Algoritmos y Estructuras de Datos II
Recuperatorio del segundo parcial – 30 de noviembre de 2015

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 **P**romocionado, **A**probado, **R**egular, **I**nsuficiente o **N**o Entregó. Notar que **I** es distinto de **N**.
- 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** y a lo sumo una **I**. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Diseño

El complejo de canchas "Hnos. Blazer" está organizando un torneo de fútbol. Como hay muchos equipos anotados y no hay posibilidad de jugar todos contra todos, "Hnos. Blazer" decidió usar un sistema con puntaje dinámico. Al iniciar el torneo se elige la escala de puntajes a utilizar, que son números enteros entre 0 y un cierto m que se decide al arrancar.

Inicialmente a los equipos se les asigna puntaje 0, y de vez en cuando se les modifica el puntaje. Como se trata de conseguir que los equipos que se enfrentan tengan un puntaje parecido, se desea hacer un sistema para almacenar el estado de los puntajes en todo momento, y además permitir consultar, dado un equipo y una distancia d, cuántos posibles rivales tiene cuyo puntaje difiere del suyo en a lo sumo d. Los equipos se representan con string acotado de 20 caracteres.

El siguiente TAD es una especificación para este problema.

TAD EQUIPO es STRING [0 .. 20].
TAD TORNEO

observadores básicos

máxPuntaje	: torneo	$\longrightarrow$	nat	
equipos	: torneo	$\longrightarrow$	conj(equipo)	
puntaje	: equipo e × torneo c	$\longrightarrow$	nat	$\{e \in \text{equipos}(t)\}$

generadores

nuevo	: nat m × conj(equipo) c	$\longrightarrow$	torneo	$\{m > 0 \wedge \neg\emptyset?(c)\}$
cambiarE	: equipo e × puntaje p × torneo t	$\longrightarrow$	torneo	$\{e \in \text{equipos}(t) \wedge p \leq \text{máxPuntaje}(t)\}$

otras operaciones

#cercanos	: equipo e × nat d × torneo t	$\longrightarrow$	nat	$\{e \in \text{equipos}(t)\}$
#puntosEntre	: conj(equipo) c × nat d × nat h × torneo t	$\longrightarrow$	nat	$\{c \subseteq \text{equipos}(t) \wedge d \leq h\}$
ranking	: torneo t	$\longrightarrow$	secu(equipo)	
crearRanking	: conj(equipo) c × torneo t	$\longrightarrow$	secu(equipo)	
máxPte	: conj(equipo) c × torneo t	$\longrightarrow$	equipo	$\{\neg\emptyset?(c)\}$

axiomas

```
máxPuntaje(nuevo(m, c))          ≡ m
máxPuntaje(cambiarE(e, p, t))    ≡ máxPuntaje(t)
equipos(nuevo(m, c))             ≡ c
equipos(cambiarE(e, p, t))       ≡ equipos(t)
puntaje(e,nuevo(m, c))           ≡ 0
puntaje(e,cambiarE(e', p, t))    ≡ if e = e' then p else puntaje(e, t) fi
#cercanos(e, d, t)               ≡ #puntosEntre(equipos(t),puntaje(e, t) − d,puntaje(e, t) + d,t) − 1
#puntosEntre(c, d, h, t)         ≡ if ∅?(c) then
                                       0
                                   else
                                       if d ≤ puntaje(dameUno(c),t) ∧ puntaje(dameUno(c),t) ≤ h then 1 else 0 fi +
                                       #puntosEntre(sinUno(c),d, h, t)
                                   fi
ranking(t)                       ≡ crearRanking(equipos(t), t)
crearRanking(t, es)              ≡ if ∅?(c) then <> else máxPte(es,t) • crearRanking(t, es - máxPte(es)) fi
máxPte(es, t)  ≡ if ∅?(sinUno(es) then
                     dameUno(es)
                 else
                     if puntaje(dameUno(es),t) ≥ puntaje(máxPte(sinUno(es),t)) ∧
                     dameUno(es) ≤ máxPte(sinUno(es),t) then
                         dameUno(es,t)
                     else
                         máxPte(sinUno(es),t)
                     fi
                 fi
```

