UBA Facultad de Ciencias Exactas y Naturales Departamento de Computación Algoritmos y Estructura de Datos I
Segundo parcial 27/06/2016

Nro. de orden: 7
LU: 449/12
Apellidos: PAWLOW
Nombres: DANTE

A

1	2	3	4	TOTAL
16	32	28	15	92

<u>Aclaraciones</u>: El parcial NO es a libro abierto. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos. **Entregar cada ejercicio en hoja separada.**

Ejercicio 1. [**20 puntos**]

Dado el siguiente programa, y asumiendo que se cuenta con el siguiente requiere: $|a| == n \wedge n \ mod\ 2 == 0$

a) Dar la especificación del ciclo (P_c, Q_c, I, f_v y cota)

b) Demostrar que $(I \wedge f_v \leq c) \Rightarrow \neg B$

```
bool buscarPositivo(int[] a, int n) {
  bool hayPositivo = false;
  int i = n/2;
  while (!hayPositivo && i < n) {
    hayPositivo = a[i] > 0 || a[n-i-1] > 0;
    i++;
  }

  return hayPositivo;
}
```

Ejercicio 2. [**35 puntos**]

Dado el siguiente código, que contiene un único ciclo cuya precondición y postcondición son:
$P_c : i == 0 \wedge a == pre(a)$
$Q_c : |a| == |pre(a)| \wedge i == n \wedge$
$(\forall j \leftarrow [0..|a|), j \ mod\ 2 == 0) a_j == pre(a)_j \wedge$
$(\forall k \leftarrow [0..|a|), k \ mod\ 2 == 1) a_k == pre(a)_{k-1}$.
Asumir que se cuenta con el siguiente requiere: $|a| == n \wedge n \geq 2$ $\wedge\ n \ mod\ 2 == 0$

```
void copiarPosParesEnImpares(int[] a, int n) {
  int i = 0;
  while (i < n-1) {
    a[i+1] = a[i];
    i = i+2;
  }
}
```

a) Determinar si el siguiente invariante es correcto para el ciclo dado:
$I : 0 \leq i \leq n \wedge |a| == |pre(a)| \wedge$
$(\forall j \leftarrow [0..i), j \ mod\ 2 == 0) a_j == pre(a)_j \wedge$
$(\forall j \leftarrow [0..i), j \ mod\ 2 == 1) a_j == pre(a)_{j-1} \wedge$
$(\forall j \leftarrow [i..n)) a_j == pre(a)_j$.
Si el invariante es correcto realizar las siguientes demostraciones. Si no lo es, presentar una versión corregida y usar esa versión para las siguientes demostraciones:

b) Demostrar $P_c \Rightarrow I$

c) Demostrar $(I \wedge \neg B) \Rightarrow Q_c$

d) Demostrar que el cuerpo del ciclo preserva el invariante

Ejercicio 3. [**30 puntos**] A partir de la especificación de un problema que dado un arreglo y un entero k devuelve el elemento del arreglo que sea mayor a otros k elementos; y las siguientes funciones auxiliares:

problema `tieneKmenores` (a: $[\mathbb{Z}]$, n, k: $\mathbb{Z}$) = res : $\mathbb{Z}$ {
  requiere $|a| == n \wedge 0 \leq k < n$;
  requiere $todosDistintos(a, n)$;
  asegura $(\exists i \leftarrow [0..n)) cantidadMenores(a, a_i) == k \wedge a_i == res$;
}

aux `yaLoEncontre` (a: $[\mathbb{Z}]$, i, k: $\mathbb{Z}$) : Bool $= (\exists j \leftarrow [0..i)) cantidadMenores(a, a_j) == k$;
aux `resEsResultado` (a: $[\mathbb{Z}]$, res, k: $\mathbb{Z}$) : Bool $= res \in a \wedge cantidadMenores(a, res) == k$;
aux `todosDistintos` (a: $[\mathbb{Z}]$, n: $\mathbb{Z}$) : Bool $= (\forall i, j \leftarrow [0..n), i \neq j) a_i \neq a_j$;
aux `cantidadMenores` (a: $[\mathbb{Z}]$, x: $\mathbb{Z}$) : $\mathbb{Z} = |[y \mid y \leftarrow a, y < x]|$;

a) Implementar un programa en C++ que respete la especificación dada, tenga como signatura **int** tieneKmenores(**int**[] a, **int** n, **int** k y su ciclo principal respete el siguiente invariante: $I : 0 \leq i \leq n \wedge (yaLoEncontre(a, i, k) \Rightarrow resEsResultado(a, res, k))$.

b) Indicar la complejidad del programa implementado.

c) Es posible hacer un programa de menor complejidad si se cuenta con la función sort[1] que ordena el arreglo y cuya complejidad es $\mathcal{O}(nlog(n))$? En caso negativo, justificar. En caso afirmativo, escribir el programa e indicar su complejidad.

d) Es posible hacer un programa de menor complejidad si se cuenta con la siguiente precondición?
requiere: $(\forall j \leftarrow [0..n-1)) a_j < a_{j+1}$
En caso negativo, justificar. En caso afirmativo, escribir el programa e indicar su complejidad.

