BIG DATA
TP - Données massives

Nicolas Lagaillardie

22 octobre 2018

1 Objectif

Le but du TP est de déterminer si le graphe (orienté) des catégories de Wikipedia contient des circuits. Il sagit décrire un programme qui lit les données et en fait une représentation en mémoire, puis implémente l'algorithme de Tarjan.

1.1 Définitions

Circuit Ensemble de sommets et d'arêtes tel qu'il peut y avoir plusieurs fois le même sommet, mais au plus une fois la même arête.

Composante fortement connectée En théorie des graphes, une composante fortement connexe d'un graphe orienté G est un sous-graphe de G possédant la propriété suivante, et qui est maximal pour cette propriété : pour tout couple (u, v) de nœuds dans ce sous-graphe, il existe un chemin de u à v.

Algorithme de Tarjan En théorie des graphes, l'algorithme de Tarjan permet de déterminer les composantes fortement connexes d'un graphe orienté. L'algorithme de Tarjan est de complexité linéaire, comme l'algorithme de Kosaraju, mais a l'avantage de ne faire qu'une passe sur le graphe au lieu de deux.

2 Compréhension des données

Trois graphes nous sont fournis, correspondant aux versions anglaise, française et espagnole de Wikipédia. Je me suis concentré sur la version anglaise, avant de vérifier mon algorithme sur les autres versions.
Sur chacune de ses lignes, le premier fichier du dossier contient les arcs du graphe, et le second fichier, les noms *humains* des nœuds. Le graphe étant orienté, les arcs sont dirigées du premier lien vers le second lien de chaque ligne.
Les commandes suivantes permettent de compter respectivement le nombre d'arcs et de nœuds dans le graphe :

```
$ sort enwiki-20110405-CategoryIdGraph.txt | uniq -c | wc -l

$ sort enwiki-20110405-CategoryIdName.txt | uniq -c | wc -l
```

Dans ce graphe particulier, nous avons 1622626 d'arcs et 674448 nœuds.

3 Choix de la structure de données et implémentation de l'algorithme

J'ai implémenté l'ensemble de l'algorithme en *Python*. Pour la structure des données, j'ai construit une liste d'adjacence selon le principe suivant :

a. Récupération des données du fichier contenant les arcs
b. Pour chacune des lignes, ajouter le tuple dans un tableau **linksList** et mettre à jour l'ID maximal rencontré **maxi**.
c. Récupération des données du fichier contenant les nœuds
d. Pour chacune des lignes, stocker dans l'ordre de recontre le nom *humain* du nœud dans un tableau **URLList**, le numro de cette ligne à l'index [ID du nœud] dans un deuxième tableau **indexList** et l'ID du nœud dans un dernier tableau **vertices**.
e. Création de la liste d'adjacence qui est un tableau **adjacentList** de taille *le nombre d'éléments*, et qui, à chaque ID, va chercher l'index d'insertion dans **indexList** et ajouter l'ID des voisins à partir de **linksList** dans un sous tableau

Pour l'algorithme de Tarjan, j'ai suivi le pseudo-code affiché sur la page Wikipédia qui lui est relatif.

4 Résultats et bilan

Voici les résultats obtenus, avec une moyenne de temps d'exécution de 11 secondes :

nde:	Type	Size	
0	list	2	['Comic_science_fiction', 'Futurama']
1	list	10	['Games', 'Sports', 'Sports_culture', 'Sports_entertainment', 'Enterta ...
2	list	3	['Meals', 'Foods', 'Desserts']
3	list	7	['Fictional_locations', 'Fantasy_worlds', 'Mythopoeia', 'Artificial_my ...
4	list	3	['Fictional_human_races', 'Middle-earth_Men', 'Middle-earth_Rohirrim']
5	list	3	['Dogs', 'Dog_breeding', 'Dog_breeds']
6	list	4	['Ungulates', 'Aquatic_organisms', 'Marine_mammals', 'Cetaceans']
7	list	21	['Arts_in_Vietnam', 'Cinema_of_Vietnam', 'Vietnamese_films', 'Vietnam_ ...
8	list	7	['Star_Trek_species', 'Space_in_fiction', 'Astrobiology', 'Extraterres ...
9	list	12	['Disney', 'Walt_Disney_Company_subsidiaries', 'Film_companies', 'Film ...
10	list	3	['Science_fiction_games', 'Science_fiction_video_games', 'Star_Control ...
11	list	4	['Television_episodes', 'Television_episodes_by_series', 'Star_Trek_ep ...
12	list	2	['Chess_biographies', 'Chess_players']

FIG. 1 – *Résultats pour le Wikipédia anglais*

En étudiant les résultats, on se rend compte qu'il y a bien des circuits, qui sont au nombre de 6138 pour le Wikipédia anglais, et on note bien les liens *humains* qu'il existe entre les différentes pages d'un même circuit. Par exemple les éléments *Meals*, *Foods* et *Desserts* font partie d'un même circuit, tout comme *Chess_ biographies* et *Chess_ players*. Les résultats semblent donc cohérents. Par ailleurs, le temps d'exécution semble très faible au vu de la taille des données, ce qui est un avantage indéniable. En particulier parce qu'aucune autre méthode d'accélération n'a été utilisée : GPU, multi-CPU ou autre. Néanmoins, cette durée d'exécution assez faible est obtenue au détriment d'un important espace mémoire occupé. En effet, dans le fichier *indexList*, une partie de la liste est inutilisée puisqu'il existe moins d'éléments que le plus grand ID. Ainsi, une piste d'amélioration serait de réduire cet espace mémoire occupé, tout en contenant la durée d'exécution totale.