Auteurs :
Tondereau Arthur
Gadacha Amine

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Compte rendu TP4 – Analyse de logs Apache

I. Spécifications

I.1. Spécifications globales

Le programme fourni permet l'analyse de fichier logs Apache, et en particulier de l'accès aux documents auxquels ils font référence. Sans options le programme fournira une liste des 10 documents les plus consultés, avec pour chacun son nombre d'accès.
Les documents du log qui ont pour racine le domaine local (ici l'intranet INSA) sont renommés pour supprimer cette racine.

Certaines options permettent de discriminer quelles activités du log doivent être analysée, d'autres permettent de discriminer quels documents doivent être pris en compte. Le détail de ces options est disponible dans le manuel.

I.2. Spécifications détaillées

Les options sont supportées dans n'importe quel ordre.
Si une option est mentionnée plusieurs fois alors les arguments pris en compte sont ceux du dernier appel, les autres sont ignorés. Si plusieurs options sont potentiellement conflictuelles, celle appelée en dernier a la priorité.
La concaténation des options n'est pas supportée : "-eg" n'est pas reconnu par exemple
Si une option est appelée sans arguments alors qu'il était obligatoire alors le prochain paramètre est considéré comme un argument, ou lève une erreur si cette option était le dernier paramètre [TEST 21]
Si une option n'est pas reconnue, alors le programme s'arrête. [TEST 20]

Option	Description	Test(s) associé	Réponse attendue	statut de retour	Test associé
Pas d'options	Affiche sur la console les 10 des documents les plus consultés, avec leur nombre de hits	Plus de 10 documents dans le log	Les 10 plus consultés sont renvoyé	0	01
		Nombre de docs dans]0:10[	Tous les documents sont renvoyé	0	02
		Log vide	Aucun documents consultés	1	03
		Fichier non existant ou impossible à ouvrir	Erreur d'ouverture	1	05
		Pas de fichier spécifié	Aucun fichier spécifié	1	04
-g nomGraphe.dot	Produit le graphe des accès entre documents, avec le nombre d'accès sur les arrêtes	Plusieurs documents, liés entre eux	Renvoie un graphe ou tous les documents apparaissent connectés entre eux	0	17

		Un seul document	Renvoi un graphe constitué de 2 noeuds: le referer et le document cible		0	18
		Graphe non connecté	affichage normal du graphe		0	19
-e	Exclut les documents avec extension image, css, JS pour les traitements	Presence de fichiers de ces types	Effectue le traitement selon les autres options sans prendre en compte les documents aux extensions prévues dans la description		0	06
		Aucun fichier à exclure	Aucun effet sur le traitement		0	07
		Tous les fichiers à exclure	Aucun documents affiché		1	08
-t heure	Prend en compte les hits présent dans le créneau horaire [horaire;horaire+1[, sans prise en compte du décalage horaire	Presence de hits dans le créneau horaire	Effectue le traitement selon les autres options de tous les fichiers présents		0	09
		Aucun hits présents dans ce créneau horaire	Ne renvoie rien		1	10
		Tous les hits sont présents dans ce créneau horaire	Effectue le traitement selon les autres options, aucun effet		0	11
		Heure n'appartient pas à [0:23]	Arrêt du programme	1		12
-a	Affiche tous les documents du log sur la console, par ordre décroissant de popularité	Nombre quelconque >0 de documents	Affiche tous les documents		0	13
		0 documents	Aucun documents consultés		1	14
-n nombre	Affiche [nombre] documents du log ou le sous graphe correspondant, classés par popularité	Plus de [nombre] documents dans le log	Les [nombre] plus consultés sont renvoyé		0	15
		Nombre de docs dans]0:nombre[	Tous les documents sont renvoyé		0	16

Précisions sur les spécifications des options :

-e : les accès à des documents exclus sont retirés. Tout accès à un document accepté est conservé, même s'il provient d'un document exclu.

