

D-295

Algoritmos y Estructuras de Datos II

Primer Parcial - Viernes 3 de abril de 2024

#Orden	Libreta	Apellido y Nombre	E1	E2	E3	E4	Nota Final
	363/23	Martínez, Fausto Nicolás	18	18	20	30	55

- Es posible tener una hoja (2 carillas), escrita a mano, con las anotaciones que se deseen, para los dos apartados de TADs.
- Cada ejercicio debe entregarse en hojas separadas.
- Incluir en cada hoja el número de orden asignado, número de libreta, número de hoja, apellido y nombre.
- El parcial se aprueba con 70 puntos.

A

E1. Especificación de problemas [20 pts]

Se llama números perfectos a aquellos naturales mayores a cero que son iguales a la suma de sus divisores positivos propios (divisores incluyendo al 1 y sin incluir al propio número). Se desea especificar el problema `reemplazarNumerosPerfectos`, que dada una secuencia de enteros devuelve la secuencia pero con los valores que se corresponden con números perfectos reemplazados por el índice donde se encuentran.

Ejemplos

- `reemplazarNumerosPerfectos([0, 3, 9, 6, 4, 28, 7]) = [0, 3, 9, 3, 4, 5, 7]`

Los únicos números reemplazados son el 6 y el 28 porque son los únicos números perfectos de la secuencia.

E2. TADs [30 pts]

Dada la siguiente especificación del TAD `Diccionario<K, V>`:

```

TAD Diccionario<K, V> {
  obs data: seq<tupla<K, V>>

  proc diccionarioVacío(): Diccionario<K, V>
    asegura {res.data = {}}

  proc está(in d: Diccionario<K, V>, in k: K): bool
    asegura {res = true ↔ estaClave(d.data, k)}

  proc definir(inout d: Diccionario<K, V>, in k: K, in v: V)
    requiere {d = D0}
    asegura {estaClave(D0.data, k) → |d.data| = |D0.data|}
    asegura {¬estaClave(D0.data, k) → |d.data| = |D0.data| + 1}
    asegura {(k, v) ∈ d.data}
    asegura {(∀i: Z)((0 ≤ i < |D0.data| ∧ D0.data[i]0 ≠ k) → D0.data[i] ∈ d.data)}

  proc obtener(in d: Diccionario<K, V>, in k: K): V
    requiere {estaClave(d.data, k)}
    asegura {(∃i: Z)(0 ≤ i < |d.data| ∧ d.data[i]0 = k ∧ res = d.data[i]1})}

  pred estaClave(data: seq<tupla<K, V>>, k: K)
    {(∃i: Z)(0 ≤ i < |data| ∧ data[i]0 = k)}
}

```

- ¿Es necesario escribir el predicado de igualdad? Si es necesario escríbalo. Si no, explique por qué.
- Agregue a la especificación el procedimiento `obtenerODefinir`. Este procedimiento recibe un diccionario, una clave y un valor. Si la clave ya está definida en el diccionario, devuelve su valor asociado. En cambio, si la clave no está definida en el diccionario, la define con el valor pasado por parámetro y devuelve ese mismo valor.
- ¿Cómo se podría modificar el TAD para que sea posible conocer la cantidad de veces que se consultó cada clave? Explique detalladamente en castellano qué cosas se agregarían y cuáles se modificarían.

E3. Precondición más débil [20 pts]

Dado el siguiente condicional determinar la precondición más débil que permite hacer valer la postcondición propuesta.

```

if (a mod 2 = 0)
  a := a * a
else
  a := -|a|
endif

```

$$Q \equiv (\exists j : \mathbb{Z})(j \geq 0 \wedge j^2 = a)$$

- Describir en palabras la WP esperada.
- Derivarla formalmente a partir de los axiomas de precondición más débil. Para obtener el puntaje máximo deberá simplificarla lo más posible.

