ISIS 2103 Diseño de Algoritmos
Semestre 2015-1
Tarea No. 1
Prof. Rodrigo Cardoso
Fecha de entrega: Febrero 23, 2015, 8:30 (por Sicua+)

---

1 **[30/100]** El lenguaje GCL, elegido para expresar los algoritmos en el curso, tiene peculiaridades que no son corrientes en los lenguajes de programación comerciales. Dé respuestas para las siguientes preguntas:

1a Suponga que GCL se enriquece con una instrucción nueva `S`, pero que ésta se puede implementar con instrucciones GCL ya conocidas. ¿Es factible enriquecer el cálculo de Hoare con una regla que permita concluir la corrección de `S` con respecto a una especificación dada?

Sí. Basta extender el cálculo de Hoare con una regla que equivalga al uso de las reglas del cálculo correspondientes a las reglas de las partes que componen `S`. Es decir, si `S` es implementado con una instrucción GCL `S'`, se define la regla

$$\texttt{S}\text{-corrección:}\qquad \frac{\{Q\}\ S'\ \{R\}}{\{Q\}\ S\ \{R\}}$$

[15/30]

1b Dados dos programas, `S1` y `S2`, se dice que `S1` *simula* `S2` cuando, para toda especificación $\langle Q,R\rangle$ (pre / poscondición) se tiene que

$$\{Q\}\ S2\ \{R\} \Rightarrow \{Q\}\ S1\ \{R\}$$

Defina la relación

$$\text{S1 equivale S2} \equiv (\text{S1 simula S2}) \wedge (\text{S2 simula S1})$$

Muestre que `equivale` es una relación de equivalencia. Explique, en términos operacionales, cuándo dos programas son equivalentes.

Obsérvese que `simula` es una relación de orden parcial:

`simula` es reflexiva:

```
   S simula S
=
   {Q} S {R} ⇒ {Q} S {R}
=
   true
```

`simula` es transitiva:

```
   (S1 simula S2) ∧ (S2 simula S3)
=
   ({Q} S2 {R} ⇒ {Q} S1 {R}) ∧ ({Q} S3 {R} ⇒ {Q} S2 {R})
⇒           ⟨⇒ transitividad⟩
   {Q} S3 {R} ⇒ {Q} S1 {R}
=
   S1 simula S3
```

Por tanto, `equivale` es una relación de equivalencia.

[15/30]

Variante

`equivale` es reflexiva:

```
   S equivale S
=
   (S simula S) ∧ (S simula S)
=
   {Q} S {R} ⇒ {Q} S {R}
=
   true
```

`equivale` es transitiva:

```
   (S1 equivale S2) ∧ (S2 equivale S3)
=
   (S2 simula S1) ∧ (S2 simula S1) ∧ (S3 simula S2) ∧ (S2 simula S3)
=
   ({Q}S1{R}⇒{Q}S2{R})  ∧  ({Q}S2{R}⇒{Q}S1{R})  ∧  ({Q}S2{R}⇒{Q}S3{R})  ∧
   ({Q}S3{R}⇒{Q}S2{R})
⇒    〈⇒ transitividad〉
   ({Q}S1{R} ⇒ {Q}S3{R}) ∧ ({Q}S3{R} ⇒ {Q}S1{R})
=
   (S1 simula S3) ∧ (S3 simula S1)
=
   S1 equivale S3
```

[15/30]

---

2 [30/100] Dadas las funciones de variable real positiva

$\log \lceil n \rceil$	$3^{\log n}$	n^{π}
$n!$	$(e/2)^n$	$(e/3)^n$
$(+k \mid 1 \le k \le n : k/2^k)$		$(*k \mid 1 \le k \le \lfloor \log n \rfloor : k/2^k)$

Ordénelas en una secuencia $f_1, f_2, \ldots, f_8$, de manera que $f_i = O(f_{i+1})$, para $i=1,2,\ldots,8$.

