 2c * 1 = 2cn \Rightarrow c(n+1) \leq 2cn$
(Caso Inductivo) Para n, $c(n+1) \leq 2cn$. Probamos para $n+1$.
$c((n+1)+1) = cn + 2c = c(n+1) + c \leq 2cn + c = 2c(n+1)$

Por lo tanto, tomando $c \geq 2c'$ se cumple la desigualdad.
$\therefore S_{contraerList} \in \Theta(n)$
Como en `reduceS` no ocurre ninguna paralelización y al tener la misma recurrencia, realizando el procedimiento anterior de aplicar el tercer caso del Teorema Maestro y Regla de suavidad. podemos concluir nuevamente que

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

1.4. scanS

```
scanList _ e [] = (emptyList , e)
scanList f e [x] = (singletonList e, f e x)
scanList f e xs = let (ys, r) = scanList f e (contraerList f xs)
                  in (expandirList f xs ys, r)
  where
   expandirList _ [] _ = []
   expandirList _ [_] ys = ys
   expandirList f (x:_:xs) (y:ys) = let (z, zs) = (f y x) ||| expandirList f xs ys
                                    in y:z:zs
```

Asumimos que $W_f \in O(1)$ y $S_f \in O(1)$.

Trabajo

La funcion `scanS` va iterar sobre la longitud de la lista, llamamos n al largo de la lista. Entonces nos quedan las siguientes ecuaciones de trabajo:
$W_{scanS}(0) = c_0$
$W_{scanS}(1) = c_1$
$W_{scanS}(n) = c_2 + W_{contraerList}(n) + W_{scanS}(\lceil \frac{n}{2} \rceil) + W_{expandirList}(n)$

Para determinar $W_{scanList}$ hay que calcular $W_{contraerList}$ y $W_{expandirList}$. $W_{contraerList}$ ya fue calculado en el apartado anterior y demostramos que $W_{contraerList} \in \Theta(n)$. Pasamos a calcular $W_{expandirList}$. Esta funcion itera sobre la longitud de la primer lista, tenemos los siguientes trabajos:
$W_{expandirList}(0) = c_3$
$W_{expandirList}(1) = c_4$
$W_{expandirList}(m) = c_5 + W_f(x_0, y_0) + W_{expandirList}(m-2) = c_5' + W_{expandirList}(m-2)$

Podemos ver que esta recurrencia es igual a la recurrencia de $W_{contraerList}$. Entonces podemos afirmar que $W_{expandirList} \in \Theta(n)$.

Nuevamente, supongamos una entrada $n = 2^k$.
Entonces $W_{scanS}(n)$ quedaría: $c_2 + W_{contraerList}(n) + W_{scanList}(\frac{n}{2}) + W_{expandirList}(n)$.
Luego, tomando $a = 1$, $b = 2$, $c = \frac{1}{2}$ y $N > 1$ tenemos:

$$W_{contraerList} + W_{expandirList} \in \Omega(n^{\log_2(1+1)}) = \Omega(n^{\log_2 2}) = \Omega(n)$$

$$W_{contraerList}(\frac{n}{2}) + W_{expandirList}(\frac{n}{2}) = \frac{n}{2} + \frac{n}{2} = \frac{1}{2} * (n+n) = c * (W_{contraerList}(n) + W_{expandirList}(n))$$

Entonces, por 3^{er} caso del Teorema Maestro, podemos concluir que $W_{scanS}(2^k) \in \Theta(W_{contraerList}(2^k) + W_{expandirList}(2^k)) = \Theta(2^k)$.

