Estructuras de Datos Avanzadas
Grado en Ingeniería Informática

Parte práctica del examen ordinario del 24 de enero de 2022

Normas generales:

- La duración del examen es de 2 horas (más 10 minutos para iniciar el entorno informático).
- Esta parte práctica puntúa sobre 7 puntos. Los otros 3 puntos del examen corresponden al test.
- Desde el campus virtual hay que descargar un archivo ZIP que contiene el **esqueleto** de los dos ejercicios del examen así como los test unitarios correspondientes. Subir el proyecto al disco R (desde myApps). A continuación abrirlo desde la aplicación ApacheNetbeans desde el escritorio de desarrollo. En caso de que esto de problemas se puede crear un proyecto en ApacheNetbeans vacío; copiar dentro de la carpeta del proyecto la carpeta src y la carpeta test proporcionadas; añadir las librerías JUnit y Hamcrest; y finalmente cerrar y volver a abrir ApacheNetbeans.

Ejercicio 1 Creación de estructura de datos [2 puntos]

Se desea implementar un nuevo tipo de árbol genérico, el cual, a partir del contenido de un nodo, sea capaz de encontrar dicho nodo con una complejidad media W(1). Se proporciona la clase QuickTree que inicialmente tiene exactamente el mismo comportamiento que LinkedTree. Se pide:

a) Implementar el método search que se ha añadido vacío. Dicho método debe permitir encontrar el Position de un nodo dado su contenido. Implementa el método search de manera que tenga complejidad media W(1) modificando también los métodos de inserción (addRoot, add y add con índice) si fuese necesario. [1.25 puntos]

b) Implementar el método remove para que la estructura de datos se comporte de manera coherente ante los borrados. [0.75 puntos]

NOTA 1: Los métodos attach y subtree de LinkedTree no se tendrán en cuenta y pueden ser ignorados.

NOTA 2: Siempre que se respete la interfaz de NAryTree y la del método search, se permite modificar su implementación como se crea necesario: añadiendo las estructuras de datos secundarias que se necesiten, añadiendo métodos privados, eliminado la herencia y copiando código...

Ejercicio 2 – Aplicación [5 puntos]

La compañía aérea URJCFlights desea desarrollar un sistema que le permita gestionar sus vuelos a lo largo del mundo. Como se trata de una empresa en expansión desea poder añadir nuevos destinos con facilidad. Además, el sistema se debe encargar de proporcionar itinerarios viables a los viajeros desde un aeropuerto a otro, siempre que la compañía opere en ellos, priorizando vuelos directos. En cada aeropuerto habrá un número de vuelos concretos de la compañía.

Los vuelos tendrán, además de un origen y un destino, una hora de salida.

Se pide:

a) Implementar las estructuras de datos necesarias para que la clase `URJCFlights` funcione correctamente. Se debe dotar del contenido necesario para el correcto funcionamiento del proyecto a las clases `Airport` y `Flight`. [0.5 puntos]

b) Implementar el método `newAirport` que inserta un nuevo aeropuerto desde el que la compañía `URJCFlights` estará autorizada a operar. [0.25 puntos]

c) Implementar el método `newConnection` que añade nuevas conexiones a un aeropuerto que ya forma parte de la compañía. Para ello recibirá una lista de pares `Airport-Integer`, que representa la conexión que se establece entre aeropuertos y la distancia entre ellos. Este método no añade vuelos, solo la capacidad para poder viajar de un aeropuerto a otro. Por ejemplo si a Madrid, le pasan el par París-1103, la compañía guardará que es posible añadir vuelos de Madrid a París (y de París a Madrid) y que la distancia entre ambos aeropuertos es 1103 KM, pero hasta que no se añadan vuelos no se podrá volar entre estos aeropuertos. Si la lista contiene algún aeropuerto con el que la compañía no opera se deberá lanzar una excepción. [0.75 puntos]

d) Implementar el método `newFlight` que dado un aeropuerto y un vuelo, añade el vuelo al aeropuerto siempre que exista conexión con el aeropuerto destino del vuelo. [0.5 puntos]

e) Implementar el método `availableAirportsConnection`, que devuelve la lista de aeropuertos con los que puede operar el aeropuerto que se le ha pasado como parámetro. [0.5 puntos]

f) Implementar el método `availableFlights`, que dado un aeropuerto devuelve la lista de todos los vuelos cuyo origen sean dicho aeropuerto. [0.25 puntos]

g) Implementar el método `searchItinerary`, que recibe un aeropuerto origen y otro destino y devuelve todos los vuelos que habría que tomar para llegar desde el aeropuerto origen al de salida. En caso de que no sea posible viajar entre estos aeropuertos, porque no exista conexión entre ellos o no existan vuelos que los comuniquen en el itinerario, el método devuelve `null`.[2.25 puntos]

NOTA 1: Se recomienda hacer uso del método `getPath` de la clase `BradthSearch` para realizar la operación `searchItinerary`.