Ejercicio 4. [**15 puntos**] Implementar un programa en C++ que dado un entero positivo k devuelva un vector de $2*k+1$ posiciones que calcule la función $f(x) = x^2$ para el rango $[-k...k]$. Por ejemplo, para $k = 3$, se debe devolver el siguiente vector $[9, 4, 1, 0, 1, 4, 9]$. El programa debe tener la siguiente aridad: vector<**int**> parabola(**int** k).

[1] La función sort para un arreglo a de tamaño n se utiliza escribiendo sort(a,a+n), no es necesario que la implementen.

orden 7

DANTE PAWLOW
L.U.: 449/12

HOJA 1
FECHA 27-06-2016

ojo que aunque $i == n$
$a[n-1] > 0$ podría pasar

1) requiere $|a| == n \wedge n \bmod 2 == 0$

a) $Qc: ((i == n \wedge (\forall j \leftarrow [0..n-1])\ a[j] \leq 0) \vee$

$(n/2 < i \leq n \wedge (\exists x \in [0..n-1])\ a[x] > 0)) \wedge a == pre(a)$

(X) deberías decir algo de hayPositivo (es lo q' devuelve el código y acá lo perdés)

$Pc:$ hayPositivo == false $\wedge\ i == n/2 \wedge a == pre(a) \wedge n == |a|$

$fv:\ n - i$ cota: $c = 0$.

Escribo abajo el I:

$I:$ ~~I~~ $n/2 \leq i \leq n \wedge |a| == |pre(a)| \wedge$

~~$((\forall j \leftarrow (n-1-i..i))\ a[j] == pre(a)[j] \wedge a[j] \leq 0) \wedge$~~

$(\forall j \leftarrow [0..n-1])\ a[j] == pre(a)[j] \wedge$

~~$((\forall x \leftarrow (n-1-i..i))\ a[x] \leq 0 \Rightarrow$ hayPositivo == false$) \vee$~~

~~$((\exists y \leftarrow (n-1-i..i))\ a[y] > 0 \Rightarrow$ hayPositivo == true$))$~~

(if $((\exists x \leftarrow [n-1-i..i))\ a[x] > 0)$ then hayPositivo == true else hayPos == false)

b) q.v.q $(I \wedge fv \leq c) \Rightarrow \neg B$

$B: (\neg$ hayPositivo $\wedge\ i < n) \Rightarrow \neg B:$ (hayPositivo $\vee\ i \geq n$)

por I $n/2 \leq i \leq n$

$fv \leq c \Leftrightarrow n - i \leq 0 \Leftrightarrow n \leq i$

$(n/2 \leq i \leq n) \wedge (n - i \leq 0) \Rightarrow n \leq i$

$\therefore (I \wedge fv \leq c) \Rightarrow \neg B$ (por tautología true $\vee$ x == true)

→ a) $I:$ $n/2 \leq i \leq n \wedge a == pre(a) \wedge$

if $(\exists x \leftarrow (n-1-i..i))\ a[x] > 0)$ then hayPositivo == true

else hayPositivo == false

(más conciso: ~~[illegible]~~ hayPositivo == $(\exists x\ (n-1-i..i))\ a[x] > 0$)

NOTA

Ej 2 (1/2) | DANTE PAWLOW | L.U.: 449/12 | orden 7 | HOJA 2 | FECHA 27-06-2016

2) requiere: $|a| == n \wedge n \geq 2 \wedge n \bmod 2 == 0$

a) El invariante parece ser correcto para el ciclo dado.

b) q.v.q: $P_c \Rightarrow I$

Agrego al invariante que $i \bmod 2 == 0$ ✓

$P_c: i == 0 \wedge a == pre(a)$

- Como $i == 0 \Rightarrow i \bmod 2 == 0$

- $i == 0 \Rightarrow 0 \leq i \leq n$ ✓
- $a == pre(a) \Rightarrow |a| == |pre(a)|$ ✓ ✓
- Como $i == 0 \Rightarrow [0..i)$ es vacío

