Soluciones, MARP I

Mario Calvarro Marines

Índice general

Hoja 1

Ejercicio 1

No será coste constante amortizado. Contraejemplo:

```
fun multiapilar (p: pila , k: nat)
    mientras k > 0 hacer
        apilar (p)
        k = k - 1
    fmientras
ffun
```

Coste $O(k)$. Si considero la secuencia de n llamadas a multiapilar tendremos:

$$\hat{c_i} = \frac{1}{n}\sum_{i=1}^{n} c_i = \frac{1}{n}\sum_{i=1}^{n} k = k.$$

Ejercicio 2

Sí:

$$\underbrace{1\dots\underbrace{10\dots0}_{k-1}}_{n} \to^{\text{incrementar}} \underbrace{1\dots\underbrace{01\dots1}_{k-1}}_{n} \to^{\text{decrementar}} \underbrace{1\dots\underbrace{10\dots0}_{k-1}}_{n} \to \dots n \text{ operaciones.}$$

Incrementar y decrementar: $O(k) \Rightarrow n$ operaciones $O(nk)$.

Ejercicio 3

Hemos de calcular $\frac{\sum_{j=1}^{n} c_j}{j}$. Lo hacemos primero para $j = 2^n$.

Consideramos $C_{in} = \{2^i : 0 \leq i \leq n\}$ y $\overline{C}_{in} = (1, \dots, 2^n)\ C_n$.

$$\sum_{i=1}^{\infty} C_i = \sum_{i \in C_n} C_i + \sum_{i=1}^{n} C_i = \sum_{i=1}^{\infty} 2^i + \sum_{i \in C_n} 1 = 2^{n+1} - 1 + \mid \overline{C}_n \mid\mid \overline{C}_n \mid = \mid 1 \dots 2^n \mid - \mid C_n \mid = 2^n - (n+1)$$

$$\Rightarrow \sum_{i=1}^{j} C_i = \left(2^{n+1} - 1\right) + \left(2^n - n - 1\right) = 3 \cdot 2^n - n - 2 \Rightarrow \frac{\sum_{i=1}^{j} C_i}{2^n} \leq 3 \in O(1)$$

De forma similar si tomamos $j = 2^n + j' : j' \in 1, \ldots, 2^n - 1$.

$$\Rightarrow \sum_{i=0}^{j} 3 \cdot 2^n - n - 2 + j \Rightarrow \frac{\sum_{i=0}^{j} C_i}{j} \leq 3 \in O\left(1\right)$$

Ya que $\begin{cases} 2^n \leq j \\ j' \leq j \end{cases}$

Ejercicio 4

Utilizo el método del potencial para calcular el coste amortizado:

$\Phi\left(D_i\right)$ elementos en la lista tras operación i-ésima

- Añadir un número:

$$\hat{c}_i = c_1 + \Phi\left(D_i\right) - \Phi\left(D_{i-1}\right) = 1 + (k+1) - k = 2 \in O\left(1\right)$$

- Reducir-lista:

$$\hat{c}_i = c_1 + \Phi\left(D_i\right) - \Phi\left(D_{i-1}\right) = k + 1 + 1 - k = 2 \in O\left(1\right)$$

Ejercicio 5

```
proc contar(C[0, ..., k - 1] de {0, 1}, E/S posSig: nat)
    j := 0
    mientras j < k AND j < posSig AND C[j] = 1
        C[j] := 0
        j := j + 1
    fmientras

    si j < k AND j < posSig
        C[j] := 1
    fsi

    si j < k AND j = posSig
        C[j] := 1
        posSig := posSig + 1
    fsi
fproc

proc resetear (E/S posSig: nat)
    posSig := 0
fproc
```

$$valor\left(c\right) = \sum_{j=0}^{\text{posSig}} 2^j C\left[j\right]$$

Ejercicio 6

Buscar:

Para cada A_i, hacer búsqueda binaria tiene coste en $O\left(\log 2^i\right) = O\left(i\right)$.

En el caso peor, el elemento que buscamos está en A_{k-1}, luego $\sum_{i=0}^{k-1} i = \frac{(k-1)k}{2} \in O\left(k^2\right) = O\left(\log^2\left(n\right)\right)$.

<u>Insertar</u>: Cuando insertamos n elementos como mucho viajan al nivel $\log\left(n\right)$ vaciamos A_0 de 1 elemento $\frac{n}{2}$ veces, A_1, 2 elementos $\frac{n}{4}$ veces y A_j, 2^j elementos $\frac{n}{2^{j+1}}$ veces $\Rightarrow$

$$T\left(n\right) = \sum_{j=0}^{\log n} 2^j \cdot \frac{n}{2^{j+1}} = \frac{n}{2} \log n.$$

Ejercicio 7

Base: $b\left[0, \ldots, n-1\right] \in \mathbb{Z}^n$ con $b\left[i\right] \geq 2$.

Contador: $v\left[0, \ldots, n-1\right]$ con $0 \leq v\left[i\right] < b\left[i\right]$ y $v\left[0\right]$ menos significativo.

```
proc incrementar (b: base, v: contador)
    i := 0
    mientras i < n AND v[i] = b[i] - 1 hacer
        v[i] := 0
        i := i + 1
    fmientras
    si i < n hacer
        v[i] := v[i] + 1
    fsi
fproc
```

$v\left[0\right]$ cambia n veces, $v\left[1\right]$ cambia $\frac{n}{b[0]}$ veces, $\ldots$, $v\left[i\right]$ cambia $\frac{n}{\prod_{j=0}^{i} b[j]}$ veces.

<u>M. Agregación</u>: Nº Cambios: $\sum_{i=0}^{n} \frac{n}{\prod_{j=0}^{i} b[j]}$

Como $\forall j \in \{0, \ldots, n-1\} : b\left[j\right] \geq 2$:

$$\sum_{i=0}^{n} \frac{1}{\prod_{j=0}^{i} b\left[j\right]} \leq \sum_{i=0}^{n} \frac{1}{2^i} \leq \sum_{i=0}^{\infty} \frac{1}{2^i} = 2$$

<u>Coste amortizado</u>:

$$\frac{1}{n} \sum_{i=0}^{n} \frac{n}{\prod_{j=0}^{i} b\left[j\right]} = \sum_{i=0}^{n} \frac{1}{\prod_{j=0}^{i} b\left[j\right]} \leq 2$$

Ejercicio 8

//COMPLETAR

a) Utilizamos dos pilas una de valores y otra de máximos.

   Operaciones:

```
proc apilar(p1, p2: pila; k: ent)
    si !p2.empty AND k \ge p2.top =>
        p2.push(k)
    fsi
```

```
    p1.push(k)
fproc

func maximo (p2: pila)
    si !p2.empty =>
        return p2.top
    fsi
ffunc

proc desapilar(p1, p2: pila)
    si !p1.empty =>
        si p1.top = p2.top =>
            p2.pop
        fsi
    fsi
    p1.pop
fproc
```

b) Operaciones:

```
proc anyadir(p1, p2: pilas; k: ent)
    si !p1.empty =>
        p1.push(k, max(k, p1.top)
    si no
        p1.push(k, max)
    fsi
fproc
```