Antes de ordenar las funciones, conviene calcular las sumatorias,

Sea $T(n) = (+k \mid 1 \le k \le n : k/2^k)$, $n \ge 0$. Entonces:

$$T(n+1) - T(n) = (n+1)/2^{n+1}, \quad n\ge 0$$
$=$
$$(E-1)T = \langle (1/2)*n*(1/2)^n + (1/2)*(1/2)^n \rangle$$
$\Rightarrow$ $\langle (E - 1/2)^2$ anula $n*(1/2)^n$ y también $(1/2)^n \rangle$
$$(E-1)(E-1/2)^2 T = 0$$

De este modo, hay constantes A, B, C tales que
$$T(n) = A + B/2^n + Cn/2^n \quad , \quad n\ge 0$$

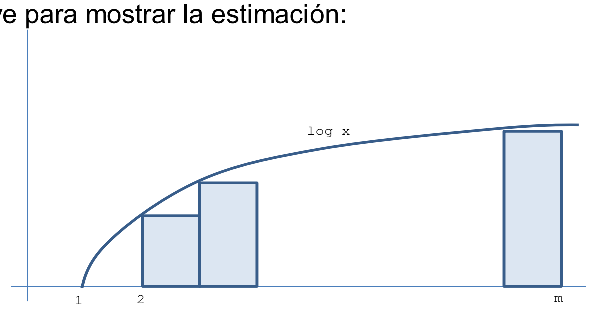
Resolviendo:

$T(n) = 2 - (2n+1)/2^n \;,\; n\geq 0$

El cálculo de $S(n)$ es un poco fuera del alcance de la clase. Debió ser una sumatoria más que una multiplicatoria. Además, límite de la cuantificación debió ir hasta $\lfloor \log n\rfloor$ (porque $\log \lfloor n\rfloor = \log n$, la nitación de parte entera no servía para nada). A pesar de todo esto, con algo de ingenio se podía estimar el prden de magnitud de $S(n)$, tal y como se plantea. Antes de probarlo conviene probar un lema para cuya demostración se usa una técnica que también puede usarse para estimar $n!$ y que resulta en la fórmula de Stirling.

Lema: $(+k|\ 1 \leq k \leq m\colon \log k) = O(m \log m)$

Dem: La siguiente gráfica sirve para mostrar la estimación:



Los rectángulos sombreados representan sumas inferiores para la integral definida de $\log x$ entre 1 y m.

Nótese que:

$(+k|\ 1 \leq k \leq m\colon \log k)$

$\leq$

$\int_1^m \log x\, dx = (1/\ln 2)(x \ln x - 1)_1^m = m \log m.$

Una estimación similar con sumas de áreas por encima de $\log x$ permite establecer que

$(+k|\ 1 \leq k \leq m\colon \log k)$

$\geq$

$\int_1^{m-1} \log x\, dx = (1/\ln 2)(x \ln x - 1)_1^{m-1} = (m-1) \log (m-1).$

De modo que, cuando $m\to\infty$, el cociente $(+k|\ 1 \leq k \leq m\colon \log k)/(m \log m) \to 1$.

Ahora, sean $S(n) = (*k|\ 1 \leq k \leq \lfloor \log n\rfloor\colon k/2^k)$ y $A(n) = \log S(n)$. Entonces:

$A(n)$

$=$

$\log (*k|\ 1 \leq k \leq \lfloor \log n\rfloor\colon k/2^k)$

$=$ ⟨log de producto es suma de logaritmos de multiplicandos⟩

$(+k|\ 1 \leq k \leq \lfloor \log n\rfloor\colon \log k/2^k)$

$=$

$(+k|\ 1 \leq k \leq \lfloor \log n\rfloor\colon \log k) - (+k|\ 1 \leq k \leq \lfloor \log n\rfloor\colon k)$

$\leq$ ⟨Lema ; suma de los primeros log n números ; $\lfloor \log n\rfloor \leq \log n$⟩

$[2*(\log n)*\log (\log n) - (\log n)^2 + \log n]/2$

De modo que

$S(n)$

$\leq$

$n^2\log n - n^2 + n - 1.$

$=$

$O(n^2 \log n)$

Estimar sumatoria T: [8/30]
Estimar sumatoria S(Bono): [+10]

Se definen:

$f_1 = (e/3)^n$

$f_2 = (+k|\ 1 \le k \le n:\ k/2^k)$

$f_3 = \log \lceil n \rceil$

$f_4 = 3^{\log n}$

$f_5 = (*k|\ 1 \le k \le \lfloor \log n \rfloor:\ k/2^k)$

$f_6 = n^\pi$

$f_7 = (e/3)^n$

$f_8 = n!$

Bomno general [+2/30]
Ordenar (4 por cada función en su puesto, excepto f_4) [28/30]
Ordenar bien f_4(Bono) [+ 5]

Se probará que:

$f_1 = (e/3)^n$	$= O(1)$
$f_2 = (+k\|\ 1 \le k \le n:\ k/2^k)$	$= \theta(1)$
$f_3 = \log \lceil n \rceil$	$= \theta(\log n)$
$f_4 = 3^{\log n}$	$= \theta(n^{1.58\ldots})$
$f_5 = (*k\|\ 1 \le k \le \lfloor \log n \rfloor:\ k/2^k)$	$= \theta(n^2 \log n)$
$f_6 = n^\pi$	$= \theta(n^{3.14\ldots})$
$f_7 = (e/2)^n$	$= \theta(1.35\ldots^n)$
$f_8 = n!$	$= \theta((2\pi n)^{1/2}(n/e)^n)$

Rebajar 2 por cada error de justificación

Para justificar el ordenamiento:

Nótese que

$f_1(n) = (e/3)^n = (.90\ldots)^n \le 1$

Por tanto:

$f_1(n) = O(1)$.

También:

$f_2(n) = (+k|\ 1 \le k \le n:\ k/2^k) = 2 - (2n+1)/2^n = \theta(1)$.

$\lim_{n\to\infty} f_2(n)/1 = 2$.

Sin embargo, no es cierto que $1 = O(f_2)$.

También:

$1 = O(\log \lceil n \rceil) = f_3(n)$, de manera que

$\mathbf{f_2 = O(f_3)}$.

Hasta aquí:

$\mathbf{f_1 = O(f_2)}$

$\mathbf{f_2 = \theta(f_3)}$.

Ahora:

$f_4(n) = 3^{\log n} = 2^{\log n \log 3} = n^{\log 3} = n^{1.58\ldots}$

$\lim_{n\to\infty} f_3(n)/f_4(n) = \lim_{n\to\infty} \log n/ n^{1.58} = 0.$

$\mathbf{f_3 = O(f_4)}$.

En esoe punto:

$f_5(n) = \theta(n^2 \log n)$

$\lim_{n\to\infty} f_4(n)/f_5(n) \lim_{n\to\infty} n^{1.58}/(n^2 \log n) = 0$

$\mathbf{f_4 = O(f_5)}$.

Se tiene que

$f_6(n) = n^{\pi} = n^{3.14\ldots}$

$\lim_{n\to\infty} f_5(n)/f_6(n) \lim_{n\to\infty} (n^2 \log n)/n^{3.14} = 0$

$\mathbf{f_5 = O(f_6)}$.

Ahora:

$f_6(n) = n^{3.14\ldots} \le (1.35\ldots)^n = (e/2)^n = f_7(n)$, para n suficientemente grande, i.e.,

$\mathbf{f_6 = O(f_7)}$.

Finalmente, ya que

$f_8(n) = n! = \theta((2\pi n)^{1/2}(n/e)^n)$

es claro que

$\mathbf{f_7 = O(f_8)}$.

---

3 **[40/100]** Dada una matriz de enteros `A[0..m-1,0..n-1]`, donde cada fila está ordenada ascendentemente, y un entero `x:int`, se quiere saber si `x∈A`. Para esto se usa la función `busa`, abajo descrita; a su vez, dentro de su cuerpo, `busa` llama a la función `busbin`:

```
funct busbin (b[0..n-1]: int; p,q:int, x:int):int
{Pre Q:  (∀k|: 0≤k<n-1:  b[k]≤b[k+1]) ∧ p≤q-1}
{Pos R:  (busbin=n ∧  x∉b[0..n-1]) ∨ (0≤busbin<n ∧ x=b[busbin])}
⌈ if p=q-1
     then  if    b[p]=x      then busbin:= p
                             else busbin:= n
           fi
     else  r:= (p+q) ÷2;
           if  b[r]<x  then  busbin:= busbin(b,p,r)
                       else  busbin:= busbin(b,r+1,q)
           fi
  fi
⌋

funct busa (A[0..m-1,0..n-1]: int; x: int): boolean
{Pre Q:  (∀i|: 0≤k<m: (∀j| 0≤j<n:  A[i,j] ≤ A[i,j+1]))}
{Pos R:  (¬busa ∧ x∉A) ∨ (busa ∧ x=A[i,j])}
```

```
        }
⌈ i:= 0;
  j:= n;
  do i≠m ∧ j=n    →    j:= busbin(A[i,.],0,n)
                       if j≠n      → skip
                       [] j=n      → i:= i+1
                       fi
  od;
  busa:= (i≠m);
⌋
```