Fin TAD

Se pide:

(a) Diseñar una estructura para representar el TAD propuesto, con los siguientes requisitos de complejidad:

- NUEVO(m, c) debe tomar $O(m + \#(c))$ donde m es el máximo puntaje y c el conjunto de equipos.
- CAMBIARE(e, p, t) debe tomar $O(log\ n)$ dónde n es la cantidad total de equipos en t.
- PUNTAJE(e, t), MÁXPUNTAJE(t) y EQUIPOS(t) deben tomar $O(1)$.
- #CERCANOS(e, d, t) debe tomar $O(d)$ donde d es el valor del parámetro.
- RANKING(t) debe ser $O(n)$ dónde n es la cantidad total de equipos en t. Notar que $n \neq m$.

b) Indicar en castellano el invariante de representación de la estructura propuesta, explicando para qué sirve cada parte, o utilizando nombres autoexplicativos.
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) Implementar los algoritmos correspondientes a las operaciones CAMBIARE, #CERCANOS y RANKING respetando las complejidades temporales estipuladas. Justificar los órdenes de complejidad.

d) -opcional- Implementar el resto de los algoritmos y justificar los órdenes de complejidad.

Ej. 2. Ordenamiento

a) Se tiene un arreglo de secuencias no vacías de números naturales, cada una de los cuales contiene a todos los números dentro de un intervalo determinado. Se quiere generar un arreglo de naturales que contenga a todos los elementos de todas las secuencias, sin repetidos, ordenados de menor a mayor (ver Figura 1).

Proponga un algoritmo que resuelva el problema en $O(n + (k \log k))$, donde k es la cantidad de secuencias y n la cantidad total de elementos (es decir, la suma de las longitudes de todas las secuencias). Indique y **justifique** claramente la complejidad temporal en cada paso. Puede utilizar (sin reescribir) cualquiera de los algoritmos vistos en clase y las estructuras auxiliares que considere necesarias.

entrada	[[8 5 7 6] [11 10 12] [4 3 5 6]]
salida	[3 4 5 6 7 8 10 11 12]

Figura 1: Entrada y resultado esperado para el algoritmo de ordenamiento.

b) Dado un arreglo de naturales A de tamaño n implementar un algoritmo cuyo resultado sea otro arreglo I, de tamaño n que cumpla: $(\forall\ 0 \leq i < j < n)\ A[I[i]] < A[I[j]]$.
Justificar la complejidad que debe ser $O(n \log n)$.

Ej. 3. Dividir y conquistar

Dado un natural h_0 las matrices de Hadamard de tamaño $2^k \times 2^k$ se definen como:

- H_0 es la matriz de 1×1, cuyo único elemento es h_0
- Para $k > 0$, H_k se define como la matriz de tamaño $2^k \times 2^k$:

$$\begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix}$$

a) Escribir un algoritmo usando la técnica de dividir y conquistar que multiplique un vector u de tamaño $n = 2^k$ y la matriz de Hadamard H_k en a lo sumo $O(n \log n)$ operaciones.

b) Justificar la complejidad temporal.

c) Indicar claramente en su algoritmo las partes que corresponden a dividir y combinar subproblemas.

Repaso: Multiplicación de una matriz D por un vector F.

$$D \times F = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{pmatrix} \times \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} d_{11}f_1 + d_{12}f_2 + \cdots + d_{1n}f_n \\ d_{21}f_1 + d_{22}f_2 + \cdots + d_{2n}f_n \\ \vdots \\ d_{m1}f_1 + d_{m2}f_2 + \cdots + d_{mn}f_n \end{pmatrix}$$