Algoritmos y Estructuras de Datos II
Segundo parcial – 15 de noviembre de 2014

Aclaraciones

- El parcial es **a libro abierto**.
- Cada ejercicio debe entregarse **en hojas separadas**.
- Incluir en cada hoja el número de orden asignado, número de hoja, apellido y nombre.
- Al entregar el parcial, completar el resto de las columnas en la planilla.
- Cada ejercicio se calificará con **MB**, **B**, **R** o **M**, y podrá recuperarse independientemente de los demás. Para aprobar el parcial se puede tener hasta 1 (un) ejercicio **R** (regular), siempre que no se trate del ejercicio 1. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Diseño

Se quiere diseñar el sistema de control de acceso al famosísimo barrio privado Los Pepinos. Cuando se crea el sistema se informa la nómina de patentes habilitadas y el listado de personas que residen en el barrio. La gente puede ingresar al barrio únicamente en auto. Cuando ingresan se registran con su DNI. Los autos se identifican por patente, que se puede representar como un *string* de seis caracteres. Un auto puede ingresar al barrio si su patente está habilitada y al menos uno de los pasajeros reside en el barrio. Las personas que ingresan y no residen en el barrio se consideran invitadas por el auto con el que ingresaron (dicho auto es el "anfitrión"). Las personas pueden salir en cualquier auto, sin importar si se trata de su anfitrión. Se quiere saber además cuántos individuos son sospechosos. Un individuo se considera sospechoso si es un invitado cuyo anfitrión no está presente en el barrio. Notar que si el anfitrión vuelve, el individuo deja de ser sospechoso. A continuación se presenta la especificación del correspondiente tipo abstracto de datos:

TAD PEPINOS

observadores básicos

residente?	: dni × pepinos	⟶	bool	
habilitado?	: patente × pepinos	⟶	bool	
autos	: pepinos	⟶	conj(patente)	
personas	: pepinos	⟶	conj(dni)	
anfitrión	: dni d × pepinos p	⟶	patente	$\{d \in \text{personas}(p) \wedge_{\text{L}} \text{invitado?}(d, p)\}$

generadores

iniciar	: conj(patente) × conj(dni)	⟶	pepinos	
llegaAuto	: patente a × conj(dni) ds × pepinos p	⟶	pepinos	$\{a \notin \text{autos}(p) \wedge ds \cap \text{personas}(p) = \emptyset \wedge \text{habilitado?}(a, p) \wedge \exists d \in ds.\ \text{residente?}(d, p)\}$
saleAuto	: patente a × conj(dni) ds × pepinos p	⟶	pepinos	$\{a \in \text{autos}(p) \wedge ds \subseteq \text{personas}(p) \wedge \#ds > 0\}$

otras operaciones

invitado?	: dni d × pepinos p	⟶	bool	$\{d \in \text{personas}(p)\}$
sospechoso?	: dni d × pepinos p	⟶	bool	

axiomas

$\text{residente?}(d, \text{iniciar}(as, ds)) \equiv d \in ds$
$\text{residente?}(d, \text{llegaAuto}(a, ds, p)) \equiv \text{residente?}(d, p)$
$\text{residente?}(d, \text{saleAuto}(a, ds, p)) \equiv \text{residente?}(d, p)$

$\text{habilitado?}(a, \text{iniciar}(as, ds)) \equiv a \in as$
$\text{habilitado?}(a, \text{llegaAuto}(a', ds, p)) \equiv \text{habilitado?}(a, p)$
$\text{habilitado?}(a, \text{saleAuto}(a', ds, p)) \equiv \text{habilitado?}(a, p)$

$\text{autos}(\text{iniciar}(as, ds)) \equiv \emptyset$
$\text{autos}(\text{llegaAuto}(a, ds, p)) \equiv \text{autos}(p) \cup \{a\}$
$\text{autos}(\text{saleAuto}(a, ds, p)) \equiv \text{autos}(p) \setminus \{a\}$

$\text{personas}(\text{iniciar}(as, ds)) \equiv \emptyset$
$\text{personas}(\text{llegaAuto}(a, ds, p)) \equiv \text{personas}(p) \cup ds$
$\text{personas}(\text{saleAuto}(a, ds, p)) \equiv \text{personas}(p) \setminus ds$

$\text{anfitrión}(d, \text{llegaAuto}(a, ds, p)) \equiv$ **if** $d \in ds$ **then** a **else** $\text{anfitrión}(d, p)$ **fi**
$\text{anfitrión}(d, \text{saleAuto}(a, ds, p)) \equiv \text{anfitrión}(d, p)$

$\text{invitado?}(d, p) \equiv d \in \text{personas}(p) \wedge \neg\text{residente?}(d, p)$

$\text{sospechoso?}(d, p) \equiv d \in \text{personas}(p) \wedge_{\text{L}} \text{invitado?}(d, p) \wedge_{\text{L}} \text{anfitrión}(d, p) \notin \text{autos}(p)$

Fin TAD

Contexto de uso:

- llegaAuto(**inout** p : pepinos, **in** a : patente, **in** ds : conj(dni)) y saleAuto – en $\mathcal{O}(n \log(m))$, donde $n = \#(ds)$ es la cantidad de personas que ingresan/egresan, y $m = R + I$, con R la cantidad total de residentes e I la cantidad total de invitados presentes en el barrio (incluyendo a los que ingresan/egresan).