Para el problema de calcular `busa(A[0..m-1,0..n-1],x)` con el algoritmo anterior, defina como operación básica la asignación de variables.

3a ¿Cuál es el tamaño del problema? (i.e., de qué variable(s) depende el tamaño del problema; o bien, defina el tamaño en función de variables de la llamada).

La complejidad temporal del problema depende de m y de n.

[5/40]

3b Estime el peor caso del tiempo T_{busa} (como $\theta(\ldots)$).

En primer lugar hay que estimar el peor caso de T_{busbin}.

El tamaño del problema `busbin` es el tamaño del subarreglo del arreglo total sobre el que se busca. Las variables p y q delimitan un subarreglo en el que se debe buscar. Entonces, el tamaño del problema para `busbin` se describe con p y q. Se cumple que

$$T_{busbin}(n) = \begin{cases} 1 & , \text{ si } n=1 \\ 1 + \max(T_{busbin}(\lfloor n/2 \rfloor), T_{busbin}(\lceil n/2 \rceil)) & , \text{ si } n>1 \end{cases}$$

Esto se puede aproximar así:

$$T_{busbin}(n) = 1 \quad , \text{ si } n=1$$
$$= 1 + T_{busbin}(n/2)$$

[5/40]

Ahora se puede hacer una transformación de dominio

$u_0 = 1$

$u_k = n$

$u_{k-1} = n/2 \quad , k\geq 1$

De modo que $u_k = 2u_{k-1}$ y, por tanto:

$u_{k+1} - 2u_k = 0 \quad , k\geq 0$

Entonces:

$(E-2)u = 0$

y debe existir una constante α tal que

$u_k = \alpha\, 2^k \quad , k\geq 0.$

Con la restricción de que $u_0=1$, se tiene que $\alpha=1$ y que

$u_k = 2^k = n$
$k = \log n$

Y entonces,

$$T_{busbin}(u_k) = 1 \quad , \text{ si } k=0$$
$$= 1 + T_{busbin}(u_{k-1}) \quad , \text{ si } k>0$$

Por tanto, llamando $v_k \coloneqq T_{busbin}(u_k)$:

$$v_k = 1 + v_{k-1} \quad , \text{ si } k>0$$

$\equiv$

$$(E-1)v = \langle 1 \rangle$$

$\Rightarrow$

$$(E-1)^2 v = 0$$

Ahora, deben existir constantes β, γ, tales que

$$v_k = \beta + \gamma\ k \quad , k\geq 0$$

Por tanto:

$$v_0 = T_{busbin}(u_0) = T_{busbin}(1) = 1 = \beta$$
$$v_1 = T_{busbin}(u_1) = T_{busbin}(2) = 1 + T_{busbin}(u_1) = 1 + 1 = 2 = \beta + \gamma$$

Es decir:

$$v_k = 1 + k, \ k\geq 0$$
$$T_{busbin}(n) = T_{busbin}(u_k) = v_k = 1+k = 1 + \log n.$$

[25/40]

Variante

Usar Teorema Maestro.

[25/40]

La complejidad de T_{busa} depende de `m` y de `n`. Se observa que `busa` llama a `busbin`, a lo sumo, `m` veces.

$$T_{busa}(m,n) = \theta(m * T_{busbin}(n))$$
$$= \theta(m*(\log n +1))$$
$$= \theta(m \log n)$$

[5/40]

3c ¿Cuándo se presenta el peor caso?

Si encuentra a `x` en un paso intermedio, termina; pero el peor caso se da cuando `x` no está o cuando `x` está en la última fila).

[5/40]