Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Final, 12 de septiembre de 2017

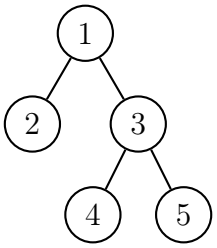
Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

1. **(2 puntos)** Se desea contar el número de elementos de un vector que cumplen que la suma de los elementos a su izquierda es menor estricto que la suma de los elementos a su derecha. Se pide especificar una función que resuelva el problema; implementar como cuerpo de dicha función un algoritmo iterativo eficiente que resuelva el problema; escribir los invariantes y funciones de cota que permitan demostrar la corrección del algoritmo propuesto, y justificar adecuadamente el coste asintótico en el caso peor del algoritmo.

2. **(3 puntos)** Dado un conjunto de $n > 0$ números enteros (n par), se desea dividir el conjunto en dos subconjuntos con $n/2$ elementos cada uno de forma que la diferencia (en valor absoluto) entre las sumas de los elementos de los subconjuntos sea mínima. Implementar un algoritmo que devuelva una solución mínima e imprima por pantalla su valor.

3. **(2 puntos)** Implementa una función que, dado un árbol binario de enteros y un número entero no negativo k, determine el número de hojas cuya profundidad es mayor que k. Por ejemplo, para el siguiente árbol



la función devolvería 3 si $k = 0$ o $k = 1$, devolvería 2 si $k = 2$ y devolvería 0 si $k \geq 3$.

Aparte de implementar este subprograma, debes indicar la complejidad del mismo.

4. **(3 puntos)** La DGT nos ha pedido ayuda para gestionar el *carnet por puntos*. Los conductores están identificados de manera unívoca por su DNI y la cantidad de puntos de un conductor está entre 0 y 15 puntos inclusives. La implementación del sistema se deberá realizar como un TAD `CarnetPorPuntos` con las siguientes operaciones:

   - `nuevo(dni)`: Añade un nuevo conductor identificado por su `dni` (un `string`), con 15 puntos. En caso de que el dni esté duplicado, la operación lanza un error "Conductor duplicado".
   - `quitar(dni, puntos)`: Le resta puntos a un conductor tras una infracción. Si a un conductor se le quitan más puntos de los que tiene, se quedará con 0 puntos. En caso de que el conductor no exista, lanza un error "Conductor inexistente".
   - `consultar(dni)`: Devuelve los puntos actuales de un conductor. En caso de que el conductor no exista, lanza un error "Conductor inexistente".

- `cuantos_con_puntos(puntos)`: Devuelve cuántos conductores tienen un determinado número de puntos. En caso de que el número de puntos no esté entre 0 y 15 lanza un error "Puntos no válidos".

La implementación de las operaciones debe ser lo más eficiente posible. Por tanto, debes elegir una representación adecuada para el TAD, implementar las operaciones y justificar la complejidad resultante.