Algoritmos y Estructuras de Datos II
Recuperatorio del segundo parcial – 7 de julio de 2017

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 está aprobado si el primer ejercicio tiene al menos **A**, y entre los ejercicios 2 y 3 hay al menos una **A**.

Ej. 1. Diseño

Estamos desarrollando una agenda que registra tareas a realizar en determinados días del año. Estas tareas pueden agregarse en cualquier momento y se registran para ser realizadas cada año a partir de este momento, en el día agendado (por simplicidad tomamos que un día es un nat entre 1 y 366). La agenda mantiene también el día actual y se le puede preguntar qué tareas hay pendientes para hoy. A lo largo del día, cada una de estas tareas tendrá que darse por cumplida o bien ser pospuesta para otro día. Dado que las tareas son anuales, posponer una tarea altera el día agendado definitivamente, no sólo para esta ocasión sino también para años futuros. Es posible decirle a la agenda que se desea pasar al siguiente día, pero para ello no debe haber tareas pendientes, es decir, todas las tareas del día actual deberán haberse cumplido o pospuesto.

TAD AGENDA

observadores básicos

diaActual	: agenda	$\longrightarrow$	dia
tareas	: agenda	$\longrightarrow$	conj(tarea)
diaDe	: agenda a $\times$ tarea t	$\longrightarrow$	dia $\quad \{t \in \text{tareas}(a)\}$
pendientesDeHoy	: agenda a	$\longrightarrow$	conj(tarea)

generadores

iniciar	: dia *hoy*	$\longrightarrow$	agenda
agendar	: agenda a $\times$ tarea t $\times$ dia d	$\longrightarrow$	agenda $\quad \{t \notin \text{tareas}(a)\}$
cumplir	: agenda a $\times$ tarea t	$\longrightarrow$	agenda $\quad \{t \in \text{pendientesDeHoy}(a)\}$
posponer	: agenda a $\times$ tarea t	$\longrightarrow$	agenda $\quad \{t \in \text{pendientesDeHoy}(a)\}$
pasarDeDía	: agenda a	$\longrightarrow$	agenda $\quad \{\emptyset?(\text{pendientesDeHoy}(a))\}$

otras operaciones

tareasDe : agenda $\times$ dia $\longrightarrow$ conj(tarea)

...

axiomas

...

Fin TAD

Asumiendo que la comparación de tareas es $O(1)$, se debe realizar un diseño que cumpla con los siguientes órdenes de complejidad temporal en el peor caso, siendo n la cantidad total de tareas en la agenda y t la mayor cantidad de tareas agendadas para un mismo día:

- Agendar una nueva tarea en $O(\log n)$.
- Dar por cumplida una tarea del día actual en $O(\log t)$.
- Posponer una tarea del día actual en $O(\log t)$.
- Conocer el día agendado para una cierta tarea en $O(\log n)$.
- Conocer las tareas pendientes del día actual en $O(1)$.
- Conocer las tareas de cualquier día en $O(1)$.
- Pasar de día en $O(1)$.

Se pide:

1. Escriba la estructura de representación del módulo "Agenda" explicando detalladamente qué información se guarda en cada parte de la misma y las relaciones entre las partes. Describa también las estructuras de datos subyacentes.
2. Escriba el algoritmo para posponer una tarea y justifique el cumplimiento de los órdenes solicitados. Para cada una de las demás funciones, descríbalas en castellano, justificando por qué se cumple el orden pedido con la estructura elegida.

Ej. 2. Ordenamiento

Se tiene una secuencia de alturas $h_1, \ldots, h_n$. Decimos que un intervalo $h_t, h_{t+1}, \ldots, h_{t+k}$ es un *edificio* si para todo $i \in [t, t+k)$, el valor $|h_i - h_{i+1}|$ no es mayor que una cierta tolerancia dada θ y además tanto $|h_t - h_{t-1}|$ como $|h_{t+k} - h_{t+k+1}|$ (en caso de existir) son mayores que θ. El *ancho* de un edificio es la cantidad de alturas que lo componen. Por ejemplo, si la secuencia de alturas es

$$[100, 101, 100, 103, 80, 77, 74, 200, 32, 30]$$

y $\theta = 3$, entonces los edificios de esta secuencia de alturas serían $[100, 101, 100, 103]$, $[80, 77, 74]$, $[200]$, $[32, 30]$, y sus anchos serían 4, 3, 1 y 2, respectivamente.

Dado un arreglo de enteros que representan alturas y una tolerancia θ, se desea ordenar el arreglo de alturas reposicionando los edificios ordenándolos según sus anchos en forma creciente (aunque sin alterar el orden relativo de las alturas dentro de cada edificio). En el ejemplo anterior, el resultado esperado sería

$$[200, 32, 30, 80, 77, 74, 100, 101, 100, 103].$$

Escribir un algoritmo que tome un arreglo de enteros (i.e., alturas) y un valor θ y resuelva el problema con complejidad no peor que $O(n)$, donde n es la cantidad de alturas dada. Justifique la correctitud del algoritmo y su complejidad temporal.

Ej. 3. Dividir y Conquistar

Se tiene un arreglo de naturales sin repeticiones que estaba ordenado pero al cual *se lo rotó circularmente* k posiciones a la derecha. Por ejemplo, $[10, 13, 1, 4, 6, 7, 8]$ fue rotado $k = 2$ veces mientras que $[6, 7, 8, 10, 13, 1, 4]$ fue rotado 5 veces. Escribir un algoritmo que reciba uno de estos arreglos y devuelva el máximo elemento en tiempo estrictamente menor que $O(n)$. Justifique la correctitud del algoritmo y su complejidad temporal.