Al ser la $f(n) = n$ suave y W_{scanS} no decreciente, por Regla de suavidad, concluimos que $W_{scanS} \in \Theta(n)$

Profundidad

La recurrencia para la profundidad de `scanS` es la siguiente:
$S_{scanS}(0) = k_0$
$S_{scanS}(1) = k_1$
$S_{scanS}(n) = k_2 + S_{contraerList}(n) + S_{scanS}(\lceil \frac{n}{2} \rceil) + S_{expandirList}(n)$

Podemos ver que no ocurre ningún tipo de paralelización. Además, ya vimos que $S_{contraerList} \in \Theta(n)$. Con lo cual quedaría demostrar que $S_{expandirList} \in \Theta(n)$ y con eso aplicar nuevamente el tercer caso del Teorema Maestro.
La recurrencia de $S_{expandirList}$ es la siguiente:
$S_{expandirList}(0) = k_3$
$S_{expandirList}(1) = k_4$
$S_{expandirList}(m) = k_5 + \text{máx}\{S_f, S_{expandirList}(m-2)\}$

Es fácil ver que esta recurrencia es igual a la recurrencia de $S_{contraerList}$, por lo tanto podemos asegurar realizando el mismo procedimiento y utilizando las mismas justificaciones que $S_{expandirList} \in \Theta(n)$.
Entonces, tomando los mismo valores que en S_{scanS} para a, b, c, ϵ y N en el tercer caso del Teorema Maestro y por Regla de suavidad, concluimos que $S_{scanS} \in \Theta(n)$

2. Implementación de secuencias para arreglos

2.1. mapS

Tenemos la siguiente implementación de la función `mapS`:

```
mapArr f arr = let largo = lengthArr arr
               in  tabulateArr (\i -> (f (nthArr arr i))) largo
```

Trabajo

Podemos ver que dada la definición de la función, tenemos los siguientes trabajos:

$$W_{mapArr}(f,n) = W_{lengthArr}(n) + W_{tabulateArr}(f',n) = c_1 + O(\sum_{i=0}^{n-1} W_{f'}(i))$$

Donde n es el largo de la secuencia, c_1 es el factor constante de `let`. Además definimos $f' = (\lambda i \to (f(nthArr\ arr\ i)))$ Vamos a demostrar que

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

Demostración:

$$W_{f'}(f,i) = W_f(x_i) + W_{nthArr}(i) = W_f(x_i) + c_2 = O(\max(W_f(x_i), 1)) = W_f(x_i)$$

Ya que el mínimo trabajo que puede realizar f es 1. Entonces $W_{f'}(f,i) \in O(W_f(x_i))$. Por lo que nos queda:

$$O(1) + O(\sum_{i=0}^{n-1} W_{f'}(i)) = O(1) + O(\sum_{i=0}^{n-1} W_f(i)) = \max(O(1), O(\sum_{i=0}^{n-1} W_f(i))) = O(\sum_{i=0}^{n-1} W_f(i))$$

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

Profundidad

Para calcular la profundidad tenemos:

$$S_{mapArr}(f,n) = S_{lengthArr}(n) + S_{tabulateArr}(f',n) = c_1 + O(\max_{i=0}^{n-1} S_{f'}(x_i))$$

Donde f' es la función definida anteriormente. Se puede plantear su profundidad como:

$$S_{f'}(f,i) = S_f(x_i) + S_{nthArr}(i) = S_f(x_i) + c_1 = O(\max(S_f(x_i), 1)) = S_f(x_i)$$

Entonces $S_{f'}(f, i) \in O(S_f(x_i))$ De esta forma obtenemos:

$$c_1 + O(\max_{i=0}^{n-1} S_f(x_i)) = O(\max(1, \max_{i=0}^{n-1} S_f(x_i))) = O(\max_{i=0}^{n-1} S_f(x_i))$$

$$\therefore S_{mapArr}(f, n) \in O(\max_{i=0}^{n-1} S_f(x_i))$$

2.2. appendS

Tenemos la siguiente implementación de la función `appendS`:

```
appendArr arr brr = let n = lengthArr arr
                        m = lengthArr brr
                    in tabulateArr copiar (n+m)
                  where
                    copiar i | i < n = nthArr arr i
                             | otherwise = nthArr brr (i-n)
```

