C.U.F.R. J-F. Champollion — L2 Informatique S5
Albi — 2015

TP 6 (1 séance)
Arbre couvrant de poids minimum

Le but de ce TP est de mettre en œuvre l'algorithme de Kruskal permettant la construction d'un arbre couvrant de poids minimum.
Les plus rapides et les plus motivés pourront réfléchir à une mise en ouvre de l'algorithme de Prim.

Le TP sera illustré à l'ide du graphe G6, donné par sa liste d'arêtes valuées :
ListAretes=[{{1,2,7},{1,5,6},{1,6,2},{2,3,4},{2,5,5},{3,4,1},{3,5,2},{4,5,3},{5,6,1} }

1. Classe GrapheNonOrienteValue

Créer la classe GrapheNonOrientéValue héritant de la classe *GrapheNonOrienteList* permettant de définir un graphe non orienté valué.
Les attributs spécifiques à cette classe sont :
private int[][] Poids; // La matrice des poids où P[i][j] est le poids de l'arc i->j
private ArrayList<int[]> ListeArete; // La liste des arêtes sous forme d'un triplet d'entiers {i,j,poids(i,j)}

Ecrire les deux constructeurs suivants :

- *public GrapheNonOrienteValue(int nb) :* construisant un graphe à nb sommets sans arêtes.
- *public GrapheNonOrienteValue(int nb,int[][] T) :* qui à partir d'un vecteur d'arêtes valuées de la forme {i,j,poids(i,j)} construise les attributs du graphe (donc ses attributs en tant que *GrapheNonOrienteList* plus ses attributs spécifiques en tant que *GrapheNonOrienteValue.*

Ecrire les méthodes d'affichage suivantes :
- *public void affichePoids() :* qui affiche la matrice des poids
- *public static void afficheAretes(ArrayList<int[]> L)* qui affiche la liste des arêtes
- public void affiche() : qui appelle les deux affichages précédents.

2. Opération sur les graphes représentés par liste d'arêtes

La mise en ouvre de l'algorithme de Kruskal nécessitera les méthodes suivantes déjà écrites dans la classe *GrapheNonOrienteList :*

- public void ajoutArete(int i, int j) : qui ajoute une arête à un graphe non orienté (TP1)
- public void enleveArete(int i, int j): qui enlève une arête à un graphe non orienté (TP1)
- public boolean isCyclic() : qui teste si un graphe non orienté est cyclique (TP3 application des parcours)

3. Mise en œuvre de l'algorithme de Kruskal

L'algorithme de Kruskal est bien adapté à une représentation par liste d'arêtes. Cette Liste peut être triée dans l'ordre croissant des poids d'arête par la méthode fournie :

public static ArrayList<int[]> triInsertion (ArrayList<int[]> L)

L'algorithme consiste alors simplement à ajouter les arêtes (dans l'ordre de leur poids), à condition qu'elle ne provoque pas de cycle. L'algorithme s'arrête quand on a ajouté n-1 arêtes. On a alors un graphe sans cycle à n sommets et n-1 arêtes donc un arbre et les arêtes ayant été choisies dans l'ordre croissant des poids, il est de poids minimal.

Ecrire la méthode :

- *public GrapheNonOrienteList Kruskal()* : mettant en œuvre cet algorithme.