INFO-F-203 - Rapport

Projet 1

Yahya Bakkali
Matricule : 445166

Maxime Hauwaert
Matricule : 461714

Date : Novembre 2018

Table des matières

1 Introduction générale

Ce projet a pour but de mettre en pratique des concepts sur les graphes vus au cours d'algorithmique 2 pour une meilleure compréhension et maîtrise de ceux-ci.

2 Sous-arbre de poids maximum

2.1 Introduction

Dans ce problème nous manipulons des arbres constitués de noeuds ayant un poid. Le problème consiste à transformer un arbre $T = (V, E)$ en arbre $T' = (V', E')$ de façon à maximiser la fonction

$$w(V') = \sum_{v \in V'} w(v)$$

2.2 Choix d'implémentation

Nous avons décidé de ne pas modifier l'arbre de départ mais de créer une liste qui contiendra le nom de tous les noeuds à désactiver, pour qu'à l'affichage on puisse voir l'arbre de départ avec les noeuds activés (en rouge) ainsi que ceux désactivés (en gris).

2.3 Algorithme

Algorithme 1 maxContribution

```
Require: liste noeuds_à_desactiver
1: poid_total = noeud.poid
2: for chaque enfant du noeud do
3:    if enfant.maxContribution()<= 0 then
4:       ajout enfant à noeuds_à_desactiver
5:    else
6:       ajout enfant.maxContribution() à poid_total
7:    end if
8: end for
9: return poid_total
```

La complexité de cet algorithme est de $0(n)$ car il parcourt chaque noeud de l'arbre, $O(n)$ et toutes les opérations faites sur chaque noeud sont en $O(1)$. Donc la complexité finale est de $O(n)$.

2.4 Arbres aléatoires

Cette génération aléatoire d'arbre a une très bonne (Fucking word I forgot). Tous les arbres sont possibles. Il y a de 1 à n noeuds qui composeront l'arbre, 'n' étant 15 dans ce projet. Chaque noeud choisira tout simplement de qui il veut être l'enfant parmis les noeuds déja placés.

3 Les hypergraphes et hypertrees

3.1 Introduction

3.2 Choix d'implémentation

3.3 Algorithmes

Algorithme 2 find_cliques

Require: R,P,X

```
1: if P et X sont vides then
2:    if La clique R est de taille >= 2 then
3:       ajouter R a la liste des cliques
4:    end if
5: else
6:    pivot = élement aléatoire de l'ensemble P ∪ X
7:    for chaque sommet S dans l'ensemble P \ {sommets liés au pivot} do
8:       newP = P ∩ {sommets liés à S}
9:       newR = R ∪ {S}
10:      newX = X ∩ {sommets liés à S}
11:      find_cliques(newP,newR,newX)
12:      P = P \ {S}
13:      X = X ∪ {S}
14:   end for
15: end if
```

Algorithme 3 is_chordal

```
1: unnumbered = ensemble des sommets du graphe
2: s = sommet choisi aléatoirement dans unnumbered
3: unnumbered = unnumbered \ {s}
4: numbered = {s}
5: while unnumbered ! = {∅} do
6:    Vertex = le sommet de unumbered qui a le plus de connection aux sommets dans numbered
7:    unnumbered = unnumbered - Vertex
8:    numbered = numbered + Vertex
9:    clique_wanna_be = {sommets liés à Vertex} ∩ numbered
10:   subGraph = Un sous-graphe induit des sommets appartenant à clique_wanna_be
11:   if le subGraph n'est pas complet then
12:      return False
13:   end if
14: end while
15: return True
```

Algorithme 4 Algorithm_X

```
Require: Matrice M
1: if M est vide then
2:    Ajouter la solution partielle aux solutions finales
3: else
4:    c = collone de M contenant un minimum de 1
5:    l = ligne de M tel que M_{l,c} = 1
6:    Ajouter l à la solution partielle
7:    for chaque colonne j tel que M_{l,j} = 1 do
8:       for chaque ligne i tel que M_{i,j} = 1 do
9:          Supprimer la ligne i de M
10:      end for
11:      Supprimer la colonne j de M
12:   end for
13:   Algorithm_X(M)
14: end if
```

3.4 Hypergraphes aléatoires

4 Librairies utilisées

4.1 Numpy

C'est une librairie très utile dans l'utilisation d'opérations mathématiques tel que les fonctions sinus/cosinus, etc et dans la manipulation de l'aléatoire.

4.2 Matplotlib

C'est une librairie assez utile dans l'affichage d'objet mathématique en 2D tel que des cercles, des lignes, des graphiques, etc.

4.3 Copy

C'est une librairie permettant de copier l'intégralité d'un objet sans qu'il n'y ait de liens entre l'ancien et le nouvel objet.

5 Conclusion

What do you even wanna say? (Group project?)