Trabajo

Sean n,m el tamaño de las 2 secuencias a unir. Planteamos el trabajo de la función de la siguiente forma:

$$W_{appendS}(n,m) = 2.W_{lengthArr} + W_{tabulateArr}(copiar, n+m) = 2.c_1 + O(\sum_{i=0}^{n+m-1} W_{copiar}(x_i))$$

Donde el trabajo de copiar es:

$$W_{copiar} = W_{nthArr}(i) = O(1)$$

entonces el calculo anterior queda de la siguiente forma:

$$2.O(1) + \sum_{i=0}^{n+m-1} W_{copiar}(x_i) = 2.O(1) + O(\sum_{i=0}^{n+m-1} O(1)) = 2.O(1) + O(n+m) =$$

$$O(max(1, n+m)) = O(n+m)$$

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

Profundidad

Para calcular la profundidad tenemos:

$$S_{appendS}(n,m) = 2.S_{lengthArr} + S_{tabulateArr}(copiar, n+m) = 2.c_1 + O(\max_{i=0}^{n+m-1} S_{copiar}(x_i))$$

Donde la profundidad de copiar es:

$$S_{copiar} = S_{nthArr}(i) = O(1)$$

$$2.O(1) + \max_{i=0}^{n+m-1} S_{copiar}(x_i) = 2.O(1) + O(\max_{i=0}^{n+m-1} O(1)) = 2.O(1) + O(1) = 3.O(1) = O(1)$$

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

2.3. reduceS

Dada la siguiente definicion de reduceArr:

```
reduceArr f e arr | largo == 0  = e
                  | largo == 1  = f e (nthArr arr 0)
                  | otherwise = let ctr = contraerArr f arr largo
                                in reduceArr f e ctr
                  where largo = lengthArr arr
```

donde `contraerArr` esta definido como:

```
contraerArr f arr largo | even largo = tabulateArr contraerPar rango
                        | otherwise  = tabulateArr contraerImpar (rango+1)
    where
      rango = (div largo 2)
      contraerPar i = f (nthArr arr (2*i)) (nthArr arr ((2*i)+1))
      contraerImpar i | i == rango =  nthArr arr (2*i)
                      | otherwise = contraerPar i
```

Trabajo

Sea n el largo de la secuencia, f la función a aplicar. Tomando en cuenta que $f \in O(1)$, para poder analizar su costo de forma correcta primero debemos analizar el trabajo de la funcion `contraerArr`:

$$W_{contraerArr}(f, n) = W_{tabulateArr}(f', n) = \Theta(\sum_{i=0}^{n/2} W_{f'}(x_i))$$

Dependiendo de la paridad del argumento n, f' puede terner dos formas:

- En el caso de que n sea par $\rightarrow f' = contraerPar$
- En el caso de que n sea impar $\rightarrow f' = contraerImpar$

Por lo dicho anteriormente f tiene trabajo constante. Como `nthS` también tiene trabajo constante, podemos decir que el trabajo de f' es constante sin importar que forma adopta. Entonces $W_{f'} \in \Theta(1)$.
Continuando con el trabajo de `contraerArr`:

$$\Theta(\sum_{i=0}^{n/2} W_{f'}(x_i)) = \Theta(\sum_{i=0}^{n/2} \Theta(1)) = \Theta(n/2)$$

Sabemos que $n/2 \in \Theta(n) \Rightarrow W_{contraerArr}(f, n) \in \Theta(n)$ Ahora estamos en condiciones de analizar el costo de `reduceArr`:

$W_{reduceArr}(f, 0) = c_0$
$W_{reduceArr}(f, 1) = W_f(x_i) + W_{nthArr}(1) + c_1$
$W_{reduceArr}(f, n) = W_{contraerArr}(f, n) + W_{reduceArr}(f, \lceil n/2 \rceil) + c_2$

