Master DCISS – Université Grenoble Alpes
Jérôme David

TP Représentation de graphes étiquetés et plus court chemin

Le but de ce TP est de représenter un graphe étiqueté, et d'implémenter l'algorithme de recherche du plus court chemin. Cela peut servir par exemple à calculer l'itinéraire le plus court entre deux villes.

Pour cela, il faut définir les classes données par la suite. Pour toutes les classes, vous redéfinirez la méthode `toString()` pour avoir une représentation sous forme de chaîne de caractères significative.

`Sommet` : Un sommet est représenté par un nom. Deux sommets sont égaux si ils ont le même nom. Un sommet maintient l'ensemble de ses voisins ainsi que la distance à ses voisins.
Cette classe dispose entre autres des constructeurs et méthodes suivants :

- `public Sommet(String nom)`
  - Créer un sommet avec le nom passé en paramètre et aucun voisins.
- `public void ajouterVoisin(Sommet voisin, int distance)`
  - Ajoute une arrête entre le sommet et le voisin passé en paramètre. L'arête est étiqueté avec la distance passée en paramètre.
- `public int distance(Sommet s)`
  - Retourne la distance entre le sommet courant et celui passé en paramètre. Si le sommet s n'est pas voisin alors -1 est retourné.
- `public Iterator<Sommet> voisins()`
  - Retourne un itérateur sur les voisins de ce sommet.
- `public String getNom()`
  - Retourne le nom du sommet

`Chemin` : Représente un chemin dans le graphe ainsi que sa longueur (somme des distances entre deux sommets successifs du chemin). Deux chemins sont comparables en fonction de leur longueur puis du nombre de sommets qui les composent, puis finalement du nom des sommets. On peut également itérer sur les sommets du chemin (cette classe devra implémenter Iterable<Sommet>)
Cette classe dispose entre autre des constructeurs et méthodes suivants :

- `public Chemin(Sommet debut)`
  - Créer un chemin de longueur 0 qui ne contient que le sommet passé en paramètre.
- `protected Chemin(Chemin debut, Sommet suite)`
  - Créer un chemin qui est la concaténation du chemin et du sommet passés en paramètre. Si le sommet n'est pas un voisin du dernier sommet de chemin début alors une exception sera levée. La longueur de ce chemin devra être mise à jour.
- `public Sommet arrivee()`
  - Retourne le dernier sommet de ce chemin.
- `public Collection<Chemin> etendre()`
  - Méthode qui retourne la collection des chemins qui est le résultat de la concaténation de ce chemin avec les voisins du dernier sommet de ce chemin. Attention, cette méthode doit d'assurer de ne retourner que des chemins sans cycle (i.e. un sommet n'est présent

qu'une seule fois dans le chemin).

`Graphe` : Il maintient l'association entre le nom des sommets et l'objet qui les représente. Cette classe dispose des constructeurs et méthodes suivants :

- `public Graphe()`
  - Créer un graphe vide
- `private Sommet getOrCreate(String nomSommet)`
  - Retourne le sommet qui possède le nom passé en paramètre. Si aucun sommet n'a ce nom, alors un nouveau sommet avec ce nom est créé.
- `public void ajouterArete(String nomSommet1, String nomSommet2, int distance)`
  - Ajoute une arête étiquetée par la distance entre les deux sommets portant les noms passés en paramètre.
- `public Chemin plusCoursChemin(String nomSommetDebut, String nomSommetFin)`
  - Algorithme du plus cours chemin. Pour cela, vous implémenterez la méthode de recherche du plus court chemin de Dijkstra (http://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra). Vous réaliserez cette méthode en dernier et vous pourrez vous appuyer sur l'exemple donné dans cette page.