Méthodes Empiriques et Langages de Script
TP 8

G. A. Musillo
musillo4@etu.unige.ch

January 9, 2006

1 Exercice 1: alignements, tracés et distances entre deux mots et tracés

On peut représenter la similarité entre deux mots u et v au moyen d'un alignement. Si la longueur du mot u est égale à la longueur du mot v, aligner ces deux mots revient simplement à aligner chacun de leurs caractères. Les mots *perle* et *pearl* peuvent etre alignés de la facon suivante:

$$\langle \begin{array}{ccccc} p & e & r & l & e \\ p & e & a & r & l \end{array} \rangle$$

Cet alignement nous fait voir la distance qui sépare le mot *perle* du mot *pearl*. Pour cet alignement, cette distance vaut trois, car il suffit de subsituer aux trois derniers caractères du mot *perle* les caractères *a*, *r* et *l*. Remarquez que la distance est symétrique: il suffit d'appliquer trois substitutions au mot *pearl* pour obtenir le mot *perle*. Cette distance qui sépare deux mots de meme longueur est appelée *distance de Hamming*. Elle est définie comme le nombre de substitutions qu'il faut opérer sur un mot u pour obtenir un mot v (i.e. le nombre de caractères qui distingue u de v).

Comment aligner deux mots dont les longueurs respectives sont différentes ? Considérez l'alignement suivant:

$$\langle \begin{array}{cccccc} p & e & r & l & e & - \\ p & e & r & l & e & s \end{array} \rangle$$

Cet alignement montre qu'il suffit de suffixer le caractére s au mot *perle* pour obtenir le mot *perles*. Dualement, il suffit de supprimer le caractère final s du mot *perles* pour obtenir le mot *perle*. Voici un alignement plus complexe des mots *acga* et *atgcta*:

$$\langle \begin{array}{cccccc} a & c & g & - & - & a \\ a & t & g & c & t & a \end{array} \rangle$$

On peut montrer que cet alignement est optimal: il n'est pas possible de transformer *acga* en *atgcta* en appliquant moins de trois opérations. On appèle *distance de Levenshtein* entre deux mots u et v le plus petit nombre d'opérations nécessaires pour transformer la chaîne u en la chaîne v. Trois opérations sur les chaînes sont permises:

1. l'opération d'insertion d'un caractère de v dans u à une position donnée
2. l'opération de suppression d'un caractère de u à une position donnée
3. l'opération de substitution d'un caractère de u à une position donnée par un caractère de v

Par exemple, considérez les chaînes *abbc* et *babb*. Afin de transformer *aabc* en *babb*, on pourrait supprimer le *a* initial (on obtiendrait alors la chaîne *bbc*), puis insérer un *a* entre les deux *b*s (on obtiendrait alors *babc*) et enfin substituer au *c* final un *b*. Pour cet exemple, la distance vaut 3, puisque trois opérations ont été appliquées. Voici l'alignement qui la représente:

$$\left\langle \begin{array}{ccccc} a & b & - & b & c \\ - & b & a & b & b \end{array} \right\rangle$$

Pourtant, cette distance n'est pas la plus courte, car seules deux opérations (lesquelles ?) sont nécessaires pour transformer la chaîne *abbc* en la chaîne *babb*.
De nombreuses applications utilisent la distance de Levenshtein ou une variante de celle-ci. En voici quelques unes:

- correction orthographique: où l'on propose quelques mots distincts mais proches de ceux que l'utilisateur a entré
- bioinformatique: où l'on calcule la similarité entre deux séquences d'ADN ou deux séquences de protéines
- gestion de fichiers: où l'on recherche les différences qui distinguent deux fichiers (par exemple, la commande Unix *diff*)
- téléinformatique: où l'on ne transmet que les opérations qui permettent de transformer certaines données (par exemple, un serveur graphique qui génère les scènes d'un jeu ne transmet que ce qui a été modifié d'une scène à l'autre et non toutes les scènes)
- reconnaissance de la parole: où l'on devine les mots correspondants à certaines formes phonétiques

Nous verrons bientot l'algorithme qui permet de calculer la distance de Levensthein qui sépare deux mots.

Il existe une facon simple de mettre en évidence la similarité entre deux mots u et v de longueurs respectives m et n. On définit une matrice $m \times n$, appelée le tracé de u et v. Les composantes du tracé sont définies de la manière suivante: si $u_i = v_j$, alors $[i, j]$ vaut 1, sinon $[i, j]$ vaut 0.

On vous demande de programmer un module qui exporte deux procédures: une procédure qui calcule la distance de Hamming et une procédure qui calcule et affiche le tracé de deux mots quelconques. Résolvez également les problémes suivants:

- quel est l'ensemble de mots décrit par le tracé suivant ?

$$\begin{matrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{matrix}$$

- quel est l'ensemble de paires de mots décrit par le tracé suivant ?

$$\begin{matrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}$$

- calculez au moyen de votre programme le tracé des mots *acgat* et *atgcta*, parcourez ce tracé et déduisez-en au moins deux alignements dont l'alignement optimal. Décrivez votre démarche.