Siendo c_0 y c_1 costos de arranque y c_2 el valor constante del trabajo de $W_{lenghtArr}$ Para utilizar el teorema de suavidad, supongamos que n = 2^k. Con esto podemos reescribir los costos como:
$W_{reduceArr}(f, 0) = c_0$
$W_{reduceArr}(f, 1) = W_f(x_i) + W_{nthArr}(1) + c_1$
$W_{reduceArr}(f, n) = W_{contraerArr}(f, n) + W_{reduceArr}(f, n/2) + c_2$

Utilizando la tabla de costos dada en la materia, se obtiene:
$W_{reduceArr}(f, 1) = O(1) + (1) + c_1 = 2.O(1) + c_1 = O(1)$
Si $n > 1$, se sabe que $W_{contraerArr}(f, n) \in \Theta(n)$. Ahora si, tomando $a = 1$, $b = 2$, $\epsilon = 1$, $c = \frac{1}{2}$ y $N > 1$ tenemos:

$$W_{contraerArr} \in \Omega(n^{\log_2(1+1)}) = \Omega(n^{\log_2 2}) = \Omega(n)$$

$$W_{contraerArr}(\frac{n}{2}) = \frac{n}{2} = \frac{1}{2} * n = c * W_{contraerArr}(n)$$

Luego, por el tercer caso del Teorema Maestro, podemos afirmar que $W_{reduceS}(2^k) \in \Theta(W_{contraerArr}(2^k)) = \Theta(2^k)$.

Al ser la función $f(n) = n$ suave y $W_{reduceS}$ no decreciente, por Regla de suavidad, concluimos que $W_{reduceS} \in \Theta(n)$

Profundidad

La profundidad de `contraerArr` tiene la siguiente forma:

$$S_{contraerArr}(f, n) = S_{tabulateArr}(f', n) = \Theta(\max_{i=0}^{n/2} S_{f'}(x_i))$$

Al igual que en el trabajo calculado anteriormente f' va a tomar dos formas distintas según la paridad de n. La profundidad de f y `nthArr` son constantes, por lo que la profundidad de f' es constante. Entonces $S_{f'} \in \Theta(1)$. Retomando el calculo de `contraerArr`:

$$\Theta(\max_{i=0}^{n/2} S_{f'}(x_i)) = \Theta(\max_{i=0}^{n/2} \Theta(1)) = \Theta(1)$$

Por lo que podemos concluir que $S_{contraerArr}(f, n) \in \Theta(1)$. Nos queda analizar la profundidad de `reduceArr`:

$S_{reduceArr}(f, 0) = c_0$
$S_{reduceArr}(f, 1) = S_f(x_i) + S_{nthArr}(1) + c_1$
$S_{reduceArr}(f, n) = S_{contraerArr}(f, n) + S_{reduceArr}(f, \lceil n/2 \rceil) + c_2$

Siendo c_0 y c_1 costos de arranque y c_2 el valor constante del trabajo de $S_{lenghtArr}$ Para utilizar el teorema de suavidad, supongamos que n = 2^k. Con esto podemos reescribir los costos como:
$S_{reduceArr}(f, 0) = c_0$
$S_{reduceArr}(f, 1) = S_f(x_i) + S_{nthArr}(1) + c_1$
$S_{reduceArr}(f, n) = S_{contraerArr}(f, n) + S_{reduceArr}(f, n/2) + c_2$

Utilizando la tabla de costos dada en la materia, se obtiene:
$S_{reduceArr}(f, 1) = O(1) + (1) + c_1 = 2.O(1) + c_1 = O(1)$
Si $n > 1$, se sabe que $S_{contraerArr}(f, n) \in \Theta(1)$. Ahora si, tomando $a = 1$, $b = 2$, tenemos:

$$S_{contraerArr} \in \Theta(n^{\log_2(1)}) = \Theta(n^0) = \Theta(1)$$

Luego, por el segundo caso del Teorema Maestro, podemos afirmar que $S_{reduceS}(2^k) \in \Theta((2^k)^0 \log 2^k) = \Theta(k)$.

