Actividad 7. TAD Map

Objetivo

Conocer el TAD MAP como estructura adecuada para almacenar una colección de objetos clave/valor y saber utilizar un Map para resolver problemas de grafos.

Procedimiento

Para resolver esta actividad se seguirá la *técnica del rompecabezas*.

1. Todos los alumnos deben formarse en el diseño del TAD Map (revisar las transparencias que sobre el TAD Map están disponible en faitic, Documentos e Ligazóns/Teoría/Map.rar, que incluye lecturas recomendadas). Trabajo individual, no presencial.
2. Formar grupos de 3 personas, donde cada miembro tendrá un rol específico. El profesor tomará nota de la composición de los grupos en horas presenciales de grupo reducido (7, 8 o 9 de noviembre).
3. Tras la preparación inicial e individual se reunirán los alumnos del grupo para la puesta en común y unificación de conceptos (resolviendo dudas si las tienen). Trabajo grupal y presencial en grupo reducido (7, 8 o 9 de noviembre). Como resultado de esta reunión el experto en TAD Map subirá a faitic la especificación acordada, nombrando el fichero como **"Especificacion_Map_nombre de los tres alumnos.rar"**. Una vez que el profesor da el visto bueno, el alumno responsable de esta parte comenzará con la implementación (Ver ejercicio 1).
4. A partir del 17 de noviembre, los otros dos miembros del grupo deben formarse en uno de los algoritmos sobre grafos propuesto por el profesor (Dijkstra y Número cromático). Para ello deben revisar las transparencias sobre grafos y esquemas algorítmicos que están disponibles en faitic, Documentos e Ligazóns/Teoría/Grafos.pdf y Documentos e Ligazóns/Teoría/EsquemasAlgoritmicos.pdf. Para la clase de grupo reducido deben traer una propuesta de implementación de dichos algoritmos (Ver ejercicios 2.a y 2.b). Trabajo individual no presencial.
5. Con la propuesta de resolución de los algoritmos, los expertos de cada grupo se reunirán para la puesta en común (21, 22 o 23 de noviembre). Posteriormente cada experto explicará a los miembros de su grupo su algoritmo. Trabajo grupal y presencial en grupo reducido. Una vez implementado el algoritmo se subirá a faitic; cada uno sube el algoritmo en que es experto. Nombre de los ficheros **"Dijkstra_nombre de los tres alumnos.rar"** y **"Cromático_nombre de los tres alumnos.rar";** además el otro miembro del grupo debe subir la implementación del TAD Map con el siguiente nombre **"TAD_Map_nombre de los tres alumnos.rar"**. Fecha tope para subir estos documentos 11 de diciembre.

Evaluación

Evaluación grupal:
Cada grupo será llamado por el profesor para explicar alguno de los algoritmos implementados y/o el TAD Map (12, 13 o 14 de diciembre).

Evaluación individual:
Mediante una prueba el día 15 de diciembre y que representa el 20% de la nota final.

Tiempo estimado

8 horas

Ejercicios

1.- Especifica e implementa el **TAD MAP<K,V>.** Para la implementación, desarrolla un proyecto en java que contenga la interfaz Map<K,V> y una clase que implemente dicha interfaz haciendo uso de una Tabla Hash.

La **tabla hash** se utiliza para almacenar la colección de objetos <clave, valor> y debe utilizar la estrategia de *encadenamiento abierto o separado* (un array de listas) para resolver el problema de las colisiones. Para trabajar con listas se puede usar el TAD Lista de la librería *aed1.jar* o la interface *List* de java.

Durante el proceso de búsqueda, inserción o borrado de un elemento de la tabla hash, el primer paso a realizar es transformar la clave en un índice del array, es decir, aplicar una función hash. Este índice indicará la lista en la que hay que buscar, insertar o eliminar el elemento correspondiente. Esta función debe implementarse mediante un método privado que, para esta actividad, codificará el *método de división*:

private int funcionHash (K clave)

Un problema de esta función (método de división) es que para poder aplicarla es necesario que el objeto que se pasa como parámetro sea de tipo int. Para resolver este problema existe en java el método **hashCode()**, que convierte cualquier objeto en un int. Es decir, **clave.hashCode()** convierte el parámetro de entrada de la función a un valor entero, lo cual permite trabajar con cualquier objeto en la tabla hash.

Una vez realizada la implementación del TAD MAP<K,V>, prueba su correcto funcionamiento haciendo uso del framework JUnit y de la clase de prueba *HasHMapTest.java* que se proporciona en faitic, *Documentos e Ligazons/Actividades/Test (JUnit)*.

