Projet 1 – projet *offline*. Clone de egrep avec support partiel des ERE.

BM Bui-Xuan

Expression régulière (RegEx) et la norme ERE : Une expression régulière est une représentation d'un langage algébrique reconnaissable par automate fini déterministe, dans le sens de la théorie de langage formel. Nous considérerons uniquement les expressions régulières répondant à la norme ERE décrite sur la page suivante, que nous notons également RegEx : `http://pubs.opengroup.org/onlinepubs/7908799/xbd/re.html`

Recherche de motif par RegEx : Etant donné un fichier textuel comprenant l lignes, le problème de recherche de motif contenu dans le fichier peut être décomposé en exécutant l fois la recherche du même motif, sur chacune des l lignes du fichier. Le but général de ce projet est d'implémenter une telle recherche de motif. Plusieurs stratégies peuvent être prises en compte. Pendant les séances de cours, nous proposons trois stratégies majeures, rappelées ci-après. Cependant, il est libre à chaque groupe d'implémenter ses propres stratégies, à condition que ce qui est décrit dans le rapport est en concordance avec ce qui est implémenté. Les trois stratégies de recherche de motif proposées dans les 3 premières séances de l'UE sont les suivantes.

— Si le motif n'est pas réduit à une suite de concaténations, nous allons d'abord transformer le motif en un arbre de syntaxe (1) puis en un automate fini non-déterministe avec ϵ-transitions selon la méthode Aho-Ullman (2) puis en un automate fini déterministe avec la méthode des sous-ensembles (3) puis en un automate équivalent avec un nombre minimum d'états (4) et enfin l'automate est utilisé pour tester si un suffixe d'une ligne du fichier textuel donné initialement est reconnaissable par cet automate (5). Il s'agit de la Séance 1 de l'UE.

— Si le motif est réduit à une suite de concaténations, il s'agit de la recherche d'un facteur dans une chaine de caractères : nous pouvons alors utiliser l'algorithme Knuth-Morris-Pratt (KMP) à la place de la machine de guerre décrite ci-dessus. Il s'agit de la Séance 2 de l'UE.

— La troisième stratégie est spécifique aux cas où le motif est non seulement réduit à une suite de concaténations, mais également composé exclusivement de caractères alphabétiques. Afin de cibler ces cas de figure, nous supposons que le fichier textuel est muni (par un précalcul) d'une table des indices représentant la liste de tous les mots (du langage humain) apparaissant dans le fichier textuel (*forward indexing*) ; ou qu'il est muni d'un *radix tree* contenant tous ces mots, selon la méthode des indices renversés (*inverted indexing*). Dans ce cas, la recherche du motif est effectuée en parcourant la table des indices (*forward index*) ou en parcourant le *radix tree* (*inverted index*). Il s'agit de la Séance 3 de l'UE.

Analyse de la performance de la recherche : Chaque groupe est libre de choisir ses stratégies de recherche de motif [1], à condition que la recherche de motif donne le même résultat que celui retourné par la commande `egrep` [2]. Si un motif est ciblé afin d'accélérer le calcul, comme le cas de KMP ou du *radix tree*, il convient de bien expliquer la méthode utilisée, ainsi que de faire confronter son implémentation avec la méthode de la Séance 1, voire confronter ses performances avec celles (finement optimisées) de la commande `egrep` elle-même.

ATTENTION : si on décide d'implémenter KMP (resp. *radix tree*), il faut tout de même l'expliquer dans le rapport, ainsi qu'inclure dans celui-ci les tests de performance de cette implémentation.

1. On peut notamment penser aux *hashtables* ou autres *Bloom-filters*.
2. En clair, il faut implémenter au moins ce qui a été vu en Séance 1.

