Realiza la siguiente secuencia de inserciones en un árbol AVL:

6, 9, 10, 7, 4, 2, 8, 3

¿Quién es el hijo derecho del nodo 3 en el árbol resultante?

Respuesta:

Se dispone de una tabla hash de tamaño 13 que resuelve las colisiones por medio de prueba cuadrática, con c1=0 y c2=1. Si h(k) = k mod TAM_TABLA, donde TAM_TABLA es el tamaño de la tabla hash, indica cuál sería el resultado de la inserción de los siguientes elementos (---- indica que ese hueco queda libre): 5886, 7894, 2345, 6328, 1323, 9877, 4887

t(0) ->

t(1) ->

t(2) ->

t(3) ->

t(4) ->

t(5) ->

t(6) ->

t(7) ->

t(8) ->

t(9) ->

t(10) ->

t(11) ->

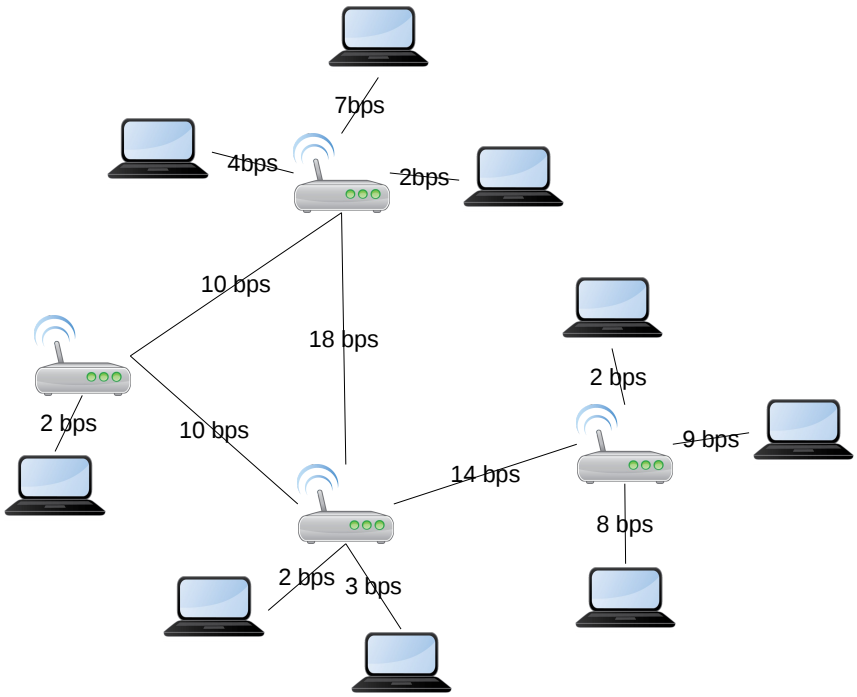
t(12) ->

Estructuras de Datos Avanzadas
Grado en Ingeniería Informática
Examen Ordinario del 18 de septiembre de 2017

A

Ejercicio 1 [3.5 puntos]

La empresa URJC Redes quiere abordar una reestructuración de su infraestructura de red para minimizar costes, maximizando a su vez el ancho de banda de las comunicaciones. Para ello, va a utilizar una topología en la que los routers están conectados entre sí (no necesariamente todos con todos) y los dispositivos terminales sólo se conectan a un router. La siguiente figura presenta de forma resumida un ejemplo de dicha topología. En dicha figura los Routers son objetos de color gris, y los terminales portátiles de color negro.



Antes de hacer la inversión en equipos, el departamento de informática de empresa URJC Redes se compromete a desarrollar una aplicación que le permita simular el funcionamiento de la nueva infraestructura de red. El equipo de desarrollo ya ha implementado:

- La clase `Terminal`, caracterizada por una dirección MAC y una IP (ambas implementadas por un `String`).
- La clase `Router`, con dos MAC y dos IP (un par da servicio a los terminales y otro a los routers)
- La interfaz `Host`, de la que derivan las clases `Terminal` y `Router`.

Sus implementaciones se proporcionan en el proyecto adjunto dentro del paquete `exam.january2016.A.lan`

Adicionalmente, se tiene que poder definir el ancho de banda en cada enlace mediante un número entero que indica los bits por segundo (bps) del enlace.

Se desea dotar de contenido a la clase `NetworkManager`, cuyo esqueleto se proporciona en el proyecto. Dicha clase será una herramienta para la gestión de la simulación.

En particular se pide:

a) [1.5 puntos] **Definir las propiedades privadas** de la clase `NetworkManager`, de tal forma que le permita representar de manera óptima una red. Obsérvese que las redes pueden tener ciclos y que los enlaces son simétricos.

Implementar el método `addRouter` que reciba un objeto de tipo `Router`, la colección de routers a los que se debe conectar, y los anchos de banda de los enlaces con los que se conectará. Como resultado, el método actualizará en consecuencia la red.

Implementar el método `getRouters` que recibirá un router y devolverá los otros routers a los que está conectado.

b) [1.0 punto] **Implementar el método** `addTerminal` que reciba un objeto de tipo `Terminal`, otro de tipo `Router` y el ancho de banda del enlace con el que se conectará el terminal al router. Como resultado, el método actualizará en consecuencia la red.

Implementar el método `getRouter` que recibirá un terminal y devolverá el router al que está conectado.

c) [1.0 puntos] **Implementar el método** `findHops` que determine por cuantos routers hay que pasar para llegar de un terminal a otro. Obsérvese que se dispone del algoritmo `BreathSearch` implementado en la clase `material.graphs`.

NOTA: Se proporcionan test unitarios para comprobar el funcionamiento de cada parte. La clase `NetworkManager` podrá contener en su interior cualquiera de las estructuras de datos estudiadas durante el curso, valorándose especialmente lo adecuado de la elección que se realice. Se valorará negativamente el hecho de no utilizar estructuras de datos auxiliares que reduzcan la complejidad de las operaciones pedidas.

Ejercicio 2 [3.5 puntos]

En el paquete `exam.january2016.A.tree` contiene la clase `WeightedLinkedTree`, que es una copia de la clase `LinkedTree`.

a) [1.5 puntos] **Modificar la implementación de** `WeightedLinkedTree` para que realmente incorpore un peso (número entero) en cada arista (enlace entre un nodo y un descendiente) del árbol. Modificar los métodos `insertLeft e insertRight` para que permita insertar el valor entero asociado a cada descendiente. Añadir el método `getWeight` que reciba dos nodos y devuelva el peso que hay entre los nodos. Ver los test asociados a la clase.

b) [1.0 punto] **Implementar el contenido de la clase** `AscendingWeight` que calcule la suma de pesos de un nodo dado a la raíz del árbol.

c) [1.0 puntos] **Implementar el contenido de la clase** `DescendingWeight` que calcule la suma de pesos mínima de un nodo dado cualquiera de las hojas descendientes.

NOTA: Se proporcionan test unitarios para comprobar el funcionamiento de cada parte. Se valorará un buen diseño (de clases y métodos) para obtener la máxima calificación. Se recuerda que, por lo general, es un mal diseño tener métodos que compartan mucho código. Asimismo, es imprescindible que los métodos implementados sean lo más eficientes y genéricos posible. Las soluciones recursivas no serán penalizadas.

Nombre y apellidos del alumno:_____

Firma de comprobación de asistencia al examen: