Kruskal et Prim : manipulation des graphes

Binome : Olivio Mariano et Gwenael Gervais **Version finale :** 9 Mars 2007

Le but de ce programme est de manipuler un arbre couvrant de poids minimal.
Étant donné un graphe non orienté et connexe, un arbre couvrant de ce graphe est un sous-ensemble qui est un arbre et qui connecte tous les sommets ensemble. Un graphe peut comporter plusieurs arbres couvrants différents. On peut associer un poids à chaque arête, ce qui est un nombre qui représente le coût de cette arête, et prendre la somme des poids des arêtes de l'arbre couvrant.
Un **arbre couvrant de poids minimal** est un arbre couvrant dont le poids est plus petit ou égal à celui de tous les autres arbres couvrants du graphe.
Un graphe non orienté et général possède une forêt couvrante de poids minimal.

Un problème connu utilisant l'arbre du poids minimal est le suivant :
– On génère n points aléatoirement dans un carré de coté 1
– On génère le graphe complet dont les sommets sont les points générés
– On résout le problème de l'arbre de poids minimal
– On calcule le poids total de l'arbre

Le problème de l'arbre de poids minimal est résolu principalement avec deux algorthmes classiques : *Kruskal* et *Prim* .

1 Kruskal

Quand on travaille sur un graphe connexe, certains problèmes obligent à transformer ce graphe en un arbre (graphe sans cycle) qui contient tous les sommets du graphe et quelques arêtes. On dit alors qu'on a un arbre recouvrant du graphe. Parfois, lorsque le graphe est valué, il s'agit de chercher un arbre recouvrant de poids minimum, c'est-à-dire dont la somme des poids est minimale. *Exemples :*

– Simplifier un cablage
– Supprimer les liaisons maritimes les moins rentables en preservant l'accessibilité aux differents ports.

L'algorithme consiste à d'abord ranger par ordre de poids croissant les arêtes d'un graphe, puis à retirer une à une les arêtes selon cet ordre et à les ajouter à l'arbre couvrant minimal cherché tant que cet ajout ne fait pas apparaître un cycle dans l'arbre couvrant.

2 Prim

L'algorithme consiste à choisir arbitrairement un sommet et à faire croître un arbre à partir de ce sommet. Chaque augmentation se fait de la manière la plus économique possible. L'étape d'initialisation consiste à choisir au hasard un sommet. Au bout de la première

étape, on se retrouve ainsi avec un arbre contenant 1 sommet et 0 arête. Ensuite, on construit récursivement l'arbre minimal de la façon suivante : à l'étape n, ayant déjà construit un

arbre contenant n sommets et n-1 arêtes, on établit la liste de toutes les arêtes liant un sommet de l'arbre à un sommet qui n'est pas sur l'arbre. On choisit alors une arête de poids minimal, que l'on rajoute à l'arbre ; l'arbre contient à présent n+1 sommets et n arêtes. L'algorithme se termine lorsque tous les sommets du graphe sont contenus dans l'arbre. L'algorithme de Prim implémente un Tas Min. Le projet possède deux fichiers à part pour cette dernière implémentation :
– `tas.h`
– `tas.h`

3 Structure du projet

Le dossier du projet inclut en particulier :

– **Les fichiers sources :**
`main.c`
`graphe.c`
`listesChainees.c`
`fctMenu.c`
`kruskal.c`
`prim.c`
`tas.c`

– **Le fichier de la doc :**
`intro.tex`
Le fichier *marianocommands.tex* n'est qu'un ensemble de commandes personalisés LaTeX utilisé dans le fichier principal de la doc.

– **Les fichiers *.h :**
`listesChainees.h`
`fctMenu.h`
`graphe.h`
`kruskal.h`
`prim.h`
`tas.h`

– **Le fichier** *Makefile* : l'execution de ce fichier compilera les sources pour rendre disponibles :

1. un fichier executable *main*
2. deux fichiers libraries incluant le *libtypeadj* et le *libtypeliste*
3. ce fichier *Intro.pdf*

Utilisation de l'executable :

1. Lancement :

   ```
   $ ./main
   ```

2. Un premier menu s'affiche :

```
PROJET AC 2 : KRUSKAL & PRIM
############## Menu ##############

 1 - Creation d'un graphe
 2 - Chargement d'un graphe
 9 - Sortir de l'application
```

3. On choisi entre les trois options et on suit les instructions.

Le menu principal : Un menu pour la manipulation d'un graphe et pour le lancement des algorithmes s'affiche à l'écran :

1. ```
   # nombre maximum de sommets
   7
   # sommets : voisins
   1 : 2/1, 6/5
   2 : 1/1, 6/4, 4/8
   3 : 5/4, 7/1
   4 : 2/8, 5/4, 7/5
   5 : 4/4, 3/4
   6 : 1/5, 2/4
   7 : 4/5, 3/1

   PROJET AC 2 : KRUSKAL & PRIM
   ############## Menu ##############

    1 - Creation d'un graphe
    2 - Chargement d'un graphe
    3 - Insertion d'un sommet
    4 - Suppression d'un sommet
    5 - Insertion d'une arete
    6 - Suppression d'une arete
    7 - Application algorithme Kruskal
    8 - Application algorithme Prim
    9 - Suppression du graphe
    11 - Sortir de l'application
   ```

2. De façon interactive, l'utilisateur pourra observer l'état du graphe crée :

Chargement et enregistrement : Le fichier à charger doit être dans le dossier de l'executable pour être correctement traité. Ce fichier doit aussi être syntaxiquement valable. L'enregistrement s'effectue par defaut dans le fichier contenant l'executable.

Enregistrement postscript (`.ps`) : Lors du lancement des algorithmes de *Kruskal* et *Prim* l'application demandera si l'utilisateur désire un affichage graphique du graphe en question. Cela va se faire dans un fichier `Adobe Postscript` . Il sera nécessaire de rentrer les coordonnés des différentes arrêtes pour obtenir un affichage optimal.
   ```

Compatibilité : Ce projet a été testé sous :
– Mac OS X 10.4.9 Developer Tools
– Linux Redhat Fedora Core 2.6.121.1398FC4
Il est demandé de posseder un compilateur *gcc* et *pdflatex* valables pour compiler correctement ce projet.

Bugs connus : Aucun