Universidad Nacional de Rosario
Facultad de Ciencias Exactas,
Ingeniería y Agrimensura

Estructuras de Datos II

Trabajo Práctico II
Secuencias

Román Castellarin
Juan Ignacio Suarez

20 de junio de 2018

1. Introducción

Implementacion basada en Arrays Persistentes

	W	S
filterS p x	$O(\|x\| + \sum_{i=0}^{\|x\|-1} W[p\ x_i])$	$(\log \|x\| + \max_{i=0}^{\|x\|-1} S[p\ x_i])$
reduceS f e s	$O(\|s\| + \sum_{(x\oplus y)\in \mathcal{O}_r(\oplus,e,s)} W[x \oplus y])$	$O(N)$
scanS	$O(N)$	$O(N)$
showtS	$O(1)$	$O(1)$

1.0.1. filterA es $O(|x|+\sum_{i=0}^{|x|-1} W[p\ x_i])$ en trabajo, y $O(\log |x|+\max_{i=0}^{|x|-1} S[p\ x_i])$ prof.

```
filterA p x = A.flatten (mapA g x)
    where g y = if p y then singletonA y else emptyA
```

Cuadro 1: Definicion de filterA

- **Lema:** `singletonA` y `emptyA` son $O(1)$ en trabajo y profundidad.

  Dem: Ambas hacen una cantidad constante de operaciones independientemente de la entrada.

- **Lema:** `mapA f s` es $O(\sum_{i=0}^{|s|-1} W[f\ \ s_i])$ en trabajo y $O(\max_{i=0}^{|s|-1} S[f\ \ s_i])$ en profundidad.

  Dem: Se desprende como corolario de las cotas de `tabulate g n` con $g\ i = f\ \ s_i$ y $n = |s|$.

Del primer lema, resulta que en la definicion de `map` presentada arriba, $W[g\ y] \in O(W[p\ y])$ y $S[g\ y] \in O(S[p\ y])$.

Del segundo lema anterior, resulta `map g x` es $O(\sum_{i=0}^{|x|-1} W[p\ x_i])$ en trabajo y $O(\max_{i=0}^{|x|-1} S[p\ x_i])$ en profundidad.

Es necesario calcular primero `map g x` para luego aplicar el `flatten`, por lo que la profundidad (ademas del trabajo) se suman ya que no se realizan en paralelo.

Tenemos entonces por la especificacion dada de `flatten` que `filter p x` es $O(|x| + \sum_{i=0}^{|x|-1} W[p\ x_i])$ en trabajo ya que $|g(y)| \in O(1)$, mientras que en profundidad es $O(\log |x| + \max_{i=0}^{|x|-1} S[p\ x_i])$.

1.0.2. reduceA es $O(|s| + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, e, s)} W[x \oplus y])$ en trabajo.

Utilizaremos $\mathcal{O}_r(\oplus, e, s)$ para denotar el conjunto de aplicaciones de $\oplus$ al invocar `reduceA` $\oplus$ `e s`. Notemos que para reducir una secuencia de largo n por aplicacion repetida de $\oplus$ (en cualquier orden) hacen falta $n-1$ aplicaciones, por lo que cardinalidad de este conjunto es $O(|s|)$.

Para el analisis general de costo, supondremos primero que $W[x \oplus y]$ es $O(1)$, y luego deduciremos el caso general.

- **Lema:** Para $W_\oplus, S_\oplus \in O(1)$, `contractA` $\oplus$ `s` es $O(|s|)$ en trabajo y $O(1)$ en profundidad.

  Dem: Corolario de las cotas de `tabulate g n` con $g\ i = s_{2i} \oplus s_{2i+1}$ y $n \in O(|s|)$.

Veamos que como `contract` reduce la longitud de la secuencia en un factor de $1/2$, `reduceByContraction` solo recursa $O(\log |s|)$ veces, y por lo tanto (bajo la hip. de $S_\oplus \in O(1)$) resulta $O(\log |s|)$ en profundidad.

Para el trabajo, veamos que obtenemos una serie geometrica, por lo que (bajo la hip. de $W_\oplus \in O(|s|)$) resulta $O(|s|)$.

Para una especificacion de costos general, notemos que solo debemos agregarle al trabajo obtenido, el trabajo de cada operacion $(x \oplus y)$ hecho, y la profundidad no puede empeorar mas que en un factor $\max_{(x \oplus y) \in \mathcal{O}_r(\oplus, e, s)} S[x \oplus y]$.

Por lo que finalmente obtenemos:

$$O(|s| + \sum_{(x \oplus y) \in \mathcal{O}_r(\oplus, e, s)} W[x \oplus y])$$

en trabajo,y

$$O(\log |s| \cdot \max_{(x \oplus y) \in \mathcal{O}_r(\oplus, e, s)} S[x \oplus y])$$

en profundidad.

1.0.3. scanA bla bla.

```
scanA f e s = (scan_seq, scan_last)
    where (scan_seq, scan_last) = (scanA' f e s) ||| (reduceA f e s)
          scanA' f e s = case A.length s of
            0 -> emptyA
            1 -> singletonA e
            n -> tabulateA g n
            where s' = scanA' f e (contractA f s)
                  g i | even i    = s' ! (i//2)
                      | otherwise = f (s' ! (i//2)) (s ! (i-1))
```

Cuadro 2: Definicion de scanA

Analogamente a como hicimos para el analisis de reduceA, supondremos primero que $\oplus$ es $O(1)$ en trabajo y profundidad, y consideraremos el conjunto $\mathcal{O}_s(\oplus, e, s)$ de aplicaciones de $\oplus$ al invocar `scanS ⊕ e s`.

- **Lema:** Para $W_\oplus, S_\oplus \in O(1)$, `scanA' ⊕ e s` es $O(|s|)$ en trabajo y $O(\log |s|)$ en profundidad.

  Dem: De manera similar al analisis de reduceA, vemos que la recurrencia para el trabajo es $W(n) = W(n/2) + O(n)$, donde el $O(n)$ viene del trabajo de `tabulateA` y `contractA`, y el $W(n/2)$ del trabajo de la llamada recursiva (contractA reduce en un factor de 1/2 la secuencia). De donde el trabajo resulta $O(|s|)$. De la misma forma, la recurrencia para la profundidad es $S(n) = S(n/2) + O(1)$ donde el $O(1)$ viene de la profundidad de `contractA` y `tabulateA`, ambas $O(1)$. Resulta asi una profundidad de $O(\log |s|)$

1.0.4. showtA es $O(1)$ en trabajo y profundidad.

- **Lema:** `takeA, dropA` son $O(1)$ en trabajo y profundidad.

  Dem: Corolario de las cotas de `subArray`.

```
showtA x | n == 0         = EMPTY
         | n == 1         = ELT (x ! 0)
         | otherwise      = NODE (takeA x m) (dropA x m)
   where n = A.length x
         m = n//2
```

Cuadro 3: Definicion de showtA

`showtA` realiza una cantidad constante de operaciones ya que no recursa e invoca funciones que realizan trabajo y profundidad constante.