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 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 comenzando la exploración intensiva del planeta Marte utilizando unidades *Mars Rover*. El área de exploración se ha discretizado en coordenadas enteras y en cualquier momento pueden desplegarse unidades *rover*, aunque siempre en la celda (0,0) de la discretización. Los rovers desplegados pueden moverse en cualquier momento en cualquiera de sus cuatro direcciones (norte, sur, este u oeste), siempre que sea hacia una celda válida, y estos movimientos se registran con fines estadísticos. En cualquier momento puede también desactivarse definitivamente cualquiera de las unidades activas. Dado un *itinerario* (i.e., una secuencia de movimientos desde la celda (0,0)), la consulta más requerida es la de saber cuántos rovers (activos o inactivos) realizaron exactamente este itinerario. Si bien no interesa conocer el recorrido hecho por un rover inactivo, notar que este dato es importante para la consulta por itinerario.

TAD ROVER es NAT, TAD DIRECCION es {N, S, E, O}, TAD CELDA es tupla<NAT,NAT>

TAD EXPLORACIÓN

observadores básicos

activos	: exploracion	$\longrightarrow$	conj(rover)		
inactivos	: exploracion	$\longrightarrow$	conj(rover)		
recorridoActual	: exploracion e $\times$ rover r	$\longrightarrow$	secu(direccion)		{r $\in$ activos(e)}
vecesRecorrido	: exploracion e $\times$ secu(direccion) i	$\longrightarrow$	nat		

generadores

iniciar	:	$\longrightarrow$	exploracion		
desplegar	: exploracion e $\times$ rover r	$\longrightarrow$	exploracion		{$r \notin$ rovers(e)}
mover	: exploracion e $\times$ rover r $\times$ direccion d	$\longrightarrow$	exploracion		{$r \in$ activos(e) $\wedge$ movimientoValido(e,r,d)}
desactivar	: exploracion e $\times$ rover r	$\longrightarrow$	exploracion		{$r \in$ activos(e)}

otras operaciones

rovers	: exploracion e	$\longrightarrow$	conj(rover)	
ubicación	: exploracion e $\times$ rover r	$\longrightarrow$	celda	{$r \in$ activos(e)}

axiomas $\forall e$: exploracion, $\forall r$: rover, $\forall d$: direccion

- rovers(e) $\equiv$ activos(e) $\cup$ inactivos(e)

...

Fin TAD

Se debe realizar un diseño que cumpla con los siguientes órdenes de complejidad temporal en el peor caso, siendo r la cantidad de rovers activos en el sistema:

- Desplegar y desactivar un rover en $O(\log r)$.
- Mover un rover en $O(\log r)$.
- Saber dónde está un rover dado en $O(\log r)$.
- Saber cuántos rovers realizaron el itinerario T en $O(long(T))$.
- Obtener el conjunto de rovers activos y el de inactivos, cada operación en $O(1)$.

Se pide:

1. Escriba la estructura de representación del módulo "Exploración" 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 mover un rover 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 un arreglo R con n strings sin repeticiones que define un *ranking*. Se tiene además un arreglo A de m strings tal que todos ellos aparecen en el ranking R. Se quiere ordenar el arreglo A en función del ranking definido por R. Es decir, dados dos elementos s y t de A, s será "menor" que t, si aparece en R antes que t. Por ejemplo, si tenemos

$R =$ `[Brasil, Argentina, Alemania, Chile, Colombia, Francia]` y
$A =$ `[Chile, Francia, Brasil, Chile, Argentina, Brasil]`,

entonces el orden correcto para A sería `[Brasil, Brasil, Argentina, Chile, Chile, Francia]`.

Suponiendo que el largo de todos los strings está acotado por una constante, proponga un algoritmo de ordenamiento que resuelva el problema en una complejidad $O(n + m)$, donde n y m son las cantidades de elementos en el ranking y en el arreglo a ordenar, respectivamente. Justifique la correctitud del algoritmo y su complejidad temporal.

Ej. 3. Dividir y Conquistar

Una empresa informa todos los días el cierre de sus ganancias netas (las cuales pueden ser negativas si tuvo pérdida ese día). Dado un arreglo con tales montos para un período de n días consecutivos, el directrio de la empresa necesita conocer en qué intervalo de días (dentro de este período) se obtuvo la mayor ganancia (la ganancia en un intervalo de días es la suma de los montos informados en los días dentro de ese intervalo).

Usando la técnica de D&C, dar un algoritmo que tome un arreglo con los montos de n días consecutivos (i.e., los días de 0 a $n-1$) e indique un intervalo de días $[i, j]$, con $0 \leq i \leq j < n$, en el cual se maximiza la suma de tales montos (puede asumirse que $n > 0$). El algoritmo propuesto debe tener complejidad estrictamente menor que $O(n^2)$, lo cual debe justificarse adecuadamente.

Ayuda: Si pidiésemos que el intervalo devuelto contenga sí o sí un determinado día k, entonces el problema se resuelve en tiempo $O(n)$ (¡pensar cómo!). Aprvechar ese procedimiento en el algoritmo global de D&C implementado.

Solución Ej 1

Idea para la solución:

- `itinerarios` es un dicc que asocia un itinerario a un nat que indica la cantidad de robots que hicieron ese itineario. Internamente es como un trie que en lugar de tener arreglo de chars tiene un arreglo de 4 posiciones para norte, sur, este y oeste. Se lo puede ver también como un árbol cuaternario.
- `datosDeActivos` es un dicc sobre avl que asocia los robots con tuplas de
  - posición,
  - iterador (o puntero) a `itinerarios` marcando el nodo correspondiente al recorrido actual del robot.
- `activos` es un conjunto que guarda los robots activos (puede ser sobre AVL por ejemplo y la cosa cierra igual).
- `inactivos` es un **conjunto lineal** con los robots inactivos (es importante lo de lineal para agregar en O(1)).

Solución Ej 2

Una idea de la solución:

1. Recorrer A y registrar en un DiccString d la cantidad de apariciones de cada string (cada paso se hace en $O(1)$ porque los strings tienen largo acotado).
2. Recorrer R y por cada string s buscar en d la cantidad k de apariciones de tal string (la búsqueda en d es $O(1)$). Insertar k apariciones del string en el arreglo a devolver.

Solución Ej 3

Una idea de la solución:

- El intervalo en cuestión puede estar:
  - en la primer mitad del arreglo, o
  - en la segunda mitad del arreglo, o
  - empezar en la primer mitad y terminar en la segunda mitad.
- Las dos primeras opciones se resuelven recursivamente y se obtienen los mejores intervalos de cada mitad.
- En la tercera opción, el intervalo tiene que contener al elemento del medio del arreglo, con lo cual la cosa consiste en encontrar el mejor intervalo que contiene tal elemento. Esto se resuelve maximizando el intervalo para cada lado del elemento del medio, en $O(n)$.
- Teniendo los 3 candidatos, se elije el que maximice la suma y listo.