Algoritmos y Estructuras de Datos II
Segundo parcial — 8 de Junio de 2016

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 **Promocionado**, **Aprobado**, **Regular**, o **Insuficiente**.
- El parcial completo está aprobado si el primer ejercicio tiene al menos **A**, y entre los ejercicios 2 y 3 hay al menos una **A**. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Diseño

Desde Laboratorios ROCHO® nos pidieron diseñar un sistema de seguimiento de recetas. Lo que quiere el mencionado laboratorio es controlar cuán popular son sus medicamentos y cuáles son sus médicos más fieles. A través del seguimiento de recetas en farmacias, Laboratorios ROCHO® puede conocer qué médico recetó qué medicamento en todo momento. De esta forma pueden aumentar los precios de los remedios más populares y premiar con viajes a los médicos que más recetan sus productos. En una receta se prescribe al menos un fármaco y como mucho tres. Los mismos pueden ser o no elaborados por Laboratorios ROCHO®. Además, en cualquier momento se quiere premiar al médico más obsecuente del mes (el más recetador de fármacos ROCHO®) pero **sin repetir**, es decir, un médico no puede ser dos veces premiado en el mismo mes.

TAD MÉDICO es NAT, **TAD** FÁRMACO es STRING, **TAD** RECETA es <ID, CONJ(FÁRMACO)> donde ID es NAT
TAD FECHA es <AÑO, MES, DÍA> donde AÑO es NAT, MES es ENUM{1...12}, DÍA es ENUM{1...31}

TAD ROCHO® tracking

géneros RT **exporta** generadores, observadores, otras operaciones

generadores

iniciar : conj(médico) cm × conj(fármaco) cf $\longrightarrow$ RT $\{\neg\emptyset(cm) \wedge \neg\emptyset(cf)\}$

registrarReceta : receta r × fecha f × médico m × RT rt $\longrightarrow$ RT
$\{r \notin \text{todasLasRecetas(rt)} \wedge f \geq \text{últFecha}(rt) \wedge m \in \text{médicos}(rt) \wedge 1 \leq \#\Pi_2(r) \leq 3\}$

premiar : RT rt $\longrightarrow$ RT
$\{\neg\emptyset(\text{candidatos(últFecha(f,rt),rt)}) \wedge \text{colaAConj(empleadosDelMes(últFecha(rt),rt))} \neq \text{médicos(rt)}\}$

observadores básicos

⊗ médicos : RT $\longrightarrow$ conj(médico)

⊗ vademecum : RT $\longrightarrow$ conj(fármaco)

quéRecetó : médico m × fecha f × RT rt $\longrightarrow$ conj(receta) $\{m \in \text{médicos}(rt)\}$

empleadosDelMes : fecha f × RT rt $\longrightarrow$ cola(médico)

otras operaciones

últFecha : RT rt $\longrightarrow$ fecha $\{\neg\emptyset(\text{todasLasRecetas(rt)})\}$

másObsecuente : fecha × conj(médico) cm × RT $\longrightarrow$ médico $\{\neg\emptyset(cm)\}$

candidatos : fecha × RT $\longrightarrow$ conj(médico)

totalRecetado : médico m × fecha × RT rt $\longrightarrow$ nat $\{\text{m} \in \text{medicos(rt)}\}$

⊗ todasLasRecetas : RT $\longrightarrow$ conj(receta)

⊗ contarLosDelLab : conj(receta) × conj(fármaco) $\longrightarrow$ nat

⊗ mismoTotalRecetado : conj(médico) cm × médico m × RT rt $\longrightarrow$ conj(médico) $\{cm \cup m \subseteq \text{médicos}(rt)\}$

⊗ colaAConj : cola(médico) $\longrightarrow$ conj(médico)

Nota: los axiomas con el símbolo ⊗ no están especificados aquí.

axiomas

últFecha(registrarReceta(r,f,m,rt)) ≡ f últFecha(premiar(rt)) ≡ últFecha(rt)

quéRecetó(m, f, iniciar(cm, cf)) ≡ ∅ quéRecetó(m, f, premiar(rt)) ≡ quéRecetó(m, f, rt)

quéRecetó(m, f, registrarReceta(r,f',m',rt)) ≡ quéRecetó(m, f, rt) ∪ **if** m = m' ∧ mes(f) = mes(f') **then** {r} **else** ∅ **fi**

empleadosDelMes(f, iniciar(cm, cf)) ≡ Vacía

empleadosDelMes(f, registrarReceta(r,f',m,rt)) ≡ empleadosDelMes(f,rt)

