Algoritmos y Estructuras de Datos II
Segundo parcial – 21/6/2008

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 ambos ejercicios. Un parcial se considera promocionado si está aprobado y su puntaje es 70 o superior.

Ej. 1. Diseño

Como el porcentaje de evasión en las multas de automotores es muy alta, el Gobierno de la Ciudad de Buenos Aires se decidió modificar el actual sistema para hacerlo más eficiente.

Cada multa está asociada con un vehículo, identificado por su patente o "dominio" (compuesto por tres letras y tres números), y un mismo vehículo puede tener muchas multas impagas. A su vez, cada persona se identifica por su DNI, y debe ser asociada con todos los vehículos de su propiedad. Las multas se identifican con un código numérico único, y unicamente se quiere registrar las multas impagas (al pagar una multa, ésta es eliminada del sistema).

Se quiere que las operaciones que se listan a continuación cumplan con la complejidad temporal indicada:

- agregarMulta(*inout* s : Sistema, *in* d : Dominio, *in* m : Multa)
  Agrega una multa a un dominio determinado.
  $O(1)$ en peor caso.

- verMultasDeUsuario(*in* s : Sistema, *in* u : DNI) $\rightarrow$ conj(Multa)
  Devuelve todas las multas de todos los vehículos de un usuario, por copia.
  $O(v + m)$ en caso promedio y $O(n + v + m)$ en peor caso, donde v es la cantidad de vehículos del usuario cuyo DNI es u, m la cantidad de multas del usuario u y n es la cantidad total de usuarios.

- asociarVehículo(*inout* s : Sistema, *in* u : DNI, *in* d : Dominio)
  Asocia un nuevo vehículo con un usuario determinado.
  $O(1)$ en caso promedio, $O(n)$ en peor caso, donde n es la cantidad total de usuarios.

- pagarMulta(*inout* s : Sistema, *in* m : CodMulta)
  Elimina una multa del sistema.
  $O(k)$ en el peor caso, donde k es la cantidad total de multas.

Se pide:

a) 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. Para justificar los órdenes de complejidad, describa las estructuras soporte. Si alguna de las estructuras utilizadas requiere que sus elementos posean una función especial (por ejemplo, comparación o función de hash) deberá escribirla.
b) Escribir una versión en lenguaje imperativo del algoritmo verMultasDeUsuario utilizando iteradores. Justifique su complejidad.

Ej. 2. Generalización de funciones

Una fórmula intuitiva para calcular la cantidad máxima de nodos de un árbol binario de altura h es la siguiente:

maxNodos : nat $\longrightarrow$ nat
maxNodos(h)$\equiv$ **if** $h = 0$ **then** 0 **else** $2 \times$ maxNodos$(h-1) + 1$ **fi**

Se pide:

a) Indique qué tipo de recursión tiene la función maxNodos.
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 maxNodos.

Ej. 3. Divide and Conquer

Un gnomo se encuentra escondido en un tablero de $n \times n$ casilleros, y se quiere encontrarlo. Para eso, se cuenta con la función gnomoEstas, que indica si el gnomo se encuentra dentro de una región rectangular del tablero, identificada por las posiciones de sus vértices:

$$\text{gnomoEstas}(in\ t : \text{tablero}, in\ x_1, y_1, x_2, y_2 : \text{nat}) \rightarrow \text{bool}$$
$$(x_1 \leq x_2 \leq \text{tamaño}(t) \wedge y_1 \leq y_2 \leq \text{tamaño}(t))$$

a) Escriba un algoritmo que utilice la técnica de divide and conquer para encontrar al gnomo, es decir, para determinar en qué casilla del tablero se encuentra. Puede escribir funciones auxiliares, pero deberá escribir una función que resuelva el problema, que tenga la siguiente aridad:

$$\text{descubrirGnomo}(in\ t : \text{tablero}) \rightarrow < x, y : \text{nat} >$$

b) Calcule (y justifique) la complejidad del algoritmo para el peor caso, en función de n, considerando por separado los siguientes casos:

   1) La función gnomoEstas tiene complejidad constante ($O(1)$)
   2) La función gnomoEstas tiene complejidad lineal respecto del área del rectángulo que se quiere buscar ($O((x_2 - x_1) \times (y_2 - y_1))$).

   Para simplificar el cálculo de la complejidad, puede asumir que n es potencia de 2.