  $\Rightarrow (\forall j \leftarrow [0..0), j \bmod 2 == 0)\ a_j == pre(a)_j$ == true ✓

  y $(\forall j \leftarrow [0..0), j \bmod 2 == 1)\ a_j == pre(a)_{j-1}$ == true ✓ ✓

- Por último, como $i == 0 \wedge a == pre(a)$

  $\Rightarrow (\forall j \leftarrow [0..n))\ a_j == pre(a)_j$ ✓ ✓

$\therefore P_c \Rightarrow I$

~~c) q.v.q: $(I \wedge \neg B) \Rightarrow Q_c$~~

~~(*) a) Después de la modificación de Qc considero que es necesario agregar~~

c) q.v.q $(I \wedge \neg B) \Rightarrow Q_c$

i) q.v.q: $|a| == |pre(a)|$, trivialmente cierto por invariante.

ii) q.v.q: $i == n$.

Por I: $0 \leq i \leq n \wedge i \bmod 2 == 0$

Por $\neg B$: $i \geq n-1$

Por requiere: $n \bmod 2 == 0$

$\Rightarrow i > n-1 \wedge 0 \leq i \leq n$ ¿Por qué?

$\Rightarrow i == n$

iii) q.v.q: $(\forall j \leftarrow [0..|a|), j \bmod 2 == 0)\ a_j == pre(a)_j$

Por I $|a| == |pre(a)|$, por (ii): $i == n$
Por requiere ~~xxxx~~ $n == |a|$
$\Rightarrow i == |a|$

Por I: $(\forall j \leftarrow [0..i), j \bmod 2 == 0)\ a_j == pre(a)_j$
$\hookrightarrow i == |a|$ ✓

✓

iv) q.v.q: $(\forall j \leftarrow [0..|a|), j \bmod 2 == 1)\ a_j == pre(a)_{j-1}$

Análogamente a (iii); por I:

$(\forall j \leftarrow [0..i), j \bmod 2 == 1)\ a_j == pre(a)_{j-1}$ ✓
$\hookrightarrow i == |a|$ ✓

~~v) q.v.q.~~

d) // P_c: $i == 0 \wedge a == pre(a)$

```
while ( i < n-1) {
    // E0: vale I ∧ B  ✓
    a[i+1] = a[i] ;
    //E1: vale  i==i@E0 ∧ (∀j (i+1..n)) a_j ==a@E0_j ∧
    //   (∀j← [0..i+1], j mod 2 ==0) a_j == a@E0_j ∧
    //   (∀j← [0..i+1], j mod 2 ==1) a_j == a@E0_{j-1} ∧
    // |a| == |a@E0|
    i = i+2;
    //E2: vale  i == i@E1 + 2  ∧  a ==a@E1  ✓
}//Qc:
```

¿Por qué?

✓

Ahora quiero ver que se conserva el invariante:

q.v.q. $0 \leq i \leq n$

$0 \leq i@E0 \leq n \wedge i@E1 == i@E0 \Rightarrow 0 \leq i@E1 \leq n$

Por B: $i < n-1$, como $i \bmod 2 == 0 \wedge n \bmod 2 == 0$,

mientras se cumpla la guarda $0 \leq i@E1 \leq n-2$ ✓

NOTA

Ej 2 (2/2) DANTE PAWLOW L.U: 449/12 orden 7 HOJA 3 FECHA 27-06-2016

En E2 vale que $i@E2 == i@E1 + 2$, como $0 \leq i@E1 \leq n-2$

$\Rightarrow 2 \leq i@E1 + 2 \leq n+2-2$ es equivalente a

$2 \leq i@E2 \leq n \Rightarrow 0 \leq 2 \leq i@E2 \leq n$ ✓

Cuando se deje de cumplir la guarda $i == n$, entonces sigue valiendo.

ii) q.v.q: $|a| == |pre(a)|$

En E0 vale $|a| == |pre(a)|$ por I

En E1 vale $|a@E1| == |a@E0|$,

implica $|a@E1| == |pre(a)|$

En E2 a no cambia $\Rightarrow a@E2 == a@E1 \Rightarrow |a@E2| == |a@E1|$

implica que $|a@E2| == |pre(a)|$ ✓

iii) q.v.q: $(\forall j \leftarrow [0..i), j \bmod 2 == 0)\ a_j == pre(a)_j$

En E0 vale $(\forall j \leftarrow [0..i), j \bmod 2 == 0)\ a_j == pre(a)_j$ por I

En E1 vale $(\forall j \leftarrow [0..i+1], j \bmod 2 == 0)\ a@E1_j == a@E0_j$ $\leftarrow i@E1 == i@E0$