E4. Correctitud del ciclo [30 pts]

Dado el siguiente programa con su especificación

$$P_e \equiv \{n > 0 \wedge n \bmod 2 = 0 \wedge i = 1 \wedge res = 1\}$$

```

While (i <= n/2) {
  res := res * i * (n+1-i);
  i := i+1;
}

```

$$Q_e \equiv \{res = n!\}$$

Contamos con el siguiente invariante, que sabemos que es incorrecto:

$$I \equiv \{1 \leq i \leq n/2 + 1 \wedge res = \prod_{j=1}^{2(i-1)} j\}$$

- Señale qué axiomas del teorema del invariante no se cumplen. Justifique con palabras en forma precisa.
- Escriba un invariante que resulte correcto.
- Proponga una función variante y demuestre formalmente que es correcta.

0363/23

Fausto Martínez - 1º Parcial
Algoritmos y Estructuras de Datos II

3/5/2024

Hoja 1

1) aux SumaDivisores ($n:Z$) $Z = \sum_{i=1}^{n-1} \text{if } n \bmod i = 0 \text{ then } i \text{ else } 0$ ✓
pred esPerfecto ($n:Z$) {
 $n > 0$ \wedge $n = \text{SumaDivisores}(n)$ ✓
}

proc ReemplazarPerfectos ($s: \text{seq}\langle Z \rangle$) \cdot $\text{seq}\langle Z \rangle$ {

requiere { True } ✓

asegura { ($b:Z$) ($0 \leq i < |s|$) \wedge esPerfecto($s[i]$)
 \rightarrow $\text{res}[i] = i$ }

asegura { ($b:Z$) ($0 \leq i < |s|$) \wedge \neg esPerfecto($s[i]$)
 \rightarrow $\text{res}[i] = s[i]$ }

asegura { $|s| = |\text{res}|$ } (esto debería ir arriba para que no se interrumpan los otros)

}

LU 363/23

Fausto Martínez - 1o Parcial
Algoritmos y Estructuras de Datos II3/5/2024
Hoja 2

2) a) Es necesario escribir el predicado de igualdad, ya que, por default, la igualdad se da solo si coinciden todos los observadores. Y, usando el concepto clásico de diccionario, no hay un orden en los elementos del mismo, sin embargo las secuencias sí tienen orden y eso es el problema.

Para explicitarlo, veamos el sig. ejemplo:

Consideramos que $\{ \text{"Argentina": 1, "Francia": 2} \}$ y

$\{ \text{"Francia": 2, "Argentina": 1} \}$

Según el mismo diccionario, pero sin cambiar la definición de la igualdad no lo logramos.

Por eso, defino el siguiente predicado de igualdad:

$$\text{pred sonIguales}(d_1: \text{Diccionario} \langle K, V \rangle, d_2: \text{Diccionario} \langle K, V \rangle) \{$$

$$((\forall i: \mathbb{Z})(0 \leq i < |d_1.\text{data}| \rightarrow d_1.\text{data}[i] \in d_2.\text{data}) \wedge$$

$$(\forall i: \mathbb{Z})(0 \leq i < |d_2.\text{data}| \rightarrow d_2.\text{data}[i] \in d_1.\text{data}))$$

$$\}$$

De esta manera, me aseguro de fijarme lo que me importa: que las secuencias tengan los mismos elementos, más allá del orden.

b) proc obtenerODefinir($\text{input } d: \text{Diccionario} \langle K, V \rangle, k: K, v: V$) $V \{$

requiere $\{ d = D_0 \}$

~~asegura $\{ \text{estaClave}(k, D_0) \rightarrow |d.\text{data}| = |D_0.\text{data}| \wedge$~~

$$(\forall i: \mathbb{Z})(0 \leq i < |D_0.\text{data}| \rightarrow D_0.\text{data}[i] \in d.\text{data}) \wedge$$

$$(\exists j: \mathbb{Z})(0 \leq j < |d.\text{data}| \wedge d.\text{data}[j]_0 = k \wedge \text{res} = d.\text{data}[j]_1)$$

asegura $\{ \rightarrow \text{estaClave}(k, D_0) \rightarrow |d.\text{data}| = |D_0.\text{data}| + 1$

$\wedge \text{estaClave}(z, d) \wedge \langle k, v \rangle \in d.\text{data} \wedge$

$$(\forall i: \mathbb{Z})(0 \leq i < |D_0.\text{data}| \wedge D_0.\text{data}[i]_0 \neq k) \rightarrow D_0.\text{data}[i]_1 \in d.\text{data}$$

$$\wedge \text{res} = v \}$$

}

2) En el caso en el que quisiera conocer la cantidad de veces que se consultó cada clave, implementaría el siguiente observador:

obs Contador: dict <K, Z>

Luego, dentro de los procedimientos que yo considere que involucran "consultar" la clave (en este caso, serán todos los proc que reciben k como parámetro, es decir este, definir, obtener y eventualmente obtenerDefinir).