Al ser la función $f(n) = \log n$ suave y $S_{reduceS}$ no decreciente, por Regla de suavidad, concluimos que $S_{reduceS} \in \Theta(\log n)$

2.4. scanS

Tenemos la siguiente definicion de la función:

```
scanArr f e arr | largo == 0 = (emptyArr, e)
                | largo == 1 = (singletonArr e, f e (nthArr arr 0))
                | otherwise = let ctr = contraerArr f arr largo
                                  (brr, r) = scanArr f e ctr
                              in (expandirArr f arr brr, r)
where
  largo = lengthArr arr
  expandirArr f arr brr =
  tabulateArr (\i -> if even i then (nthArr brr (div i 2))
                     else f (nthArr brr (div i 2)) (nthArr arr (i - 1))) largo
```

Trabajo

Dado un n (tamaño de la secuencia) y f la función a aplicar. Tomando en cuenta que $f \in O(1)$. En el análisis de `reduceArr` ya calculamos los costos de `contraerArr`, así que procedemos a analizar el costo de `expandirArr`:

$$W_{expand}(f,n) = W_{lengthArr}(n) + W_{tabulateArr}(f',n) = \Theta(1) + \Theta(\sum_{i=0}^{n-1} W_{f'}(x_i))$$

Donde la funcion f' es:

$$(\lambda i \rightarrow \text{if even i then (nthArr brr (div i 2))}$$
$$\text{else f (nthArr brr (div i 2)) (nthArr arr (i - 1))}$$

Como el trabajo de todas las funciones involucradas en f' son constantes, se puede concluir que $W_{f'} \in \Theta(1)$. Siguiendo con el calculo anterior:

$$\Theta(1) + \Theta(\sum_{i=0}^{n-1} W_{f'}(x_i)) = \Theta(1) + \Theta(\sum_{i=0}^{n-1} \Theta(1)) = \Theta(1) + \Theta(n) = \Theta(max(1,n)) = \Theta(n)$$

Por lo tanto $W_{expandirArr}(f,n) \in \Theta(n)$.
Ahora estamos en condiciones de calcular el trabajo de scanS:
$W_{scanS}(f,0) = W_{singletonArr}(1) + c_0$
$W_{scanS}(f,1) = W_{singletonArr}(1) + W_f(x_i) + W_{nthArr}(0) + c_1$
$W_{scanS}(f,n) = W_{contraerArr}(f,n) + W_{scanS}(f,\lceil n/2 \rceil) + W_{expandirArr}(f,n) + c_2$

Siendo c_0 y c_1 costos de arranque y c_2 el valor constante del trabajo de $W_{lenghtArr}$.
Para utilizar el teorema de suavidad, supongamos que n = 2^k. Con esto podemos reescribir

los costos como:
$W_{scanS}(f,0) = W_{singletonArr}(1) + c_0$
$W_{scanS}(f,1) = W_{singletonArr}(1) + W_f(x_i) + W_{nthArr}(0) + c_1$
$W_{scanS}(f,n) = W_{contraerArr}(f,n) + W_{scanS}(f,n/2) + W_{expandirArr}(f,n) + c_2$

Utilizando la tabla de costos dada en la materia, se obtiene:
$W_{scanS}(f,0) = O(1) + c_0 = O(1)$
$W_{scanS}(f,1) = O(1) + (1) + O(1) + c_1 = 3.O(1) + c_1 = O(1)$
Entonces:
$W_{scanS}(f,0) \in O(1)$
$W_{scanS}(f,1) \in O(1)$