2.- Haciendo uso del TAD Map del ejercicio 1, implementa los algoritmos sobre grafos que se indican a continuación:

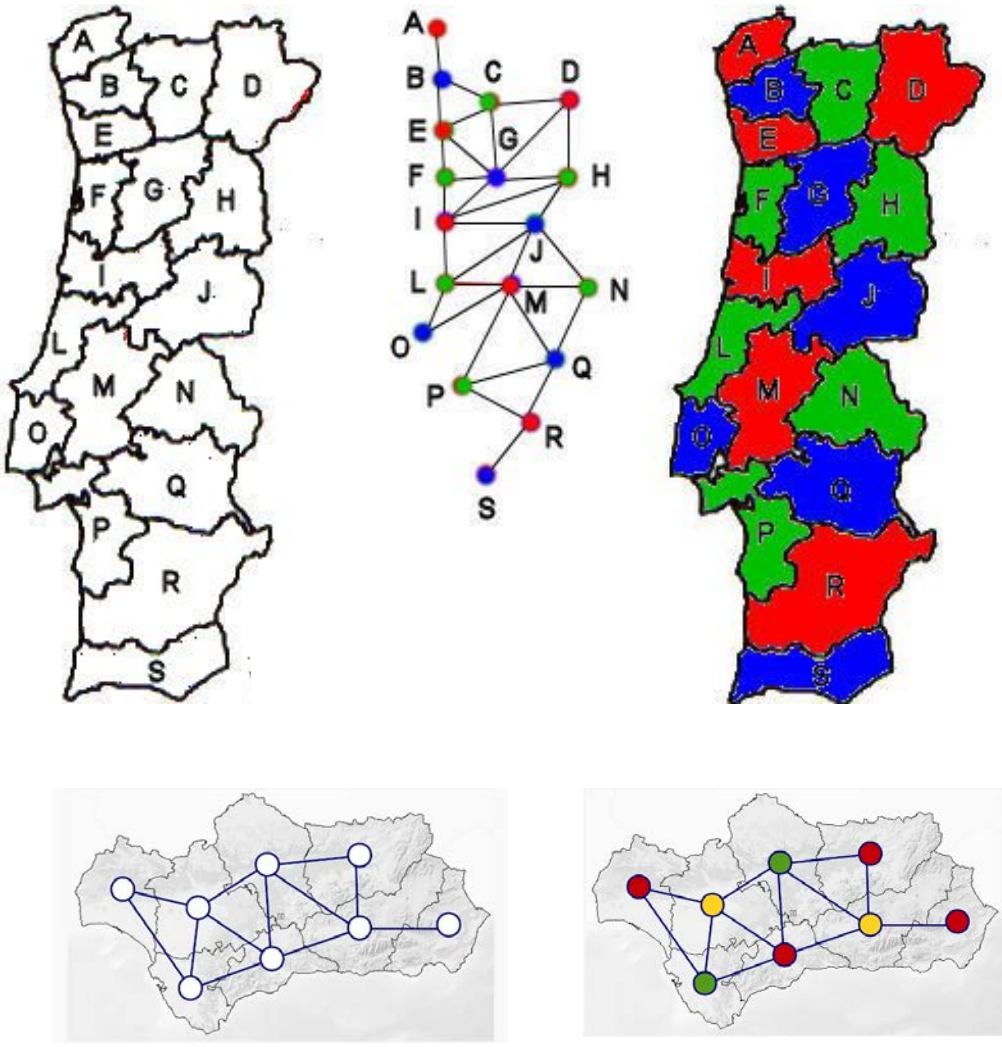
a) Escribe el **algoritmo de Dijkstra**, también llamado **algoritmo de caminos mínimos**, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo dirigido y con pesos en cada arista.

La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.

```
public static <E> Map<Vertice<E>,Integer> dijkstra(Grafo<E,Integer> g,
Vertice<E> v)
```

b) El **algoritmo de número cromático** indica el número de colores mínimo necesario para colorear los vértices de un grafo. El *teorema de los cuatro colores* justifica que el número cromático de un grafo plano es menor o igual que cuatro.

El objetivo de este ejercicio consiste en, dado un grafo plano que representa un mapa político, indicar cómo deben colorearse cada uno de los vértices de dicho grafo, empleando el menor número posible de colores (según el teorema citado anteriormente como máximo cuatro).




```
public static <E> Map<Vertice<E>,String> colorearMapa (Grafo<E,Integer> g,
String [] colores)
```

Una vez realizada la implementación de los algoritmos, prueba su correcto funcionamiento haciendo uso del framework JUnit y de la clase de prueba SolucionActividad7*Test.java* que se proporciona en faitic, *Documentos e Ligazons/Actividades/Test (JUnit)*.