Projet Offline DAAR Clone de egrep avec support partiel des ERE

réalisé par : mohammed-achraf CHARIF

2019/ 2020

Introduction

Le but du projet est de réaliser un programme en ligne de commande de recherche de chaînes de caractères, Le comportement habituel de egrep est de recevoir une expression rationnelle en argument, de lire les données sur l'entrée standard ou dans une liste de fichiers, et d'écrire les lignes qui contiennent des correspondances avec l'expression rationnelle sur la sortie standard.

Ses performances et sa facilité d'emploi en font un outil adapté pour rechercher efficacement des chaînes dans une arborescence complexe de fichiers. Il est par exemple utilisé pour trouver toutes les occurrences du nom d'une fonction dans le code source d'un programme ou de chaînes dans des fichiers de configuration.

Littérature

GNU grep est l'implémentation de la commande UNIX grep par le projet GNU lancé par Richard Stallman dans les années 1980. C'est un logiciel libre distribué selon les termes de la licence publique générale GNU.
La commande grep est initialement écrite pour UNIX par Ken Thompson. Dans le cadre du projet GNU, Mike Haerkal, implémentera grep en logiciel libre avant la fin des années 1980.
L'implémentation grep du projet GNU est aujourd'hui largement utilisée, par les systèmes Linux, FreeBSD, mais aussi Mac OS X.

En tant qu'acronyme, «grep» est un mot prononçable, en français comme en anglais. Les utilisateurs anglophones de grep utilisent souvent son nom comme un verbe, qui signifie «chercher quelque chose dans un fichier», comme on le ferait avec l'utilitaire grep. L'objet direct désigne alors les fichiers dans lesquels il faut chercher: «Kibo grepped his Usenet spool for his name.» (qui signifie: «Kibo a recherché son nom dans son spool Usenet.»). En français, le mot est plus rarement utilisé comme un verbe du premier groupe: «greppe les logs pour voir si le serveur a reçu ta demande.»
Par extension, le mot «grep» est aussi devenu un synonyme des expressions rationnelles elles-mêmes. De nombreux éditeurs ou traitements de texte proposent aujourd'hui des fonctionnalités de recherche à l'aide d'expressions rationnelles, qu'ils désignent souvent par «outil grep» ou «mode grep», dans lequel des motifs grep peuvent être saisis. Cette imprécision est source de confusion, en particulier dans les environnements non-UNIX.

Analyse et présentation théorique des algorithmes connus dans la littérature

Le Premier algorithme que j'ai utilisé est de transformer le motif en un arbrede syntaxe puis en un automate fini non-d éterministe avec-transitions selon la méthode Aho-Ullman puis en un automate fini d´eterministe avec la méthode des sous-ensemble.

J'ai utilisé aussi l'algorithme L'algorithme de Knuth-Morris-Pratt (ou d'une manière plus courte l'*algorithme KMP*) est un algorithme de recherche de sous-chaîne (de caractères), permettant de trouver les occurrences d'une chaîne dans un texte avec une complexité linéaire dans le pire cas. Sa particularité réside en un pré-traitement de la chaîne, qui fournit une information suffisante pour déterminer où continuer la recherche en cas de non-correspondance. Ainsi l'algorithme ne ré-examine pas les caractères qui ont été vus précédemment, et donc limite le nombre de comparaisons nécessaires.

Et un arbre radix ou arbre PATRICIA (pour *Practical Algorithm To Retrieve Information Coded In Alphanumeric* en anglais et signifiant algorithme commode pour extraire de l'information codée en alphanumérique) est une structure de données compacte permettant de représenter un ensemble de mots adaptée pour la recherche.

La méthode implémentée

On commence par la classe ‘Processor’ qui prend un string “Regex”, et qui détermine si la chaine est composée que des caractères alphabétiques pour éxécuter “l’arbre Radix” et “KMP” ou bien il se compose par les deux ‘|’ et ‘*’ pour utiliser la méthode des automates pour résoudre le problème.
Si la méthode est bien de celle des automates, j’ai utilisé l’approche de extraire l’arbre de syntaxe, l’insérer dans une chaine de caractère, après remplacer les caractères ‘(‘ et ‘)’ et ‘,’ par ‘ ‘ , je les empiles dans une pile à l’aide de mes classes ‘Block’ et ‘Entry’, après je fais un traitement pour ajouter les nouvelles états et les epsilons transitions dans la matrice NFA, la dérnière étape est bien transformer cette matrice NFA vers une autre DFA sans les epsilons transitions avec l’algorithme de sous-ensembles.

Conclusion

Pour la conclusion, on constate que la recherche avec KMP et RadixTree et très rapide en comparant de celle d'automates c'est évident, car ces deux dérniers sont très optimés, la méthode d'automates on trouve trop de traitement pour trouver l'arbre de syntaxe et le transformer en DFA et après en NFA et après faire la recherche d'une maniere exhaustive cela prend énormément du temps lors de la traitement, on conclut qu'il faut encore faire des otimisations a cette étape.