Si $n > 1$, se sabe que $W_{contraerArr}(f,n) \in \Theta(n)$ y $W_{expandirArr}(f,n) \in \Theta(n)$. Ahora si, tomando $a = 1$, $b = 2$, $\epsilon = 1$, $c = \frac{1}{2}$ y $N > 1$ tenemos:

$$W_{contraerArr} \in \Omega(n^{\log_2(1+1)}) = \Omega(n^{\log_2 2}) = \Omega(n)$$

$$W_{contraerArr}(\frac{n}{2}) = \frac{n}{2} = \frac{1}{2} * n = c * W_{contraerArr}(n)$$

Luego, por el tercer caso del Teorema Maestro, podemos afirmar que $W_{reduceS}(2^k) \in \Theta(W_{contraerArr}(2^k)) = \Theta(2^k)$.

Al ser la funcion $f(n) = n$ suave y $W_{reduceS}$ no decreciente, por Regla de suavidad, concluimos que $W_{reduceS} \in \Theta(n)$

Profundidad

Tenemos las siguientes recurrencias para la profundidad de `scanS`:
$S_{scanS}(f,0) = S_{singletonArr}(1) + c_0$
$S_{scanS}(f,1) = S_{singletonArr}(1) + S_f(x_i) + S_{nthArr}(0) + c_1$
$S_{scanS}(f,n) = S_{contraerArr}(f,n) + S_{scanS}(f,\lceil n/2 \rceil) + S_{expandirArr}(f,n)$

Podemos observar que no ocurre ninguna paralelización en la función. Entonces, como en el Trabajo, la Profundidad de `scanS` va a depender de las Profundidades de `contraerArr` y `expandirArr`. $S_{contraerArr}$ la calculamos en el apartado anterior y nos quedo $S_{contraerArr} \in \Theta(1)$.
Ahora calculamos $S_{expandirArr}$.

$$S_{expandirArr}(f,n) = S_{tabulateArr}(f',n) = \Theta(\max_{i=0}^{n/2} S_{f'}(x_i))$$

Al igual que en el trabajo calculado anteriormente f' va a tomar dos formas distintas según la paridad de n. La profundidad de f y `nthArr` son constantes, por lo que la profundidad de f' es constante. Entonces $S_{f'} \in \Theta(1)$. Retomando el calculo de `expandirArr`:

$$\Theta(\max_{i=0}^{n/2} S_{f'}(x_i)) = \Theta(\max_{i=0}^{n/2} \Theta(1)) = \Theta(1)$$

Por lo que podemos concluir que $S_{expandirArr}(f, n) \in \Theta(1)$.

Volviendo a la profundidad de `scanS`. Supongamos que $n = 2^k$. Entonces la profundidad quedaría:
$S_{scanS}(f, 0) = S_{singletonArr}(1) + c_0$
$S_{scanS}(f, 1) = S_{singletonArr}(1) + S_f(x_i) + S_{nthArr}(0) + c_1$
$S_{scanS}(f, n) = S_{contraerArr}(f, n) + S_{scanS}(f, n/2) + S_{expandirArr}(f, n)$

Utilizando la tabla de costos dada en la materia, se obtiene:
$S_{scanS}(f, 1) = O(1) + (1) + c_1 = 2.O(1) + c_1 = O(1)$
Si $n > 1$, se sabe que $S_{contraerArr}(f, n) \in \Theta(1)$ y $S_{expandirArr}(f, n) \in \Theta(1)$. Ahora si, tomando $a = 1$, $b = 2$, tenemos:

$$S_{contraerArr} + S_{expandirArr} \in \Theta(n^{\log_2(1)}) = \Theta(n^0) = \Theta(1)$$

Luego, por el segundo caso del Teorema Maestro, podemos afirmar que $S_{scanS}(2^k) \in \Theta((2^k)^0 \log 2^k) = \Theta(k)$.

Al ser la función $f(n) = \log n$ suave y S_{scanS} no decreciente, por Regla de suavidad, concluimos que $S_{scanS} \in \Theta(\log n)$