Algoritmos y Estructuras de Datos II
Recuperatorio Segundo Parcial – 15/08/2009

Aclaraciones

- El parcial **NO** es a libro abierto.
- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Incluya el número de orden asignado (léalo cuando circule la planilla), apellido y nombre en cada hoja.
- Al entregar el parcial complete los datos faltantes en la planilla.
- Cada ejercicio se calificará con B, R ó M. Una B no significa que el ejercicio está "perfecto", sino que cumple con los requisitos necesarios para aprobar. En los parciales promocionados se asignará una nota numérica más precisa a cada ejercicio.
- Para aprobar el parcial debe obtenerse B en el ejercicio 1 y en alguno de los ejercicios 2 y 3. Un parcial se considera promocionado si está aprobado y su puntaje es 70 o superior.

Ej. 1. Diseño

Se desea implementar un buscador web que permite indexar páginas y realizar dos tipos de búsquedas simples: 1) dado un conjunto de palabras, obtener todas las páginas en las que aparecen todas las palabras del conjunto, y 2) dado un conjunto de palabras, obtener todas las páginas en las que aparece al menos una palabra del conjunto dado. El buscador a implementar ha sido especificado de la siguiente manera.

TAD BUSCADOR

observadores básicos

ids	:		$\longrightarrow$ conj(idTexto)	
buscarAnd	:	buscador $\times$ conj(Palabra)	$\longrightarrow$ conj(idTexto)	

generadores

nuevo	:		$\longrightarrow$ buscador	
agregarTexto	:	buscador b $\times$ página p $\times$ idTexto i	$\longrightarrow$ buscador	$\neg(i \in ids(b))$

otras operaciones

buscarOr	:	buscador $\times$ conj(Palabra)	$\longrightarrow$ conj(idTexto)	

axiomas .

$\text{ids}(\text{nuevo}) \equiv \emptyset$

$\text{ids}(\text{agregarTexto}(b, p, i)) \equiv \{i\} \cup \text{ids}(b)$

$\text{buscarAnd}(\text{nuevo}, q) \equiv \emptyset$

$\text{buscarAnd}(\text{agregarTexto}(b, p, i), q) \equiv \text{buscarAnd}(b, q) \cup (\textbf{if } q \subseteq p \textbf{ then } \text{Ag}(i, \emptyset) \textbf{ else } \emptyset \textbf{ fi})$

$\text{buscarOr}(\text{nuevo}, q) \equiv \emptyset$

$\text{buscarOr}(\text{agregarTexto}(b, p, i), q) \equiv \text{buscarOr}(b, q) \cup (\textbf{if } \neg(\emptyset?(q \cap p)) \textbf{ then } \text{Ag}(i, \emptyset) \textbf{ else } \emptyset \textbf{ fi})$

Fin TAD

PAGINA es CONJ(PALABRA).

PALABRA es STRING ACOTADO.

IDTEXTO es NAT.

Se desea implementar el buscador descripto anteriormente de manera tal que las operaciones *agregarTexto* y *buscarAnd* tengan complejidad temporal $O(n * k)$ en el peor caso, donde n es el tamaño del conjunto ingresado como parámetro y k es la cantidad máxima de textos en el buscador que contienen alguna de las palabras del conjunto ingresado como parámetro. Además, *buscarOr* debe tener complejidad temporal $O(n * t)$ en el peor caso, donde n es el tamaño del conjunto y t el tamaño del resultado.

Se pide:

- Describir la estructura a utilizar, documentando claramente cómo la misma resuelve el problema y cómo cumple con los requerimientos de eficiencia. El diseño debe incluir sólo la estructura de nivel superior. De ser necesario para justificar los órdenes de complejidad, describa las estructuras soporte.
- Escribir una versión en lenguaje imperativo de los algoritmos para las operaciones *buscarAnd* y *buscarOr*.

Ej. 2. Ordenamiento y Divide and Conquer

La cantidad de parejas en desorden de un arreglo A de tamaño n es la cantidad de parejas de posiciones $1 \leq i < j \leq n$ tal que $A[i] > A[j]$. Dar un algoritmo cuya complejidad temporal sea estrictamente mejor que $O(n^2)$ en el peor caso que calcule la cantidad de parejas en desorden de un arreglo. Considerar hacer una modificación de un algoritmo de sorting.

Ej. 3. Eliminación de la recursión

Dada la siguiente función que, dado una cadena de 0 y 1 correspondientes a la representación binaria de un natural (digito más significativo a la izquierda), permiten calcular el valor natural representado

$$b2D : secu(nat)s \to nat \qquad long(s) > 0 \wedge (\forall i : nat)\ (0 < i \leq long(s) \Rightarrow 0 \leq s[i] \leq 1\ )$$

$$b2d(s) \quad \equiv \quad \textbf{if } long(s) = 1 \textbf{ then } prim(s) \textbf{ else } ult(s) + 2 * b2d(com(s)) \textbf{ fi}$$

Se pide:

a) Indique qué tipo de recursión tiene la función $b2d$.
b) Aplicar la técnica de inmersión+plegado/desplegado para obtener una función que pueda ser transformada algorítmicamente a una forma iterativa. Indique claramente la signatura y la axiomatización completa de todas las funciones que introduzca.
c) A partir del resultado anterior genere un algoritmo imperativo con la versión iterativa de la función $b2d$.