-g : Le graphe construit comporte par défaut tous les accès et tous les documents. Il est fortement conseillé d'utiliser -n sur des fichiers volumineux comportant beaucoup de documents, mais -a n'aura aucun effet et ralentira le programme.

-n : la restriction aux n documents les plus consultés est appliquée après tous les autres filtres.

II. Architecture globale

Nous avons utilisé 5 classes différentes et plusieurs structures, en essayant de garantir la séparation sémantique de celles-ci.

- La classe LectureLog lit le fichier de log passé en paramètre pour en extraire les activités. Celle-ci est très indépendante, peu interconnectée avec la classe Analog.

Elle permet de récupérer des activités du log sous la forme d'une structure. Tous les attributs des activités sont stockées même si elles ne sont pas nécessaires dans le cadre de cet outil.

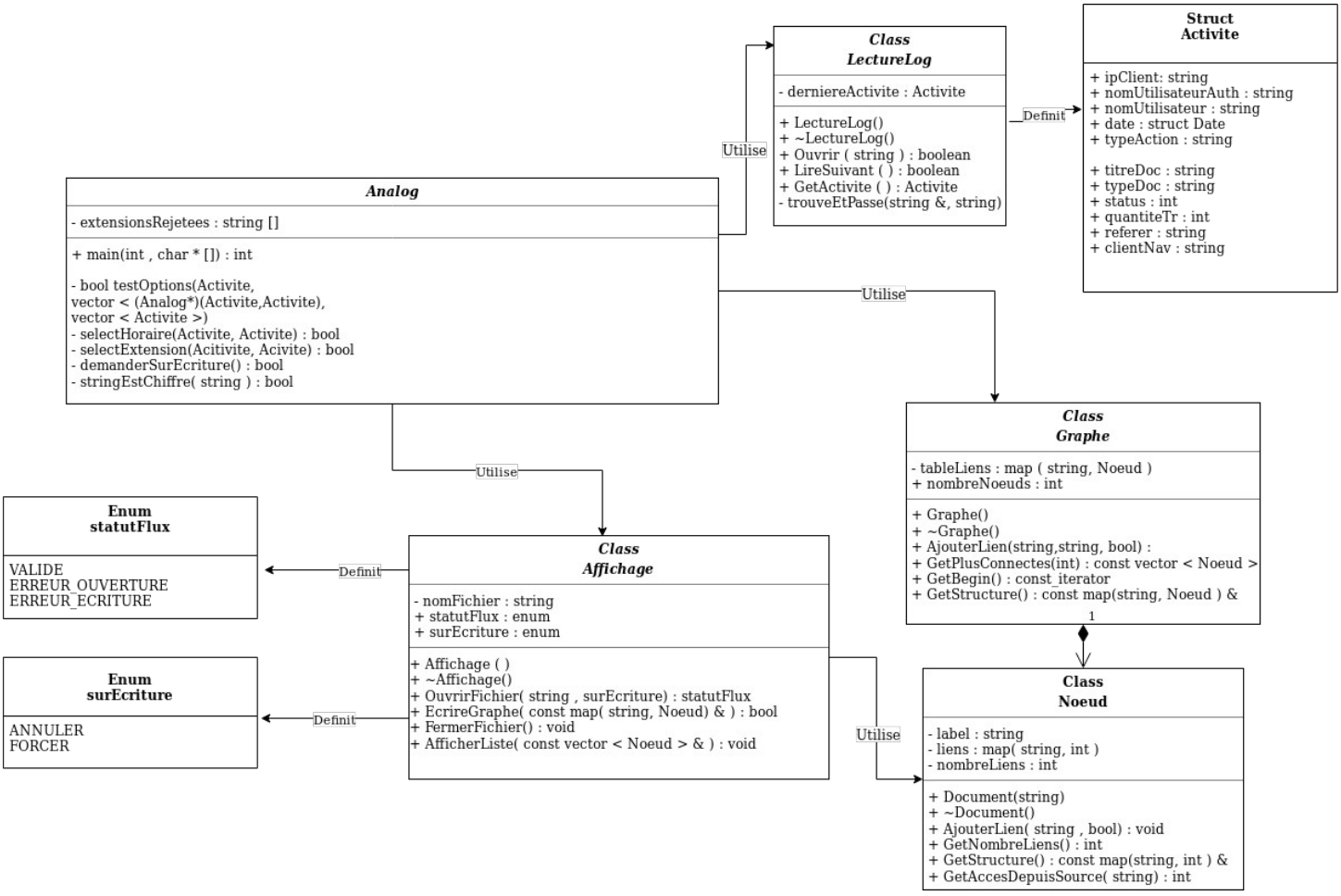
Cela permet une réutilisation facile de cette classe.

