Algoritmos y Estructuras de Datos II
Recuperatorio del segundo parcial – 19 de julio 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 de la organización de una conferencia científica.

Una conferencia tiene un comité de programa, que acepta o rechaza los trabajos (artículos científicos, también llamados "papers") que recibe. Cada trabajo tiene entre 1 y 6 autores, que pueden o no ser miembros del comité.

Cada trabajo es evaluado por 3 integrantes del comité, cada uno de los cuales debe votar de manera positiva o negativa. Si un trabajo recibe más votos positivos que negativos, se lo acepta; de lo contrario, se lo rechaza. Los trabajos que aún no recibieron tres votos no se consideran aceptados ni rechazados. Por último, debe tenerse en cuenta que ninguno de los 3 evaluadores puede ser autor del trabajo que está evaluando.

TAD PERSONA **ES** NAT

TAD TRABAJO **ES** NAT

TAD VOTO **ES** ENUM({SÍ, NO})

TAD CONFERENCIA

generadores

crearConferencia : conj(persona) *comité* $\longrightarrow$ conferencia $\{\#(comité) \geq 3\}$

enviarTrabajo : conferencia c $\times$ trabajo t $\times$ conj(persona) *autores* $\longrightarrow$ conferencia
$\{t \notin \text{trabajos}(c) \land \#(\text{comité}(c) \setminus autores) \geq 3 \land 1 \leq \#(autores) \leq 6\}$

votar : conferencia c $\times$ trabajo t $\times$ persona p $\times$ voto $\longrightarrow$ conferencia
$\{t \in \text{trabajos}(c) \land p \in \text{comité}(c) \land_L p \notin \text{autores}(c,t) \land \#(\text{claves}(\text{votos}(c,t))) < 3\}$

observadores básicos

comité : conferencia $\longrightarrow$ conj(persona)

trabajos : conferencia $\longrightarrow$ conj(trabajo)

autores : conferencia c $\times$ trabajo t $\longrightarrow$ conj(persona) $\{t \in \text{trabajos}(c)\}$

votos : conferencia c $\times$ trabajo t $\longrightarrow$ dicc(persona, voto) $\{t \in \text{trabajos}(c)\}$

otras operaciones

aceptados : conferencia c $\longrightarrow$ conj(trabajo)

trabajosDondeEsAutor : conferencia c $\times$ persona p $\longrightarrow$ conj(trabajo)

Fin TAD

Diseñe el TAD CONFERENCIA respetando los siguientes requerimientos de complejidad en **peor caso**:

- ENVIARTRABAJO(c, t, *autores*) y TRABAJOS(c) deben resolverse en $O(1)$
- TRABAJOSDONDEESAUTOR(c, p) debe resolverse en $O(T)$
- ACEPTADOS(c) debe resolverse en $O(1)$
- VOTAR(c, t, p, v) y VOTOS(c, t) deben resolverse en $O(\log(T))$

siendo $T = \#(\text{trabajos}(c))$, es decir la cantidad de trabajos enviados a la conferencia.

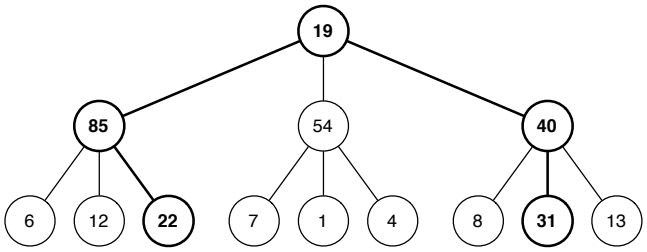
a) Exhiba la estructura de representación propuesta. Utilice nombres declarativos para las componentes de la estructura, y explique para qué sirve cada una.

b) Escriba en pseudocódigo los algoritmos de VOTAR y TRABAJOSDONDEESAUTOR.

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

Dado un árbol ternario de naturales completo, sin elementos repetidos y cuya altura sea al menos 2, interesa obtener una tupla de naturales que represente los nodos inicial y final del camino más pesado (de máxima suma) que pase por la raíz y que no pase más de una vez por el mismo nodo. Por ejemplo, dado el árbol que se exhibe en la figura, el algoritmo debería devolver o bien $\langle 22, 31 \rangle$, o bien $\langle 31, 22 \rangle$ (ambos resultados son válidos).



Escriba un algoritmo que obtenga el resultado deseado en tiempo $O(n)$ en peor caso, siendo n la cantidad de nodos del árbol. Justifique por qué su solución cumple con el orden de complejidad pedido.

Ej. 3. Ordenamiento

Una palabra es *anagrama* de otra si contienen las mismas letras con iguales cantidades de apariciones, aunque posiblemente en distinto orden. Por ejemplo, la palabra corrida es anagrama de la palabra criador.

Dado un conjunto de palabras C, se puede armar una partición de C en k subconjuntos disjuntos de modo que cada subconjunto se componga de palabras que son anagramas entre sí, y ningún subconjunto contenga una palabra que sea anagrama de una palabra de otro subconjunto. Por ejemplo, consideremos el siguiente conjunto de palabras: {rocas, cruce, malos, escudo, arcos, lomas, caros, salmo}. Dicho conjunto puede ser particionado en {{rocas, arcos, caros}, {malos, lomas, salmo}, {cruce}, {escudo}}. Aquí el mayor subconjunto tiene tamaño 3.

Diremos que un arreglo de n palabras distintas es *anagramordenado* si y sólo si se lo puede particionar en k subarreglos del modo ya mencionado y además, dentro de cada uno de los subarreglos, todas las palabras aparecen en orden alfabético. (No es relevante, en cambio, el orden entre distintos subarreglos.)

Retomando el ejemplo anterior:

[escudo, *arcos*, *caros*, *rocas*, cruce, *lomas*, *malos*, *salmo*]	es anagramordenado
[arcos, caros, rocas, *cruce*, escudo, *lomas*, *malos*, *salmo*]	es anagramordenado
[arcos, caros, rocas, *cruce*, escudo, *lomas*, *salmo*, *malos*]	**no** es anagramordenado
[arcos, *escudo*, caros, rocas, *cruce*, lomas, malos, salmo]	**no** es anagramordenado

Escriba un algoritmo que, dado un conjunto C con n palabras, y sabiendo que cada palabra está compuesta por a lo sumo 20 caracteres, devuelva un arreglo anagramordenado que contenga las mismas n palabras que C, con complejidad temporal de peor caso $O(n \cdot (k + \log(t_{ms}))$, siendo n la cantidad de palabras de C, k la cantidad de subconjuntos de anagramas en la partición de C y t_{ms} el tamaño del mayor subconjunto de la partición.

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.

Ayuda: Al armar las clases de anagramas, podría ser útil hallar un buen representante de cada clase de equivalencia.

Ayuda: Al calcular los órdenes de complejidad, recuerde que a veces hay que acotar con cierto cuidado (y explotando lo que se sabe sobre los datos) para lograr una cota bien ajustada.