Algoritmos y Estructuras de Datos II
Segundo parcial – 14 de junio 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. Podrán aprobarse los parciales de la materia con hasta 2 (dos) ejercicios **R** (regular) **entre ambos exámenes**, siempre que ninguno de ellos sea un ejercicio 1. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Diseño

El siguiente TAD especifica el comportamiento del servidor de la nueva red social "ReSocial". En "ReSocial", los miembros se dedicarán exclusivamente a "megustearse" unos a otros. Pueden registrarse nuevos miembros en todo momento, y se quiere saber siempre cuántos "me gusta" ha recibido cada miembro.

Cada tanto los miembros deciden expulsar al miembro menos popular de la red (que es alguno de los que menos "me gusta" tienen). Las personas que han sido echadas no pueden volver a ser miembros.

TAD PERSONA **ES** NAT

TAD RESOCIAL

generadores

crear	:	$\longrightarrow$ servidor		
registrar	: servidor s × persona p	$\longrightarrow$ servidor	$\{p \notin \text{miembros}(s) \land p \notin \text{echados}(s)\}$	
recibirMeGusta	: servidor s × persona p	$\longrightarrow$ servidor	$\{p \in \text{miembros}(s)\}$	
echarAlMenosPopular	: servidor s	$\longrightarrow$ servidor	$\{\text{miembros}(s) \neq \emptyset\}$	

observadores básicos

miembros	: servidor	$\longrightarrow$ conj(persona)		
echados	: servidor	$\longrightarrow$ conj(persona)		
puntaje	: servidor s × persona p	$\longrightarrow$ nat	$\{p \in \text{miembros}(s)\}$	

otras operaciones

miembrosConEstePuntaje	: servidor s × nat	$\longrightarrow$ conj(persona)		
mínimoPuntaje	: servidor s	$\longrightarrow$ nat	$\{\text{miembros}(s) \neq \emptyset\}$	
menosPopular	: servidor s	$\longrightarrow$ persona	$\{\text{miembros}(s) \neq \emptyset\}$	

axiomas ...

menosPopular(s) $\equiv$ dameUno(miembrosConEstePuntaje(s, mínimoPuntaje(s)))

...

Fin TAD

Diseñe el tipo abstracto de datos RESOCIAL respetando los siguientes requerimientos:

- En el diseño, como criterio de desempate se elegirá a quien tenga menor identificador de persona. Es decir que MENOSPOPULAR(s) deberá devolver la mínima persona entre aquellas que tengan el mínimo puntaje.
- Los requerimientos de complejidad temporal en **peor caso** son:
  - RECIBIRMEGUSTA(s, p) debe resolverse en $O(\log m_s)$
  - MENOSPOPULAR(s) debe resolverse en $O(\log m_s)$
  - ECHARALMENOSPOPULAR(s) debe resolverse en $O(\log m_s)$
  - ECHADOS(s) debe resolverse en $O(1)$

  siendo $m_s = \#(\text{miembros}(s))$, es decir la cantidad de miembros **actuales** (no echados) del servidor s.

a) Muestre la estructura de representación propuesta.
Explique para qué sirve cada parte, o utilice nombres autoexplicativos.

b) Escriba en pseudocódigo los 4 algoritmos para los que se piden requerimientos de complejidad.

Justifique 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. Dividir y conquistar

Un árbol binario es *bicolor* si sus nodos son de color rojo o negro.

Un árbol binario bicolor es *rojinegro válido*[1] si y sólo si satisface las siguientes 3 propiedades:

1. Todas las hojas son negras.
2. Los hijos de un nodo rojo, de haberlos, siempre son negros.
3. Todos los caminos (desde la raíz hasta una hoja) contienen la misma cantidad de nodos negros.

Escriba un algoritmo *EsRojinegroVálido?*, basado en la técnica de dividir y conquistar, que dado un árbol binario bicolor determine si es o no rojinegro válido. No visite el mismo nodo múltiples veces. Marque claramente en su algoritmo las distintas etapas de la técnica.

Suponga que puede determinar en O(1) si un nodo es rojo o negro.

Calcule la complejidad temporal de su algoritmo en peor caso. Justifique.

Ej. 3. Ordenamiento

Un melómano quiere ordenar su gigantesca colección de mp3 de manera tal que quede ordenada por nombre del artista, desempatando por nombre del disco y finalmente por nombre del tema.

Los discos y los temas siempre tienen nombres conformados únicamente por caracteres estándar: dígitos y letras del alfabeto latino sin acentos. Sin embargo, los artistas pueden tener nombres raros como Björk, ☿, B☯☯M!, ♏ega♉aurus, ☉♑☉, etcétera. No sabemos cuántos caracteres raros existen, pero sí ordenarlos.

Afortunadamente sabemos que los nombres de bandas tienen a lo sumo 40 caracteres; no así los nombres de discos y temas, que pueden tener cualquier longitud.

Se pide ordenar la colección de mp3 en $O(n \cdot \log(b) \cdot |d| \cdot |t|)$, donde n es la cantidad de mp3s, b la cantidad de bandas distintas, $|d|$ la máxima longitud de nombre de disco y |t| la máxima longitud de nombre de tema.

Justifique por qué su solución cumple lo pedido. Puede combinar pseudocódigo y/o descripción en castellano según considere conveniente. Procure que su justificación sea clara y legible, y no omita premisas relevantes.

[1] La definición que usamos en este ejercicio es una versión simplificada de los *red-black trees* clásicos.