

Estructuras de Datos Avanzadas Examen convocatoria adelantada. Duración 2h 21 de septiembre de 2018

Nombre y apellidos:	
Firma de comprobac	ión 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.

Ejercicios de teoría [2 puntos]

Ejercicio | [| punto]

Realiza las siguientes inserciones en un árbol rojo-negro en el orden indicado: 6, 10, 8, 15, 12, 1, 4, representando todos los pasos relevantes realizados durante las inserciones.

Ejercicio 2 [l punto]

Se tiene una tabla hash con capacidad 10 que utiliza direccionamiento abierto y hashing doble. Muestra el resultado de la tabla hash tras realizar las siguientes inserciones y borrados, teniendo en cuenta que idAlumno es la clave y su nota es el valor.

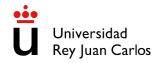
idAlumno	Nota
56	3
73	8
14	4
12	3
7	6
67	5
56	(borrado)
12	7
55	9

Ten en cuenta los siguientes datos:

$$h(k) = k \mod p$$

$$d(k) = q - (k \mod q)$$

$$p = 11, q = 7$$



Estructuras de Datos Avanzadas Examen convocatoria adelantada. Duración 2h 21 de septiembre de 2018

Ejercicios prácticos [8 puntos]

Ejercicio | [2 puntos]

La clase JustParent tiene la capacidad de detectar aquellos nodos de un árbol n-ario que, siendo padres, no son abuelos. Se pide implementar el método justParents() que devuelva aquellos nodos que son padres pero no son abuelos en una colección iterable.

Ejercicio 2 [2 puntos]

Se quiere realizar una nueva implementación de grafos que aporte las siguientes mejoras a la implementación vista en clase:

- a) [I punto] El grafo representado debe ser no dirigido.
- b) [I punto] Reducir la complejidad del acceso a los adyacentes a un vértice. En concreto, se pide realizar las modificaciones oportunas para que obtener los vértices adyacentes a uno dado tenga complejidad O(1).

Ejercicio 3 [4.0 puntos]

Necesitamos organizar el sistema de ficheros de la universidad para que no desaparezcan documentos importantes y asegurar que el acceso a todos los ficheros es eficiente. Para ello, nos han encargado implementar la clase DirInfo. Dado un directorio raíz, la clase almacenará, en la estructura de datos que considere oportuna, todos los ficheros y directorios que existen dentro de la misma. De cada fichero se necesita almacenar: si es un fichero o un directorio, su tamaño (solo es relevante si es un fichero), si está oculto y la última fecha de modificación. Se pide:

- a) [I punto] Definir la estructura de datos necesaria para almacenar todos los ficheros y directorios contenidos en un directorio dado.
- b) [I punto] Implementar el método giveMeFiles(), que devolverá todos los ficheros almacenados ordenados por tamaño (de menor a mayor).
- c) [2 puntos] Implementar el método distance(f1, f2), que dados dos ficheros o directorios fl y f2, devolverá el número de directorios intermedios que existen entre ellos.

Se valorará la eficiencia de las estructuras de datos utilizadas, y se penalizará.

NOTA: todos los métodos necesarios para acceder a los atributos de un fichero están disponibles en la clase File, propia de Java.