empleadosDelMes(f, premiar(rt)) ≡ **if** mes(f) = mes(últFecha(rt)) **then** encolar(dameUno(candidatos(f,rt)), empleadosDelMes(f,rt)) **else** empleadosDelMes(f, rt) **fi**

candidatos(f,rt) ≡ **mismoTotalRecetado**(médicos(rt) − **colaAConj**(empleadosDelMes(f,rt)), másObsecuente(f,cm,rt), rt)

```
másObsecuente(f, cm, rt)  ≡ if ∅?(sinUno(cm)) then
                              dameUno(cm)
                            else
                              if totalRecetado(dameUno(cm), f, rt) > totalRecetado(másObsecuente(f, sinUno(cm),rt), f,
                              rt) then
                                dameUno(cm)
                              else
                                másObsecuente(f, sinUno(cm),rt)
                              fi
                            fi
totalRecetado(m,f,rt)     ≡ contarLosDelLab(quéRecetó(m, f, rt), vademecum(rt))
```

Fin TAD

Se quiere diseñar del TAD ROCHO® Tracking sabiendo que en el contexto de uso se quieren guardar **solamente** los últimos K años de historia en el sistema. Además, se deben cumplir los siguientes requerimientos de complejidad temporal en **peor caso**:

- REGISTRARRECETA(r,f,m): $O(r_m + \log(M))$
  donde r_m es la longitud máxima de los fármacos de r y M es la cantidad de médicos en el sistema.
- PREMIAR: $O(\log(M))$ donde M la cantidad de médicos.
- EMPLEADOSDELMES y ~~MÁSRECETADO:~~ $O(1)$

Se pide:

(a) Diseñar una estructura para representar el TAD ROCHO® Tracking. Indicar en castellano el invariante de representación de la estructura propuesta, explicando para qué sirve cada parte, o utilizando nombres autoexplicativos.
(b) 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.
(c) Escribir los algoritmos correspondientes a las operaciones REGISTRARRECETA y PREMIAR, respetando las complejidades temporales mencionadas.
(d) Explicar cómo se podrían implementar los algoritmos de todas las operaciones exportadas con la estructura propuesta.

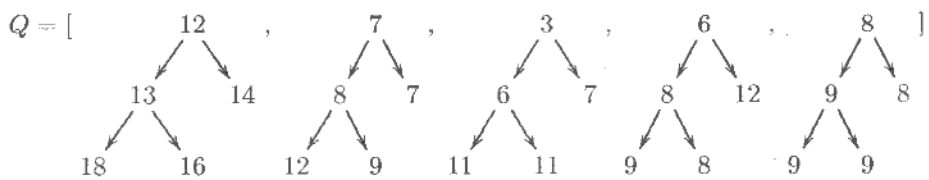
Ej. 2. Ordenamiento

Se tiene un arreglo de n colas de prioridad. Cada cola de prioridad almacena n números naturales. Supondremos que los números más chicos representan mayor prioridad (de tal manera que, por ejemplo, cada cola de prioridad podría estar representada con un *MinHeap*). Se quiere calcular el arreglo ordenado de los n números más chicos que figuran en la totalidad de la estructura.

a) Diseñar un algoritmo: másPrioritarios(**in** Q : arreglo(colaPrior(nat))) $\rightarrow$ res : arreglo(nat) que calcule lo pedido.

b) Justificar la complejidad temporal, que debe ser $O(n \log n)$ en peor caso.

Ejemplo (con $n = 5$, mostrando las colas de prioridad como si fueran *MinHeaps*):



$\text{másPrioritarios}(Q) = [3, 6, 6, 7, 7]$

Ej. 3. Dividir y conquistar

Dado un arreglo de $n > 2$ intervalos cerrados de números naturales $I_1, \ldots, I_n$ (cada uno representado como un par $< l_{inf}, l_{sup} >$), se desea encontrar dos de ellos que maximicen su intersección, es decir, un par de índices i y j con $1 \leq i < j \leq n$ tal que la cantidad de valores dentro de la intersección de los intervalos $I_i \cap I_j$ sea máxima entre todos los intervalos de entrada. *Sugerencia:* ordenar los intervalos por su extremo izquierdo.

a) Dar un algoritmo que use la técnica de *Divide and Conquer* y resuelva el problema en tiempo $O(n \log n)$ en el peor caso.
b) Marcar claramente qué partes del algoritmo se corresponden a *dividir*, *conquistar* y *unir* subproblemas.
c) Justificar detalladamente que el algoritmo cumple con la complejidad pedida.