- Les classes Graphe et Nœud permettent de stocker un graphe orienté et de le manipuler de manière encapsulée. La définition de deux classes se justifie par la gestion facilitée des métadonnées notamment. La structure de données qu'ils utilisent est décrite dans la partie suivante.

- La classe Affichage permet d'isoler toutes les fonctions de sorties du programme. Ainsi si dans une évolution future on veut passer à un affichage graphique on peut le faire en modifiant uniquement cette classe. Elle définit plusieurs énumérations, pour la gestion de l'ouverture du fichier de sortie dans l'écriture du graphe notamment.

- Analog contient le main. Elle interprète les options en ligne de commande, et est la seule à écrire sur la sortie d'erreur pour relever les différents problèmes possibles.

Les options sont lues séquentiellement pour créer le vecteur de fonctions qui sera utile aux sélections sur les activités. Ce système permet d'ajouter très facilement de nouvelles options, ce dont nous avons profité pour implémenter des fonctionnalités qui nous semblaient intéressantes.



III. Structure de données

III.1. Structure choisies

La solution retenue pour le stockage du graphe d'accès entre les documents est un stockage en listes d'adjacences.

Cela permet d'enregistrer uniquement l'information utile, un stockage en matrice d'adjacence produirait probablement des matrices très creuses : beaucoup de couples de documents n'ont aucun accès entre eux.

On veut donc associer à chaque document la liste des documents depuis lequel il est accédé, avec le nombre d'accès. On enregistre donc pour chacun ses prédéces-seurs dans le graphe, avec les valeurs des arcs associés.

La structure en $list < list < nombre\,d\,'accès >>$ ne convient pas à notre cas d'usage, puisque les clés utilisé sont les titres des documents et non des indices.

On adopte donc une structure de type :

$$map < nom\,Document , map < nom\,Document , nombre\,d\,'acc\grave{e}s >>$$

Pour finir, on veut maximiser la cohérence des classes et faciliter leur réutilisation. On va donc créer une classe Graphe et une classe Noeud, qui permettent d'une part d'encapsuler le fonctionnement de la structure de données et de ses méthodes, et d'autre part de stocker des métadonnées.

On aboutit donc à une structure pour le graphe:

$$Graphe :: map < nom\,Document , Noeud >$$

$$Noeud :: map < nom\,Document , nombre\,d\,'acc\grave{e}s >$$

Les maps utilisés sont des ordered map, qui encapsulent donc un B-arbre. Cela garantit que les opérations se feront en temp logarithmique quel que soit le jeu de donnée que le logiciel doit traiter.

Si la demande de l'utilisateur ne nécessite pas de construire le deuxième niveau des tables associatives (au niveau des Nœuds), alors elles ne sont pas créés et on tient simplement à jour le nombre d'arc entrants dans chaque documents.