Debería asegurar que si es la primera vez que accedo a la clave, que la agrego a Contador y la inicializo en 1

y si es una vez posterior, (esto se deduciría de la naturaleza del proc) sumaría 1 al contador asociado a la clave.

Siempre, por supuesto habría que aclarar que los contadores asociados a las claves que no fueron consultadas se mantendrán en el mismo valor.

habría un proc para consultar el contador

~~XXXXXX~~

3) a) Si a es par, le asigno su cuadrado, así que a efectivamente será un cuadrado por default

Si a es impar, le asigno un número negativo, así que nunca será un cuadrado perfecto.

Todo parece indicar que lo wp será $a \bmod 2 = 0$ y chequeemoslo.

b) wp (if...endif, a)

$$\equiv \text{def}(a \bmod 2 = 0) \wedge ((a \bmod 2 = 0 \wedge \text{wp}(a = a^2, a)) \vee (a \bmod 2 = 1 \wedge \text{wp}(a = -|a|, a)))$$

Axioma 4

$$\equiv \underbrace{(a \bmod 2 = 0 \wedge \text{wp}(a = a^2, a))}_{\textcircled{1}} \vee \underbrace{(a \bmod 2 = 1 \wedge \text{wp}(a = -|a|, a))}_{\textcircled{2}}$$

$$- \textcircled{1} \quad a \bmod 2 = 0 \wedge \text{wp}(a = a^2, a)$$

$$\begin{aligned} \text{Ax 1} \rightarrow & \equiv a \bmod 2 = 0 \wedge \text{def}(a^2) \wedge \mathbb{Q}^{a^2} \\ & \equiv a \bmod 2 = 0 \wedge \underbrace{(\exists j \in \mathbb{Z}) (j \geq 0 \wedge j^2 = a^2)}_{\textcircled{+}} \end{aligned}$$

⊙ Observación Este predicado es siempre True, pues tomando $j = |a|$, deducimos que existe j .

$$\textcircled{1} \equiv a \bmod 2 = 0 //$$

$$- \textcircled{2} \quad a \bmod 2 = 1 \wedge \text{wp}(a = -|a|, a)$$

$$\begin{aligned} \text{Ax 1} \rightarrow & \equiv a \bmod 2 = 1 \wedge \text{def}(-|a|) \wedge \mathbb{Q}^{-|a|} \\ & \equiv a \bmod 2 = 1 \wedge \underbrace{(\exists j \in \mathbb{Z}) (j \geq 0 \wedge j^2 = -|a|)}_{\textcircled{+}} \end{aligned}$$

⊙ Obs Este predicado es siempre False, pues $-|a|$ es un número menor o igual que 0, y en particular menor que 0 pues $a \bmod 2 = 1$. Luego, j^2 nunca puede ser estrictamente menor que 0, sea cual sea el j , por lo que deducimos que no existe ningún j que cumpla $\Rightarrow \textcircled{2} = \text{False}$

B B

práctici
Dato

ejercicios

Obteniendo, tendríamos $\textcircled{1} \vee \textcircled{2}$
que es $a \bmod 2 = 0 \vee \text{False}$
que reduce a:

$$a \bmod 2 = 0$$

Weakest Precondition

Que es, efectivamente
lo que sospechaba

MB!

