# Algoritmos y Estructuras de Datos II
# Recuperatorio del segundo parcial – 12 de diciembre 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. Para aprobar el parcial se puede tener hasta 1 (un) ejercicio **R** (regular), siempre que no se trate del ejercicio 1. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

## Ej. 1. Diseño

Una oficina cuenta con un sistema de administración de expedientes, en el cual estos cuentan con una cierta prioridad. Un grupo de empleados, es el encargado de procesarlos. Debido a ciertas normativas legales, sólo dos expedientes pueden procesarse en simultáneo.

En el momento en que un empleado decide procesar un expediente, debe tomar el de máxima prioridad entre los disponibles. Como dos empleados no pueden trabajar simultáneamente en el mismo expediente, éste queda bloqueado para el resto. En algún momento, el empleado o bien resuelve el expediente (eliminándolo de los pendientes y archivándolo con los resueltos), o bien lo desbloquea y éste vuelve a estar pendiente. Cada empleado trabaja en a lo sumo en un expediente por vez.

Además, se desea saber, para cada expediente resuelto, el historial de empleados que trabajaron en el mismo, y para cada empleado, el historial de expedientes en los que trabajó.

**Aclaraciones:** Llamamos expedientes pendientes tanto a los que están disponibles como a los que están bloqueados. Si llega un expediente con prioridad mayor a la del expediente en que está trabajando un empleado, éste *no está obligado* a desbloquear automáticamente su expediente actual y tomar el nuevo; éste último se agrega a los pendientes como cualquier otro.

La especificación es la siguiente:

**TAD** OFICINA PÚBLICA

**generadores**

CrearOficina : conj(empleado) $c$ $\longrightarrow$ oficina

AgregarExpediente : oficina $f$ $\times$ expediente $e$ $\longrightarrow$ oficina $\{e \notin Pendientes(f) \cup Resueltos(f)\}$

TomarExpediente : oficina $f$ $\times$ empleado $d$ $\longrightarrow$ oficina
$\{d \in Empleados(f) \wedge_{\text{L}} \neg TieneExpediente?(f,d) \wedge \#Pendientes(f) \geq 1 \wedge \#Bloqueados(f) < 2\}$

DesbloquearExpediente : oficina $f$ $\times$ empleado $d$ $\longrightarrow$ oficina $\{d \in Empleados(f) \wedge_{\text{L}} TieneExpediente?(f,d)\}$

ResolverExpediente : oficina $f$ $\times$ empleado $d$ $\longrightarrow$ oficina $\{d \in Empleados(f) \wedge_{\text{L}} TieneExpediente?(f,d)\}$

**observadores básicos**

Empleados : oficina $\longrightarrow$ conj(empleado)

TieneExpediente? : oficina $f$ $\times$ empleado $d$ $\longrightarrow$ bool $\{d \in Empleados(f)\}$

QueExpedienteTiene? : oficina $f$ $\times$ empleado $d$ $\longrightarrow$ expediente $\{d \in Empleados(f) \wedge_{\text{L}} TieneExpediente?(f,d)\}$

Resueltos : oficina $\longrightarrow$ conj(expediente)

Pendientes : oficina $\longrightarrow$ conj(expediente)

EnCuálesTrabajó : oficina $f$ $\times$ empleado $d$ $\longrightarrow$ secu(expediente) $\{d \in Empleados(f)\}$

QuiénesTrabajaron : oficina $f$ $\times$ expediente $e$ $\longrightarrow$ secu(empleado) $\{e \in Resueltos(f)\}$

**otras operaciones**

Bloqueados : oficina $\longrightarrow$ conj(expediente)

**axiomas**

...

**Fin TAD**

EXPEDIENTE ES NAT (y cuenta con una función que brinda su prioridad), EMPLEADO ES STRING[20].

Diseñar un sistema eficiente, que responda a las siguientes complejidades temporales en peor caso:

- AgregarExpediente: $O(log(n))$, donde $n$ es la cantidad de expedientes no resueltos.
- TomarExpediente, DesbloquearExpediente y EnCuálesTrabajó: $O(1)$.
- ResolverExpediente: $O(log(n) + log(m))$, donde $m$ es la cantidad de expedientes resueltos.
- QuiénesTrabajaron: $O(log(m))$

**Se pide:**

(a) Diseñar una estructura para representar el administrador de expedientes. Indicar en castellano el invariante de representación de la estructura propuesta, explicando para qué sirve cada parte, o utilizando nombres autoexplicativos.
(b) Implementar los algoritmos correspondientes a las operaciones pedidas, respetando las complejidades temporales estipuladas.
(c) 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.

## Ej. 2. Ordenamiento

Desde el Banco Central de la Nación nos contactaron para resolverles eficientemente un problema que tienen a menudo: quieren detectar si un fajo de billetes entregado por un cliente puede tener billetes falsos.

Se sabe que los billetes se numeran consecutivamente y que por año se imprimen $m$ billetes. Además, el banco tiene un listado $L$ de tamaño $n$ de todos los billetes falsos hasta el día de la fecha (la fecha no está acotada) No todos los billetes que existen son falsos pero en el listado del banco hay al menos un billete falso por año.

Se define la probabilidad de que un billete emitido en el año $a$ sea falso, como la cantidad de billetes que el banco tiene en su listado para el año $a$. Los billete que figuran en el listado del banco, tienen la misma probabilidad entre ellos pero mayor al resto.

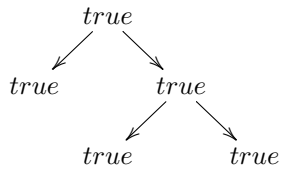
**Se pide:**
Dado el listado del banco $L$ y dado un fajo (arreglo $A$) de $p$ billetes distintos, se quiere ordenar $A$ de mayor a menor según la probabilidad de que el billete sea falso.

Resolver el problema en: $O(n\ log(m) + p\ log(m \cdot p))$

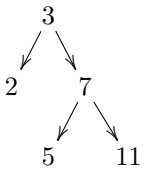
## Ej. 3. Técnicas algorítmicas

Se tiene un árbol binario $A$ de $n$ nodos. El árbol binario contiene booleanos que son siempre *true* (en otras palabras, no nos interesa su valor). Se tiene además un arreglo ordenado de menor a mayor $B$ de $n$ números naturales. Se quiere construir un árbol binario de búsqueda que tenga exactamente la misma forma que el árbol $A$ y que tenga exactamente los mismos elementos que el arreglo $B$. Notar que el árbol binario de búsqueda que se pide construir no necesariamente debe estar balanceado.

**Ejemplo** (con $n = 5$)**:**

$A =$ 


$B = [2, 3, 5, 7, 11]$

construir_abb($A$, $B$) = 


**Se pide:**

(a) Diseñar un algoritmo:

$$\text{construir\_abb}(\textbf{in } A : \text{ab(bool)}, \textbf{in } B : \text{arreglo(nat)}) \rightarrow \text{res} : \text{ab(nat)}$$

que calcule el árbol pedido.
(b) Justificar la complejidad temporal, que debe ser $O(n)$ en peor caso.
(c) Indicar claramente en su algoritmo las partes que corresponden a dividir y combinar subproblemas.