PIMO: Programación Imperativa Modular, 2015-1
Tarea 3: Semanas 6 y 7

Para entregar el viernes 27 de febrero/domingo 1 de marzo de 2015

Problemas conceptuales a las 16:00 (27 de febrero) en la Decanatura de Ingeniería de Sistemas

Problemas prácticos a las 23:59 (1 de marzo)

Tanto los ejercicios como los problemas deben ser resueltos, pero únicamente las soluciones de los problemas deben ser entregadas. La intención de los ejercicios es entrenarlo para que domine el material del curso; a pesar de que no debe entregar soluciones a los ejercicios, usted es responsable del material cubierto en ellos.

Instrucciones para la entrega

Para esta tarea y todas las tareas futuras, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada hoja impresa entregada o en cada archivo de código (a modo de comentario). Adicionalmente, agregue la información de fecha y nombres de compañeros con los que colaboró; igualmente cite cualquier fuente de información que utilizó.

¿Cómo describir un algoritmo?

En algunos ejercicios y problemas se pide "dar un algoritmo" para resolver un problema. Una solución debe tomar la forma de un pequeño ensayo (es decir, un par de párrafos). En particular, una solución debe resumir en un párrafo el problema y cuáles son los resultados de la solución. Además, se deben incluir párrafos con la siguiente información:

- una descripción del algoritmo en castellano y, si es útil, pseudo-código;
- por lo menos un diagrama o ejemplo que muestre cómo funciona el algoritmo;
- una demostración de la corrección del algoritmo; y
- un análisis de la complejidad temporal del algoritmo.

Recuerde que su objetivo es comunicar claramente un algoritmo. Las soluciones algorítmicas correctas y descritas *claramente* recibirán alta calificación; soluciones complejas, obtusas o mal presentadas recibirán baja calificación.

Ejercicios

4.1-1, 4-1-2, 4.1-3 (página 74), **4-2** (página 107), 15.1-2, 15.1-3, 15.1-4, 15.1-5 (página 370)

Problemas conceptuales

1. Suponga que H es un montón binario minimal.
   (a) Diseñe un algoritmo que determine el valor del segundo elemento más pequeño en H en tiempo $O(1)$.
   (b) Diseñe un algoritmo que determine el valor del k-ésimo elemento más pequeño en H en tiempo $O(k \log k)$. Ayuda: trate de usar un montón minimal como estructura de datos auxiliar.
2. Suponga que L es una lista doblemente encadenada con centinela. Diseñe un método *rotate*(r) para L que rota L de tal modo que el elemento en la posición i-ésima resulta en la posición $(i + r)$ *mod* n después de la rotación, en donde n es el tamaño de la lista. Este método debe tener complejidad temporal $O(1 + (r \downarrow (n - r)))$ y no debe modificar ninguno de los nodos en la lista, excepto por las referencias de un par de ellos.
3. Suponga que debe implementar una pila usando como estructura de datos subyacente un arreglo (cuyo tamaño no puede cambiar una vez es creado). Una posible estrategia es iniciar con capacidad para 10 elementos (i.e., con un arreglo subyacente de tamaño 10) y luego, cuando la pila se llene, crear un nuevo arreglo suyacente con el doble de tamaño y llenarlo con los valores almacenados en el arreglo original. Puede suponer que liberar memoria es una operación de orden temporal $O(1)$.
   (a) Explique por qué n operaciones sobre la pila (i.e., *push*, *pop*, *top*) toman tiempo $O(n)$. Ayuda: use el hecho de que una suma geométrica es del orden de su último término; por ejemplo, $1 + 2 + 4 + \cdots + n = 2n + 1 \in O(n)$.
   (b) Suponga que se usa una técnica similar para 'enconger' una pila cuyo uso baje de la mitad del espacio disponible en el arreglo subyacente. Generalice la parte (a) y demuestre que el tiempo del algoritmo resultante es $O(n)$.

Problemas prácticos

Hay tres problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.