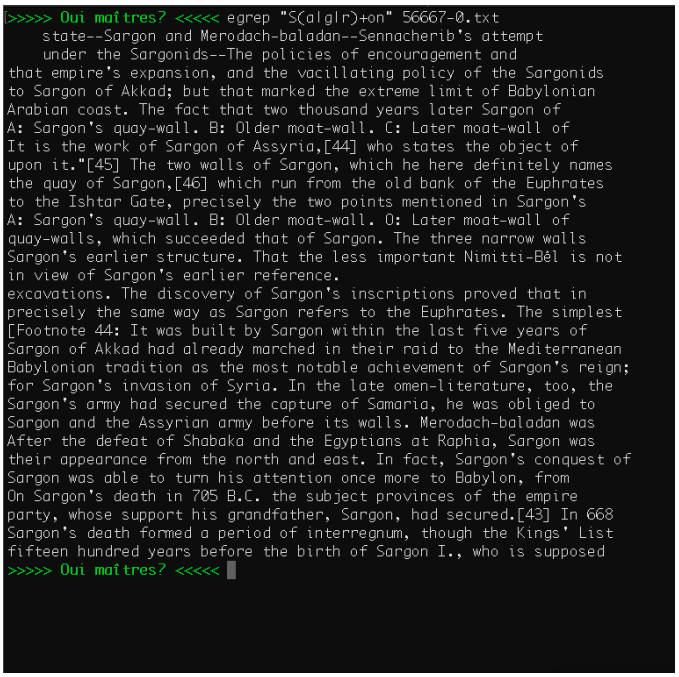

FIGURE 1 – Exemple de réponse de la commande Linux `egrep` prenant en entrée le RegEx ``S(a|g|r)+on'' et le fichier textuel `http://www.gutenberg.org/files/56667/56667-0.txt`.

1 L'énoncé du projet 1 – projet *offline*

Il s'agit de proposer un clone de la commande `egrep`. Cependant, certaines conventions de la norme ERE sont assez fastidieuses à venir complètement à bout. Nous allons nous restreindre aux éléments suivants : les parenthèses, l'alternative, la concaténation, l'opération étoile, le point (i.e. caractère universel), et la lettre ASCII. Un exemple de réponse de `egrep` est donné en Fig. 1.

Un effort particulier doit être mis sur la phase de test et celle de la rédaction du rapport. Il y aura ainsi deux notations distinctes pour le rendu de ce projet : une note pour la réalisation ($\approx 20\%$ de la note totale de l'UE) ; et une note pour le rapport ($\approx 10\%$ de la note totale de l'UE).

On veillera à expliciter les points suivants, pour chaque algorithme implémenté (y compris les algorithmes décrits pendant les séances de cours tels de ceux dans le livre Aho-Ullman, algorithme KMP, ou ceux de *radix tree*) :

— définition du problème et la structure de données utilisée.

— analyse et présentation théorique des algorithmes connus dans la littérature.

— argumentation concise appuyant toute appréciation, amélioration, ou critique à propos de ces algorithmes existants dans la littératures.

— partie test : méthode d'obtention des *testbeds*. Il est recommandé d'utiliser la base de Gutenberg disponible à l'adresse suivante : `http://www.gutenberg.org/`

— test de performance : mieux vaut privilégier les courbes, diagrammes bâton (moyenne + écart type) et diagrammes de fréquence, plutôt qu'exhiber les colonnes de chiffre sans fins...

— une discussion sur les résultats de test de performance est toujours la très bienvenue.

— conclusion et perspectives sur ce problème de recherche de motif dans un fichier textuel (a.k.a. le cas *offline* des moteurs de recherche).

Il est prudent d'avoir entre 5 et 10 pages pour un rapport avec un contenu moyen. La limitation formelle pour ce rapport est 12 pages. Il convient de bien respecter cette limitation : les pages 13+ ne seront pas lues !

Contraintes :

— A réaliser en binôme ou en individuel.

— Archiver la totalité du rendu en un seul fichier compressé contenant de la documentation (rapport $\approx$ 5-10 pages), un binaire et un README expliquant comment exécuter le binaire, le code source commenté, un Makefile (ou

Apache Ant si Java), un répertoire contenant les instances de test, et tout ce dont on juge utile à la lecture du projet sans toute fois dépasser la dizaine de Méga-octet.

— Envoyer ce fichier à `buixuan@lip6.fr`, 3 emails maximum par groupe. L'utilisation des hébergeurs en ligne (drive et compagnies) est proscrite. La nomination de préférence est daar-projet-offline-NOM.piki, où piki peut être un élément de $\{tgz, zip, rar, 7z, etc\}$.

— Deadline : 14 Octobre 2019, 13h45, cachet de serveur de messagerie faisant foi. Pénalité de retard : malus de $2^{h/24}$ points pour h heures de retard.