LU 3631
4) a) P
• P

LU 303/23

Fausto Martínez - 1º Parcial
Algoritmos y Estructuras de Datos II3/6/2024
Hoja 4

4) a) Podemos ver fácilmente que

$$\bullet P_c \rightarrow I \text{ pues si } n > 0 \wedge n \bmod 2 = 0 \Rightarrow n \geq 2$$

$$\rightarrow \frac{n}{2} + 1 \geq 2$$

y como $i = 1$, entonces

$$1 \leq i \leq 2 \leq n/2 + 1$$

A parte, como $res = 1$ \wedge $l = 1$

$$\Rightarrow res = \prod_{j=1}^{2^{(l-1)}} j = \prod_{j=1}^{2^{(1-1)}} j$$

$$\bullet l \wedge B \rightarrow G_c$$

$$\text{pues } l \wedge B \Rightarrow l = n/2 + 1 \Rightarrow res = \prod_{j=1}^{2^{(l-1)}} j = \prod_{j=1}^n j = n!$$

Por descarte, la propiedad que no se cumple es

$$\bullet \{l \wedge B\} \notin I$$

Veamos como se comporta el ciclo tomando
como ejemplo $n=8$

- $res = 1$
- 1ª iteración $i=1 \rightarrow res = 1 + 1 + (4-1) \Rightarrow res = 8$
 - 2ª iteración $i=2 \rightarrow res = 8 + 2 + (4-2) \Rightarrow res = 8 + 2 + 7$

Acá ya podemos ver que el invariante no vale, \checkmark
 pues el mismo nos dice que $res = \prod_{j=1}^{2^{(l-1)}} j$
 y acá se ve bastante directo que
 res no es eso. Sin embargo, sigamos, sospecho
 que me va a servir para el 6)

- 3ª iteración $i=3 \rightarrow res = (8 + 2 + 7) + 3 + (4-3) \rightarrow res = 17 + 6 + 3 + 2$
- 4ª iteración $i=4 \rightarrow res = (17 + 6 + 3 + 2) + 4 + (4-4) \rightarrow$
 $res = 8! \quad \wedge \quad i=5 //$

Parece que la agarrando los números como en un
suspensado orden...

$$8! = 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1$$

Por lo pronto, lo que puedo asegurar, es que el invariante que nos propusieron no se mantiene el final de cada iteración, y por tanto, no $\{1 \wedge B\} S \{1\}$

b) Basado en lo observado en el inciso anterior estoy tentado a elegir el invariante

$$I_2 \equiv 0 \leq i \leq \frac{n}{2} + 1 \wedge res = \prod_{j=i}^n c_j * \prod_{j=1}^{i-1} c_j$$

si me da el tiempo, intento probar que es correcto al terminar el inciso c) *(no traigo)*

c) Busco una fv que se haya reduciendo y cumple

4) $\{1 \wedge B \wedge fv = V_0\} S \{fv < V_0\}$

5) $1 \wedge fv \leq 0 \rightarrow \neg B$

Propongo $fv = \frac{n}{2} - i + 1$ ✓

Veamos que cumple ④ y ⑤ *(voy a usar el invariante I2 pero creo que no me afecta mucho el resultado)*

④ Para ver $\{1 \wedge B \wedge fv = V_0\} S \{fv < V_0\}$, veamos

por Teorema, que $1 \wedge B \wedge fv = V_0 \rightarrow \text{wp}(S, fv < V_0)$

$\text{wp}(res := res + c_i(n-i), i = i+1, fv < V_0)$

$\text{wp}(res := res + c_i(n-i), \text{wp}(i = i+1, \frac{n}{2} - (i+1) < V_0))$

$\text{wp}(res := res + c_i(n-i), \frac{n}{2} - i < V_0)$

$\text{wp} \equiv \frac{n}{2} - i < V_0$ ✓

luego $1 \wedge B \wedge fv = V_0 \rightarrow fv = V_0 \rightarrow \frac{n}{2} - i + 1 = V_0$

$\rightarrow V_0 - 1 < V_0 \Rightarrow \frac{n}{2} - i < V_0 \equiv \text{wp}(S, fv < V_0)$ ✓

$\therefore \{1 \wedge B \wedge fv = V_0\} S \{fv < V_0\}$

⑤ $1 \wedge fv \leq 0 \rightarrow fv \leq 0 \rightarrow \frac{n}{2} - i + 1 \leq 0$

$\Rightarrow i \geq \frac{n}{2} + 1 \Rightarrow i > \frac{n}{2} \equiv \neg B$

$\therefore 1 \wedge fv \leq 0 \Rightarrow \neg B$, luego el ciclo termina y fv es correcto

