Algoritmos y Estructuras de Datos II
Segundo parcial – Miércoles 2 de Noviembre de 2016

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, o **I**nsuficiente.
- 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**. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Diseño

Un investigador escribe varios artículos simultáneamente. Cada artículo está compuesto por secciones, que no necesariamente son escritas en orden. El investigador puede realizar modificaciones al texto de alguna sección cuando lo considere necesario, lo cual genera una nueva versión. También puede arrepentirse de sus modificaciones y volver a alguna de las versiones anteriores de la sección (generando igual una nueva versión). Para simplificar, supondremos que, salvo por esto, no es necesario poder consultar las versiones de las secciones que no sean la última (es decir, no es necesario implementar la operación VerVersion).

La revista en la que publica el investigador limita el nombre de las artículos a no más de 90 caracteres. Además el investigador suele equivocarse, y es muy probable que si realizó un cambio en una sección de un artículo, el cambio siguiente en el mismo artículo vuelva a ser en la misma sección.

La especificación es la siguiente:

Sección ES Nat, Artículo ES String[90], Texto ES Secu(Char)

TAD Manejador de artículos

observadores básicos

ExisteArtículo	:	ma × artículo	⟶	bool	
ExisteSección	:	ma n × artículo a × sección	⟶	bool	{ExisteArtículo(n, a)}
Versiones	:	ma n × artículo a × sección s	⟶	nat	{ExisteArtículo(n, a) $\wedge_L$ExisteSección(n, a, s)}
VerTexto	:	ma n × artículo a × sección s	⟶	texto	{ExisteArtículo(n, a) $\wedge_L$ExisteSección(n, a, s)}
VerVersion	:	ma n × artículo a × sección s × nat v	⟶	texto	{ExisteArtículo(n, a) $\wedge_L$ExisteSección(n, a, s) $\wedge_L$v $\leq$ Versiones(n, a, s)}

generadores

Inicio	:		⟶	ma	
AgregarArtículo	:	ma n × artículo a	⟶	ma	{¬ExisteArtículo(n, a)}
AgregarSección	:	ma n × artículo a × sección s × texto	⟶	ma	{ExisteArtículo(n, a) $\wedge_L$¬ExisteSección(n, a, s)}
ModificarSección	:	ma n × artículo a × sección s × texto	⟶	ma	{ExisteArtículo(n, a) $\wedge_L$ExisteSección(n, a, s)}
Arrepentirse	:	ma n × artículo a × sección s × nat v	⟶	ma	{ExisteArtículo(n, a) $\wedge_L$ExisteSección(n, a, s) $\wedge_L$v < Versiones(n, a, s)}

axiomas

...

Fin TAD

Se debe realizar un diseño que cumpla con los siguientes órdenes de complejidad temporal en el peor caso:

- AgregarArtículo y ExisteArtículo: $O(1)$
- ExisteSección y Versiones: $O(\log c)$
- AgregarSección: $O(\log c)$ donde c es la cantidad de secciones del artículo al que se agrega.
- ModificarSección y VerTexto: $O(1)$ si es la última sección modificada de ese artículo y $O(\log c)$ en cualquier otro caso.

Se pide:

A) *i*) Escriba la estructura de representación del módulo "Manejador de artículos". No se pide diseñar todos los módulos de la estructura sino solamente éste. Describa en castellano el resto de los módulos necesarios, incluidas las estructuras auxiliares que utilice en los algoritmos de la parte B.

ii) Escriba el invariante de representación de manera formal (usando funciones de TADs y lógica de primer orden) y también en castellano.

B) Escriba el algoritmo de **ModificarSección** y de toda otra función auxiliar que utilice para ésta, y justifique el cumplimiento de los órdenes solicitados. Para el resto de las funciones, descríbalas en castellano, justificando por qué se cumple el orden pedido con la estructura elegida.

Ej. 2. Ordenamiento

Un sistema de monitoreo de un proceso físico arroja periódicamente mediciones donde cada una consiste en un número natural, con m_t a la medición ocurrida en el instante de tiempo t.

Se conoce, además, el valor de los umbrales L y H con $L < H$ tales que, cuando ocurre una medición $m_{t_0} < L$, vale que $(\forall t < t_0 \wedge m_t < L \Rightarrow m_{t_0} < m_t)$ y, respectivamente, cuando $m_{t_0} > H$, vale que $(\forall t < t_0 \wedge m_t > H \Rightarrow m_t < m_{t_0})$. Dicho de otra manera, todos los valores menores a L aparecen en orden decreciente y todos los valores mayores a H aparecen en orden creciente.

Ejemplo: La secuencia $S = [3, 17, 2, 20, 1, 23, 5, 11, 9]$ es válida para $L = 4$ y $H = 12$, pues los valores menores a 4 aparecen en orden $3, 2, 1$ y los valores mayores a 12 aparecen en orden $17, 20, 23$.

a) Proponga un algoritmo de ordenamiento $ordenarMediciones(A : arreglo(nat), L : nat, H : nat)$ de complejidad $O(n)$, suponiendo que, de n mediciones observadas, $\log n$ de éstas caen dentro del intervalo $[L,H]$.
b) Justifique detalladamente la correctitud del algoritmo y su complejidad temporal.

Ej. 3. Dividir y Conquistar

Se dice que una matriz A de dimensión n es *pirámide invertida* si, para alguna coordenada (i,j) de la matriz (con $i, j \in [1 \dots n]$) se cumple que:

i,j	1	2	3	4	5
1	20	16	9	5	11
2	16	7	4	3	6
3	9	3	1	**0**	3
4	11	7	5	4	7
5	12	10	8	6	9

Figura 1: Pirámide invertida con fondo en i=3, j=4.

i) $A_{ij} = 0$;
ii) Toda fila k tiene valores decrecientes hasta la columna j y crecientes desde ésta en adelante;
iii) Toda columna k tiene valores decrecientes hasta la fila i y crecientes desde ésta en adelante.

En otras palabras, A es una pirámide invertida si tiene una celda (i,j) que vale 0, y el resto de las celdas tienen valores estrictamente crecientes a medida que me alejo en cualquier dirección de a una fila y/o una columna de la posición (i,j).

a) Proponga un algoritmo que utilice la técnica de *Dividir y Conquistar* para encontrar las coordenadas del fondo (valor 0) de una *pirámide invertida*. El algoritmo debe tener complejidad $O(\log n)$.
b) Justifique detalladamente la correctitud del algoritmo y su complejidad temporal, así como qué parte del algoritmo implementa cada una de las diferentes fases de la técnica.