En E2 vale $i@E2 == i@E1 + 2 \wedge a@E2 == a@E1$

$\Rightarrow (\forall j \leftarrow [0..i@E1+1], j \bmod 2 == 0)\ a@E1_j == a@E0_j$

implica $(\forall j \leftarrow [0..i@E2-1], j \bmod 2 == 0)\ a@E2_j == a@E0_j$

implica $(\forall j \leftarrow [0..i@E2), j \bmod 2 == 0)\ a@E2_j == pre(a)_j$ ✓

iv) q.v.q: $(\forall j \leftarrow [0..i), j \bmod 2 == 1)\ a_j == pre(a)_{j-1}$

Análogamente que para iii

En E1 vale: $(\forall j \leftarrow [0..i@E1+1], j \bmod 2 == 1)\ a@E1_j == pre(a)_{j-1}$

En E2 vale $i@E2 == i@E1+2 \wedge a@E2 == a@E1$, $i@E1+1 == i@E2-1$

$\Rightarrow (\forall j \leftarrow [0..i@E1+1], j \bmod 2 == 1)\ a@E1_j == pre(a)_{j-1}$

implica $(\forall j \leftarrow [0..i@E2), j \bmod 2 == 1)\ a@E2_j == pre(a)_{j-1}$

NOTA

v) q.v.q: $(\forall j \leftarrow [i..n))\ a_j == pre(a)_j$

En E0 vale $(\forall j \leftarrow [i..n))\ a^{@E0}_j == pre(a)_j$ por I

y $i < n-1$ por B.

En E1 vale $(\forall j \leftarrow [i@E1+1..n))\ a@E1_j == a@E0_j$

En E2 vale $i@E2 == i@E1+2 \wedge a@E2 == a@E1$

Como $i@E2 == i@E1+2 \Rightarrow i@E1+1 == i@E2-1$

$\Rightarrow (i@E1+1..n)$ equivale a $(i@E2-1..n)$

que a su vez es equivalente a $[i@E2..n)$

Entonces en E2 vale: $(\forall j\ (i@E1+1..n))\ a@E2_j == a@E0_j$

implica $(\forall j\ (i@E2-1..n))\ a@E2_j == a@E0_j$

implica $(\forall j\ [i@E2..n))\ a@E2_j == pre(a)_j$ ✓

$\therefore$ por (i, ii, iii, iv, v) el cuerpo del ciclo preserva el invariante.

DANTE PAWLOW
L.U.: 449/12

orden 7

HOJA 4

FECHA 27-06-2016

3) a)

```
int tieneKmenores (int[] a, int n, int k) {
    int res = 0;   // lo inicializo en 0, pero no es necesario.
    int i = 0;
    while (i < n) {
        int cant = 0;
        int j = 0;
        while (j < n) {
            if (a[i] > a[j]) {
                cant++;
            }
            j++;
        }
        if (cant == k) {
            res = a[i];
        }
        i++;
    }
    return res;
}
```

b) La complejidad va a ser de $O(n^2)$ porque hay dos ciclos que recorren linealmente toda la lista, uno anidado dentro del otro.

El resto de las operaciones (asignación, comparación, etc) tienen complejidad $O(1)$, por lo que no influyen en la complejidad "total".

NOTA

c) Como requiere que los elementos sean todos distintos, si la lista está ordenada el elemento en ~~la posición x~~ a[x] va a tener x elementos menores que él.

```
int tieneKmenores (int[] a, int n, int k){
    sort(a, a+n);   // función void que modifica la lista.
    return a[k];
}
```

En este caso la complejidad es $O(n \cdot \log(n))$ que es menor a $O(n^2)$. ¿POR QUÉ?

d) ~~El~~ La precondición planteada ~~nos~~ implica que el arreglo a va a estar ordenado de forma creciente con todos sus elementos distintos (es un menor estricto). Esto es equivalente a ordenar la lista, pero la complejidad de nuestro programa no va a tener en cuenta ~~la de~~ el costo de ordenar.

```
int tieneKmenores (int[]a, int n, int k){
  return a[k];
}
```

Este programa va a tener complejidad $O(1)$, ya que cuenta solamente con una asignación.

$O(n^2) > O(n \cdot \log(n)) > O(1)$

NOTA

DANTE PAWLOW orden 7
L.U.: 449/12

HOJA 5
FECHA 27-06-2016

4)

```
vector<int> parabola(int k) {
    vector<int> res;
    int i = -k;
    while (i <= k) {
        res.pushback(i*i);
        i++;
    }
    return res;
}
```

B

NOTA