W 363/23

Fausto Martínez - 1o Paralelo
Algoritmos y Estructuras de Datos II3/5/2024
Hoja 5

Bien, ahora intentare demostrar que I_2 es un invariante
Correcto (no por canchero si no para verificar
si esta bien o no)

Quiero ver que ① $P_c \Rightarrow I_2$ ② $I_2 \wedge B \Rightarrow WP(S, B)$ ③ $I_2 \wedge \neg B \Rightarrow Q_c$

① Como $n > 0 \wedge n \bmod 2 = 0 \Rightarrow n > 2$
 $\Rightarrow \frac{n}{2} + 1 > 2$

Luego como $i = 1 \Rightarrow 1 \leq i \leq 2 \leq \frac{n}{2} + 1$ ✓

Como $res = 1 \wedge i = 1$

$$\Rightarrow res = \prod_{j=1}^n (n+2-j) + \prod_{j=2}^n j$$

$$= \prod_{j=n+2-1}^n j + \prod_{j=2}^{(n-1)} j //$$

$\therefore P_c \Rightarrow I_2$

③ $I_2 \wedge \neg B \Rightarrow Q_c$

$$0 \leq i \leq \frac{n}{2} + 1 \wedge res = \prod_{j=n+2-i}^n j + \prod_{j=2}^{i-1} j \wedge i > \frac{n}{2}$$

$$\Rightarrow i = \frac{n}{2} + 1 \wedge res = \prod_{j=n+2-i}^n j + \prod_{j=2}^{(i-1)} j$$

$$\Rightarrow res = \prod_{j=n+2-(\frac{n}{2}+1)}^n j + \prod_{j=2}^{\frac{n}{2}+1} j$$

$$= \prod_{j=\frac{n}{2}+1}^n j + \prod_{j=2}^{\frac{n}{2}} j = \prod_{j=2}^n j = n!$$

$\therefore I_2 \wedge \neg B \Rightarrow Q_c$

② $I_2 \wedge B \rightarrow wp(S, I_2)$

$$\begin{aligned} wp(S, I_2) &\equiv wp(res = res + (n+1), I = (I), res \neq 1) \\ &\equiv wp(res = res + (n+1), wp(res = 1, res \neq 1)) \\ &\equiv wp(res = res + (n+1), res \neq 1) \\ &\equiv \underline{res + (n+1) = n!} \end{aligned}$$

$I_2 \wedge B \equiv$
 $0 \leq i \leq \frac{n}{2} + 1 \wedge res = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j \wedge i \leq n/2$

$0 \leq i \leq \frac{n}{2} \wedge res = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j$

$\Rightarrow res + (n+1) = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j + (n+1)$

$= [n \times (n-1) \times \dots \times (n+1-i) + (n+1)] \times [(i-1) \times (i-2) \times \dots \times 2]$
 $\neq 0 \wedge (n+1)$

$I_2 \wedge B \rightarrow wp(S, I_2)$

$wp(S, I_2) = wp(res = res + (n+1), I = (I), \text{res} = 0 \leq i \leq \frac{n}{2} + 1 \wedge \text{res} = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j)$
 $\equiv wp(res = res + (n+1), 0 \leq i \leq \frac{n}{2} + 1 \wedge \text{res} = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j)$

$\equiv wp\{0 \leq i \leq \frac{n}{2} + 1 \wedge res + (n+1) = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j\}$

$\equiv \{0 \leq i \leq \frac{n}{2} + 1 \wedge res = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j\}$

$I_2 \wedge B \equiv 0 \leq i \leq \frac{n}{2} + 1 \wedge res = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j \wedge i \leq \frac{n}{2}$

$\Rightarrow \text{res} + (n+1) \geq 0 \wedge i+1 \leq \frac{n}{2} + 1 \wedge res = \prod_{j=n+1-i}^n j + \prod_{j=2}^{i-1} j$

$\equiv wp(S, I_2)$

$\therefore \{I_2 \wedge B\} S \{I_2\}$

Uepp, mi interesante b es correcto ☺. Ah y del programa es correcto respecto a su especificación