Julien Charles-Nicolas
Fatima-Ezzahra Mezidi
Shali Shi

3IF
Groupe 4

Compte rendu
TP C++ n°4

Introduction

Analog est un programme permettant d'analyser un fichier de log Apache. Cet outil génère des statistiques sur le nombre de hits des documents ainsi qu'un document synthétique au format GraphViz(.dot).
Ce compte rendu contient les spécifications de l'application, puis une présentation de l'architecture globale de l'application et une explication détaillées des structures de données utilisées.

Spécifications

- Spécifications générales

L'application analog permet l'analyse d'un fichier de log Apache. Le fichier log est supposé sans erreur de syntaxe. Aucun test pour vérifier la validité du fichier à analyser n'est réalisé. Le fichier doit exister et être libre en lecture pour pouvoir être analysé.
L'application génère des statistiques sur les données du document. Seul les cibles qui ont été atteintes et donc qui ont reçu le code "200" (succès de la requête) sont prises en compte. L'utilisateur peut choisir de ne pas analyser l'ensemble des pages. Il a l'option d'exclure les documents de type css, javascript, ou image. Il peut aussi choisir l'option de compter seulement les hits d'un créneau horaire spécial (d'une heure). L'heure doit être indiqué en nombre, et être dans le format 24h. Ces deux options peuvent être combinées.
Aussi, l'autre grand aspect de l'application est qu'elle permet de générer un fichier au format GraphViz contenant la synthèse du fichier de log. Le nom du fichier à créer doit être donné par l'utilisateur. Si ce fichier existe déjà, pour un confort de l'utilisateur il aura le choix d'écraser le fichier existant ou choisir un nouveau nom de fichier. Dans le fichier créé, chaque document apparaîtra sous forme d'un noeud et chacun des arcs indiquera le nombre de hits associés. Comme avant, seul les hits ayant reçu le code "200" sont pris en compte. Cette option peut être combinée avec les deux précédentes pour ne garder que les pages qui intéressent l'utilisateur.
L'utilisateur peut écrire les options dans l'ordre qu'il veut, cela n'entraîne aucun changement.

- **Spécifications détaillées avec liens vers tests fonctionnels associés**

	Numéro test
si le nom du fichier à analyser n'est pas donné en argument au lancement de l'application, en dernière position, l'erreur de code 1 est détectée	1

si le fichier donné n'existe pas, l'erreur de code 3 est détectée	3
si le fichier n'a pas le bon format (.log), l'erreur de code 2 est détectée	4
si le fichier est protégé en lecture, l'erreur de code 3 est détectée	5
si le nom du fichier est bien donné, l'application fonctionne et le top 10 de toutes les pages est affiché	2
l'option entrée doit être valide (-e, -t ou -g), sinon l'erreur de code est 7 détectée	6
une option doit débuter par un '-', sinon l'erreur de code 4 est détectée	18
une option ne doit être écrite qu'une seule fois, sinon l'erreur de code 6 est détectée	19
une option ne doit faire qu'un caractère avec un '-',sinon l'erreur de code 4 est détectée	20
bon fonctionnement de l'option -e	7
s'il manque l'heure après le -t, l'erreur de code 5 est détectée	9
si l'heure est écrite en lettre, (par exemple on écrit "-h dix") l'erreur de code 5 est détectée	10
si l'heure n'est pas valide (t<0 ou t>=24), l'erreur de code 7 est détectée	11
bon fonctionnement de l'option -t	8
bon fonctionnement de l'option -e et -t	12
si le nom du fichier existe déjà, l'utilisateur à le choix d'écrire un nouveau nom de fichier ou d'écraser l'ancien fichier	14
si manque le nom du fichier , l'erreur de code 5 est détectée	15
si le nom de graphe est mauvais, (.dot manquant) l'erreur de code 5 est détectée	16
bon fonctionnement de l'option -g	13
bon fonctionnement des options -g, -t et -e	17

Tous les tests sont exécutés grâce au script shell. Tous les tests sont réussis.

Architecture globale de l'application

L'application comporte 3 grandes classes: analog, Log et CollectionDeLog.

La classe analog:

Contient la fonction main. Analyse les arguments entré lors du lancement de l'application. Détecte et envoie toutes les erreurs s'il y en a. Si les arguments sont correctes, appel le

constructeur de CollectionDeLog avec les bons paramètres, ceux ci dépendant des arguments du main.

La classe Log:

Modélise les lignes du .log en objet Log. Objet ayant comme attributs tous les paramètres d'une ligne du fichier .log. Dans notre application, tous les attributs ne sont pas utilisés comme l'adress ip ou le navigateur utilisé, mais pour une réutilisabilité du code, il est intéressant de les garder et pour cela nous utilisons une structure appelée Informations. Notre application utilise principalement le statut de la requête, la cible, le référent, le type du document et l'heure.

Les méthodes de cette classe sont :

Surcharge d'opérateurs

Cette classe contient une surcharge d'opérateur << et >> pour initialiser l'objet et pour l'afficher (affichage de tous ses attributs).

GetType et GetHeure

Ces deux méthodes get sont utilisées pour les attributs heure et type, qui sont ensuite utilisées pour réaliser les tris dans la classe CollectionDeLog.

SplitRefCib

C'est une méthode pour séparer les sites hébergeurs des pages des url de cible et de référent.

La classe CollectionDeLog:

Lit le fichier avec les log et le traduit en top dix et en graphe selon sa construction. Contient les structures de données pour le top 10 et le graphe.

Lorsqu'un objet est construit, il initialise ses attributs exclusion(bool) et heure(int) indiquant si la lecture du fichier log doit se faire sans prendre en compte les fichiers ayant une extensions de type image, css ou javascript et/ou en ne prenant en compte que les hits d'une certaine heure. Le constructeur appelle ensuite la méthode affichant ce que l'utilisateur attend, c'est à dire le top 10 ou le top 10 et le graphe.

Les méthodes de cette classe sont :

Constructeur et destructeur

C'est dans le constructeur que Les autres méthodes sont appelés selon les différents paramètres donnés. Ici le destructeur est inutile.

RemplirMapTopDix

Cette méthode remplit la structure de données contenant les éléments utiles pour afficher le top 10 .

RemplirMapGraph

Cette méthode remplit la structure de données contenant les éléments nécessaires à l'écriture du fichier GraphViz. A noter que pour reconnaître les différents noeuds du graphe nous utilisons une structure dessin (détaillée d'avantages par la suite).

AfficherTopDix

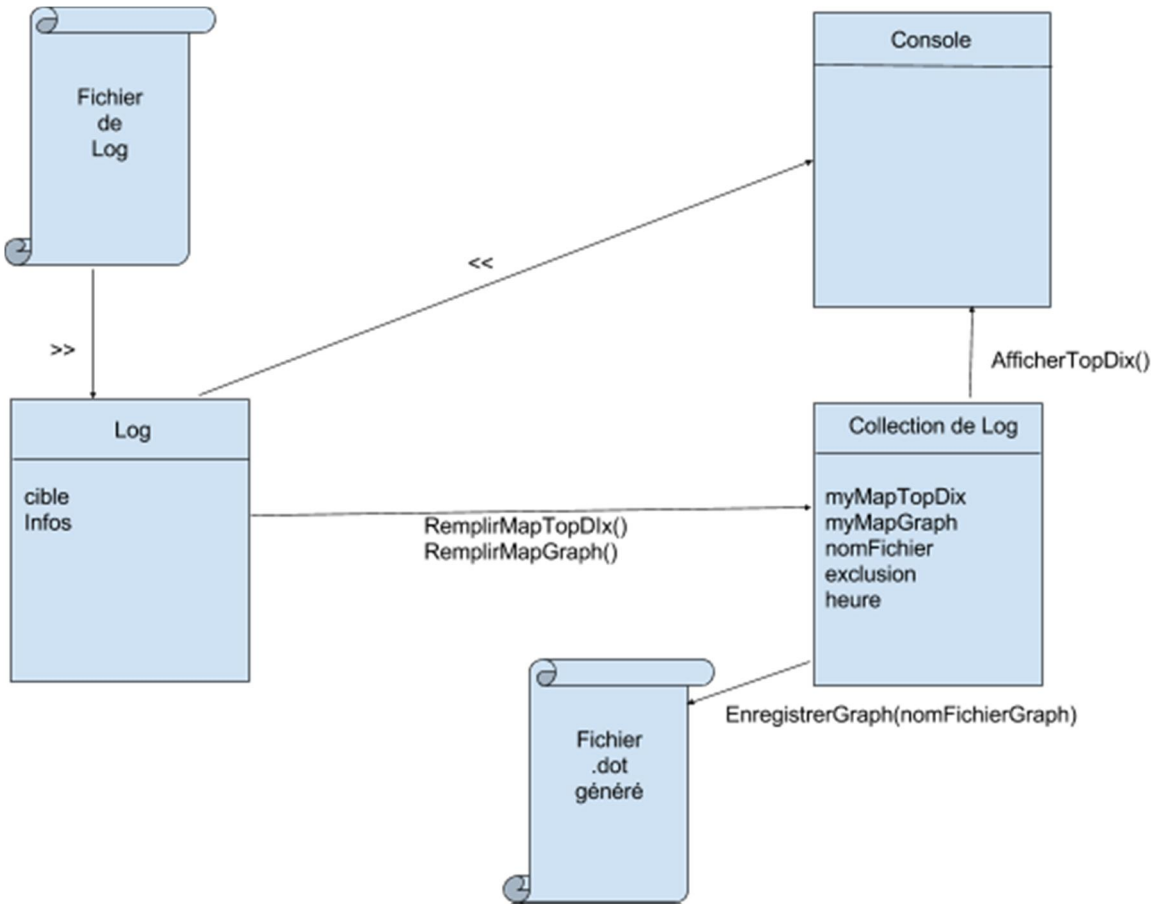
Cette méthode permet d'afficher le top 10. Elle commence par appeler la méthode qui remplit la structure de données nécessaire à l'affichage au top 10 puis affiche le top 10 dans la console.

EnregistrerGraph

Cette méthode permet l'écriture du fichier au format GraphViz. Elle commence par appeler la méthode remplissant la structure de données qui permet l'écriture de ce fichier. Ensuite, elle

écrit le fichier GraphViz dont le nom a été donné en ligne de commande auparavant et a servi d'argument de cette méthode.

Graphique d'interaction entre les classes



Présentation des structures de données

- **Top 10 - commentaire, justifications et schémas**

typedef map <string, int> mapTopDix;
mapTopDix myMapTopDix;
Structure contenant tous les logs et leur nombre de hit.
Lors de la lecture du fichier c'est cette map qui est remplie. Nous avons choisi une map avec comme clé un string correspondant au nom de la cible et comme valeur un int correspondant au nombre de hits de cette page. Cette structure de données facilite la création car nous souhaitons avoir toutes les cibles intéressantes (c'est à dire que le tri selon l'heure ou le type soit fait avant l'insertion dans la map) du fichier de log avec leur nombre de hits et la clé doit être unique. Pour trouver si une cible existe déjà dans la map, nous utilisons la méthode .find() de la structure map qui est de complexité o(log(n)). Cette opération étant effectuée à chaque ligne du document, nous avons ainsi préféré utilisé cette structure de donnée. On ne cherche pas à insérer des objets à des positions spéciales. L'insertion avec .insert() de la map nous convient.

Exemple:

/temps/4IF16.html	3905
/temps/4IF15.html	3102
/temps/4IF14.html	1568

typedef multimap <int, string> setTopDix;
setTopDix mySetTopDix;
Elle contient tous les logs triés selon leur nombre de hits.
Ayant la structure contenant toutes les cibles et leur nombre de hits, lors de l'affichage du top 10 on crée une multimap ayant cette fois en clé un int correspondant au nombre de hit, et en valeur un string correspondant à la cible. On utilise une multimap car la clé n'est pas forcément unique. Aussi la multimap étant triée par défaut sur sa clé (puis sur la valeur si la clé n'est pas unique), on affiche alors les 10 premiers éléments de cette multimap (ou moins si elle contient moins de 10 éléments).

Exemple:

3905	/temps/4IF16.html
3102	/temps/4IF15.html
1568	/temps/4IF14.html

- **Graph - commentaires, justifications et schémas**

typedef map <string,dessin> mapGraph;
mapGraph myMapGraph;
Structure contenant les cibles et leur référent avec le nombre de fois
Pour construire le graphe, il faut avoir une structure de données contenant la cible ainsi que son référent avec le nombre de fois qu'ils apparaissent. On choisit aussi une map, pour l'algorithme .find() et la clé doit être unique. La clé cette fois est un string contenant la cible et

le référent. La structure dessin contient un string correspondant à la cible, un string correspondant au référent, et un int correspondant au nombre d'apparition.

typedef map <string, int> mapNode;
mapNode myMapNode;
Elle contient le nom d'une page (cible ou referent) et le numéro du noeud associé.
Cette structure permet l'écriture du fichier au format GraphViz. C'est une map ayant pour clé un string contenant le nom d'un noeud et pour valeur le numéro du noeud auquel il est associé. Ainsi un noeud ne peut être écrit plusieurs fois. C'est pour cela que la structure map a été choisie, pour son algorithme .find() et la clé doit être unique. Il facilite aussi l'écriture des connexions entre les noeuds.

Conclusion

Notre application répond donc aux besoins donnés et traite la plupart des cas d'erreurs possibles. Elle inclut une certaine réutilisabilité en permettant la retranscription d'un objet log en ligne de log et possède une complexité basse grâce à l'utilisation de map et de multimap qui sont les structures de données les plus utiles dans le contexte de notre utilisation. En effet, nous allons plutôt stocker les logs sous forme de "dictionnaires" en associant une cible aux informations correspondantes pour la map. Et pour ce qui est de la multimap, c'est le même principe que la map mais ici une même clé (nombre de hits) peut-être associée à plusieurs informations (cible) ce qui facilite le calcul du top 10. Grâce à notre utilisation de structures (Informations et Dessin) et de map, il nous serait facile d'ajouter une à plusieurs options de tri à notre application et donc d'augmenter ses possibilités.

Nous aurions pu pousser la réutilisabilité encore plus loin en remplaçant la classe CollectionDeLog par une classe générique Collection<TypeT> qui prendrait donc dans notre cas des Log et qui contiendrait toutes les méthodes de CollectionDeLog mais en plus large.