- #sospechosos(**in** p : pepinos) : nat – cantidad de sospechosos presentes en el barrio, en $\mathcal{O}(1)$, más precisamente:

$$\#\text{sospechosos}(p) = \#(\{d : \text{dni} \mid \text{sospechoso?}(d, p)\})$$

- #anfitriones(**in** p : pepinos, **in** k : nat) – determina la cantidad de autos que actualmente son anfitriones de a lo sumo k invitados en $\mathcal{O}(\log(I))$, donde I es la cantidad total de invitados presentes en el barrio. Más precisamente:

$$\#\text{anfitriones}(p, k) = \#(\{a : \text{patente} \mid \#\text{invitadosDe}(p, a) \leq k\})$$

$$\#\text{invitadosDe}(p, a) = \#(\{d : \text{dni} \mid d \in \text{personas}(p) \wedge_{\text{L}} \text{invitado?}(d, p) \wedge_{\text{L}} \text{anfitrión}(d, p) = a\})$$

Se pide:

(a) Diseñar una estructura para representar el barrio privado. Indicar en castellano el invariante de representación de la estructura propuesta, explicando para qué sirve cada parte, o utilizando nombres autoexplicativos.
(b) Implementar los algoritmos correspondientes a las cuatro operaciones pedidas, respetando las complejidades temporales estipuladas.
(c) Justificar claramente cómo y por qué los algoritmos, la estructura y los tipos soporte permiten satisfacer los requerimientos pedidos. No es necesario diseñar los módulos soporte, **pero sí describirlos, justificando por qué pueden (y cómo logran)** exportar los órdenes de complejidad que su diseño supone.

Ej. 2. Ordenamiento

Dado un arreglo A de n números naturales, se dice que es *Benford* si y sólo si hay un elemento que aparece exactamente 1 vez, un elemento que aparece exactamente 2 veces, un elemento que aparece exactamente 3 veces, etc. Más precisamente, un arreglo A es Benford si existen elementos $a_1, a_2, \ldots, a_k$: nat tales que:

$$\text{cantApariciones}(a_i, A) = i \quad \text{para cada } i \in \{1, 2, \ldots, k\}$$

y, además, los únicos elementos del arreglo A son los ya mencionados, es decir:

$$\text{cantApariciones}(b, A) = 0 \quad \text{si } b \notin \{a_1, a_2, \ldots, a_k\}$$

Diseñar un algoritmo que ordene un arreglo Benford y cuya complejidad temporal en peor caso sea estrictamente mejor que $\Theta(n \log n)$. Es decir, la complejidad debe ser $O(n \log n)$ pero no debe ser $\Omega(n \log n)$. Explicitar la complejidad obtenida y justificar por qué el algoritmo propuesto la cumple.

Ej. 3. Técnicas algorítmicas

Sea F un arreglo de n números naturales que contiene una permutación de los números entre 1 y n, es decir, para cada $y \in [1..n]$ existe un único $x \in [1..n]$ tal que $F[x] = y$. Dado $x \in [1..n]$, escribimos $F^2[x]$ para $F[F[x]]$ y, en general, inductivamente:

$$\begin{array}{rcl} F^1[x] & := & F[x] \\ F^{n+1}[x] & := & F[F^n[x]] \end{array}$$

Diseñar un algoritmo para calcular el arreglo F^n es decir:

$$[F^n[1], \quad F^n[2], \quad F^n[3], \quad \ldots \quad F^n[n]]$$

Justificar su complejidad temporal, que debe ser $\mathcal{O}(n \log n)$ en peor caso.

Ejemplo: si F es el arreglo $[4, 1, 5, 2, 3]$, se tiene:

x	1	2	3	4	5
$F^1[x]$	4	1	5	2	3
$F^2[x]$	2	4	3	1	5
$F^3[x]$	1	2	5	4	3
$F^4[x]$	4	1	3	2	5
$F^5[x]$	2	4	5	1	3

Sugerencia: calcular el arreglo F^n a partir del arreglo $F^{\lfloor n/2 \rfloor}$, analizando los casos n par y n impar. Se puede usar la siguiente propiedad, si es necesario:

$$F^{i+j}[x] = F^i[F^j[x]]$$