La structure générale est schématisée ci contre :

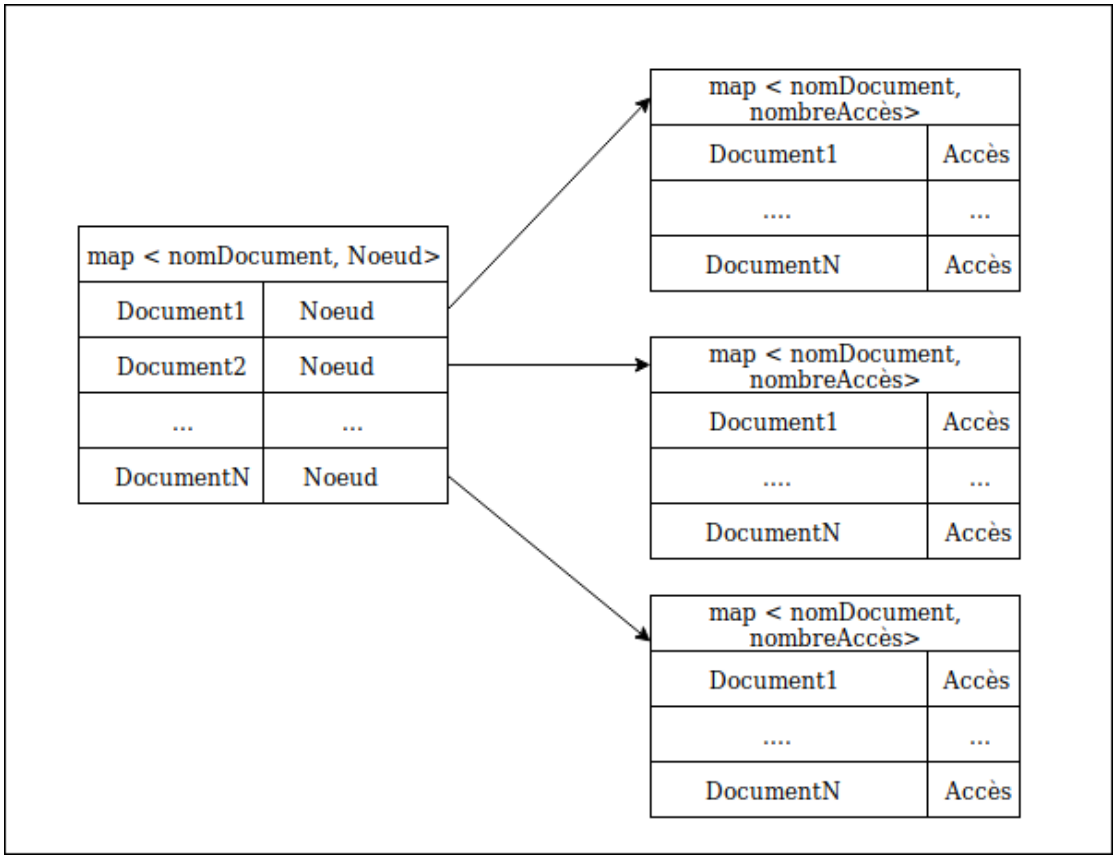


Figure 1: Schéma générale de la structure de donnée utilisée pour le stockage du graphe

Ce qui, appliqué à un exemple, est illustré ici :

