Comparaison de textes

Frédéric Magniez ("nom"@lri.fr)

Difficulté: à partir de *

1 Objectifs

Il s'agit d'écrire un programme `diffplus` qui prend en argument deux noms de fichiers et détermine une mesure de la ressemblance de ces deux fichiers. Les motivations de ce projet sont vastes et incluent la détection de plagiat ou encore la comparaison de séquences génomiques.

NOTE : Le projet proprement dit est classé en niveau de difficulté indicatif *. Toutefois, des prolongement conséquents sont décrits. Leur réalisation *complète* rend le projet plus difficile et il en sera naturellement tenu compte lors de l'évaluation.

2 Description

2.1 Distance d'édition

Etants données deux chaînes de caractères S (*source*) et T (*target*), les distances considérées sont définies par le nombre minimal d'opérations nécessaires pour passer de S à T.

Les opérations élémentaires autorisées pour la **distance d'édition** sont :
Insertion : insérer un caractère à une position quelconque de S.
Suppression : supprimer un caractère à une position quelconque de S.
Remplacement : remplacer un caractère à une position quelconque de S par un autre.

2.2 Algorithme utilisé

On utilisera l'algorithme dynamique présenté dans [WMMM90]. Il s'agit d'une variante de l'algorithme de Dijkstra sur un graphe dit d'édition dont on cherche la distance entre le sommet correspondant à S et celui correspondant à T. L'algorithme original ne prenant pas en compte les remplacements, on pourra dans un premier temps ne pas les considérer, puis adapter l'algorithme en le justifiant.

3 Prolongement

3.1 Distance de compression

Cette deuxième **distance de compression** autorise en plus des opérations sur des sous-chaînes :
Déplacement : déplacer une sous-chaîne de S à une position quelconque.
Copie : copier une sous-chaîne de S à une position quelconque.
Effacement : effacer une sous-chaîne.

Attention, si la distance d'édition est bien une distance et donc symétrique, ce n'est pas le cas de la distance de compression.

3.2 Algorithme utilisé

Pour la distance de compression, la situation est différente que celle de la distance d'édition puisque la calculer est un problème NP-difficile. On utilisera donc l'algorithme d'approximation de [EMS03, SS03]). Cet algorithme est relié à la technique de compression Lempel-Ziv-77 et fournit une 4-approximation de la distance de compression, *i.e.* un nombre $d(S,T)$ tel que $\mathsf{dist}_\mathsf{c}(S,T) \leq d(S,T) \leq 4 \times \mathsf{dist}_\mathsf{c}(S,T)$. La principale difficulté réside donc dans l'implémentation de cette variante de Lempel-Ziv-77. Une implémentation naïve donnera un algorithme quadratique alors qu'une plus raffinée [RPE81] utilisant l'algorithme de McCreight pour la construction d'abres suffixes sera linéaire.

4 Remarques de programmation

A chaque fichier sera associé une chaîne de caractères comprenant tout le fichier, y compris les sauts de ligne. La distance à calculer est entre ces deux chaînes de caractères et non ligne par ligne. Le choix de représentation et de stockage des données est libre. Il devra être approprié au calcul de la distance, et peut donc être différent selon la distance considérée.

Une série de tests bien choisis devra illustrer le comportement du programme (temps de calcul et résultats) et mettre en valeur les différences entre les différentes distances.

Enfin l'interface se fera via la ligne de commande avec un paramètre pour sélectionner la distance

```
> diffplus -s fic1 fic2
      Distance d'edition sans remplacement = 57
> diffplus -e fic1 fic2
      Distance d'edition avec remplacement = 51
> diffplus -c fic1 fic2
      Distance de compression approchee = 11
```

5 Page web

La page web `http://www.enseignement.polytechnique.fr/profs/informatique/Frederic.Magniez/03-04/INF_431/projet.html` contient les articles référencés. Si nécessaire, des indications complémentaires y seront ajoutées au fur et à mesure.

References

[EMS03] F. Ergün, S. Muthukrishnan, and S. Sahinalp. Comparing sequences with segment rearrangements. In *Proceedings of 23rd Foundations of Software Technology and Theoretical Computer Science*, pages 183–194, 2003.

[RPE81] M. Rodeh, V. Pratt, and S. Even. Linear algorithm for data compression via string matching. *Journal of the ACM*, 28(1):16–24, 1981.

[SS03] D. Shapira and J. Storer. Large edit distance with multiple block operations. In *Proceedings of 10th Symposium on String Processing and Information Retrieval*, pages 369–377, 2003.

[WMMM90] S. Wu, U. Manber, G. Myers, and W. Miller. An $O(NP)$ sequence comparison algorithm. *Information Processing Letters*, 35(6):317–323, 1990.