Universidad Rey Juan Carlos

OPCIÓN A

Estructuras de Datos Avanzadas
Examen Ordinario. Duración 2h
18 de diciembre de 2017

Nombre y apellidos: _____
Firma de comprobación de asistencia al examen:

Normas del examen

- La duración del examen es de 2 horas y media.
- Esta parte práctica puntúa sobre 8 puntos. Los otros dos puntos corresponden a las preguntas de teoría.
- El alumno debe ejecutar la aplicación *ExamWatcher* (disponible en la pestaña de evaluación del Aula Virtual) durante el examen. Tras descargarla y ejecutarla el alumno debe escribir su nombre en el campo correspondiente de la aplicación. Dicha aplicación monitoriza las acciones del usuario y por ello estará activa durante todo el examen. Cuando termine el examen, la aplicación tiene un botón para generar un fichero ZIP con el directorio donde se encuentre el código del alumno. El alumno debe subir dicho ZIP a la actividad examen práctico de enero del Aula Virtual.
- En el Aula Virtual está disponible un archivo ZIP que contiene el esqueleto de los ejercicios del examen así como los tests unitarios correspondientes. Además, contendrá todo el código visto en clase que servirá como apoyo en el desarrollo del examen.

Ejercicio 1 [2 puntos]

La clase `SumLevels` tiene la capacidad de sumar los valores de todos los nodos de unos niveles determinados de un árbol *n*-ario de enteros. Los niveles que se desean sumar se pasarán como argumento al método `sumLevels()` en forma de array de enteros, siendo el primer nivel (raíz) el nivel 0. El resultado debe ser la suma de todos los nodos situados en los niveles contenidos en el array. Para implementar el método `sumLevels()` se permite añadir cualquier atributo, estructura de datos o método a la clase `SumLevels`. Además, no se valorará ni penalizará el uso de recursividad y se podrá utilizar cualquier estructura de datos, método auxiliar o atributo que se considere necesario.

Ejercicio 2 [2 puntos]

Se desea implementar un nuevo tipo de árbol binario el cual, a partir del contenido de un nodo, sea capaz de encontrar dicho nodo con una complejidad $O(1)$. La clase `QuickBinaryTree` que se proporciona debe tener exactamente el mismo comportamiento que `LinkedBinaryTree` (se puede reutilizar el código de la clase original), y un método `search` adicional que permite encontrar un nodo dado su contenido. Implementa la clase `QuickBinaryTree` para que el método `search` tenga complejidad $O(1)$. Ten en cuenta que deberás modificar algunos métodos adicionales de `QuickBinaryTree` para que se mantenga el comportamiento solicitado. Se pide:

a) **[1 punto]** Implementar el método `search` que se ha añadido vacío. Dicho método debe tener una complejidad de $O(1)$.

b) **[1 punto]** Modifica los métodos de inserción y borrado de `QuickBinaryTree` para que se adapten al nuevo comportamiento.

NOTA: los métodos `attach` y `subtree` de `LinkedBinaryTree` no se tendrán en cuenta, pueden ser ignorados.

NOTA 2: se podrá utilizar cualquier mecanismo que se considere oportuno para cumplir con el comportamiento de la interfaz `BinaryTree` (herencia, reutilización de código de `LinkedBinaryTree`, añadir estructuras de datos, etc.)

Ejercicio 3 [4.0 puntos]

Una de las tareas más importantes en el análisis de redes sociales es la identificación de usuarios importantes. Una red social se compone de una colección de usuarios donde algunos de ellos son amigos y otros no. Para saber cuáles de estos usuarios son importantes en un grado k, siendo k un número entero, se sigue el siguiente procedimiento.

1. Se eliminan todos los usuarios de la red que tengan un número de amigos menor que k. Es importante tener en cuenta que la eliminación de un usuario puede hacer que el número de amigos de otro usuario pase a ser también menor que k, por lo que el procedimiento debe repetirse hasta que no quede en el grado ningún usuario con un número de amigos menor que k.
2. Una vez eliminados esos usuarios, es necesario identificar en cuántos grupos se ha partido la red social: si se mantiene formando un solo grupo, entonces los usuarios eliminados no eran importantes. Si por el contrario, la red social queda formada por más de un grupo, los usuarios eliminados eran importantes en grado k. Un grupo se puede definir como un conjunto de usuarios que están conectados entre sí pero no con usuarios de otros grupos.

Para realizar este análisis, se proporciona la clase `SocialNetwork`, la cual debe ser implementada por el alumno. En particular, se pide:

a) **[1 punto]** Definir la estructura de datos necesaria para almacenar la red social e implementar los métodos `addUser`, `makeFriends` y `areFriends`.

b) **[1 punto]** Implementar el método `filter`, el cuál eliminará a todos los usuarios de la red con un número de amigos menor que k, siendo k un parámetro del método. Este método devolverá los usuarios eliminados.

c) **[2 puntos]** Implementar el método `groups`, el cuál indicará en cuántos grupos se encuentra dividida la red social.