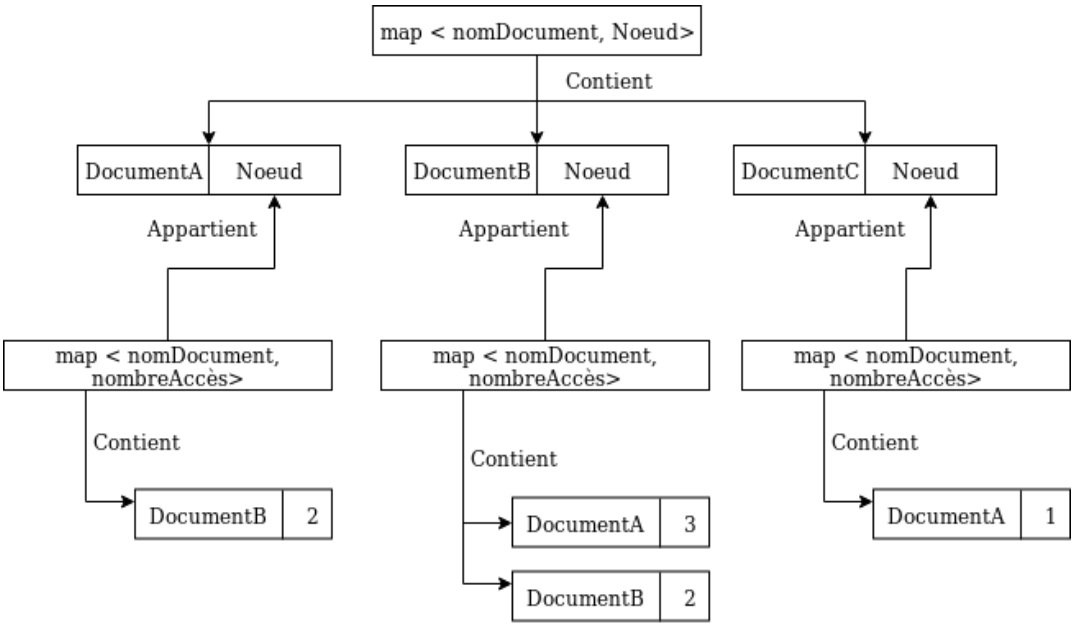


Figure 2: Struture de donnée illustrée pour le graphe 1

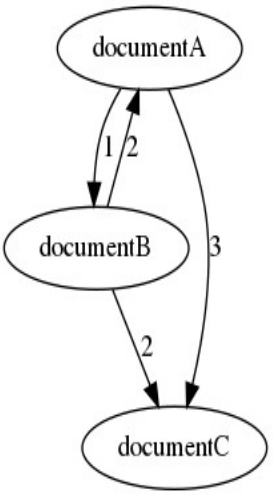


Figure 3: Graphe 1

III.2. Complexité des différentes opérations

On donne n le nombre de nœuds, p le nombre d'arcs, a le nombre de lignes du log

La complexité mémoire n'est pas mentionnée quand elle est égale à O(1), ce qui est le cas de beaucoup d'opérations qui utilisent uniquement la structure

Complexité des opérations unitaires :

- Rechercher la valeur de l'arc entre deux documents :
  Temporelle : $2*O(\log(n))=O(\log(n))$
- Insérer une connexion entre deux documents:
  Temporelle : $2*O(\log(n))=O(\log(n))$
- Extraire et interpréter une ligne du log :
  Temporelle : $O(1)$
- Extraire et interpréter l'ensemble du log :
  Temporelle : $O(a)$
  Mémoire : O(1), puisque l'on conserve une seule activité à la fois
- Construire l'ensemble de la structure du graphe (extraction et insertion) :
  Temporelle : O (a * log(n))
  Mémoire : O(n + p)

Complexité des opérations proposées par l'outil

La complexité mémoire est toujours égale à n+p, soit celle pour construire la structure. Même dans le cas de la duplication avec le tri d'une parti des documents, la nouvelle structure créée ne change pas l'ordre de la complexité.

- Afficher le graphe complet :

Interpréter tout le log + insérer toutes les lignes dans la structure + parcourir la structure, en devant accéder à des attributs des objets :

Temporelle : $O(a)+O(a*2*\log(n))+O(p*2*\log(n)+n)=O(p*\log(n)+n)$

- Afficher tous les documents,triés :

Temporelle : $O((a+n)*\log(n))$

- Extraire uniquement les m documents les plus connectés :

Temporelle : $O(p*\log(n)+n*\log(m))$

- Afficher uniquement un sous graphe :

Temporelle : $O(p*\log(n)+n*\log(m)+m^2)$