Algorithmique
TP 1 : Distance d'édition

1 Calcul de $d_{i,j}$

Soient deux chaînes s et s'. On note $d_{i,j}$ la distance d'édition entre le préfixe de taille i de la chaîne s et le préfixe de taille j de la chaîne s'.

Déjà, si $j = 0$, on a $d_{i,j} = i$ vu que c'est la distance d'un mot de taille i au mot vide. De même, si $i = 0$, on a $d_{i,j} = j$.

Il y a 3 cas à examiner pour déterminer $d_{i,j}$:

1. S'il faut effacer un caractère du préfixe de s de taille i, alors la distance est $d_{i-1,j} + 1$
2. S'il faut insérer un caractère, la distance est $d_{i-1,j} + 1$
3. S'il faut substituer un caractère, la distance est $d_{i-1,j-1} + c$ où c est le coût de substitution. Il vaut 0 si $s_{i-1} = s'_{j-1}$ et 1 sinon.

Finalement, la distance se définit comme :

$$d_{i,j} = min(d_{i-1,j} + 1, d_{i-1,j} + 1, d_{i-1,j-1} + c)$$

2 Algorithme itératif

On propose l'algorithme suivant :

On initialise un tableau d de taille $(m+1)(n+1)$ (où m et n sont les tailles des chaînes s et s'). La valeur de la ième ligne et de la jème colonne sera la distance $d_{i,j}$. On fait deux boucles imbriquées, sur les lignes puis sur les colonnes, en calculant itérativement $d_{i,j}$ avec la formule de la partie précédente. On renvoit $d_{m,n}$ à la fin de l'algorithme, qui vaut la distance d'édition entre s et s'.

3 Complexité de l'algorithme

Cet algorithme présente une complexité spatiale en $O(mn)$, à cause de la création du tableau d. Sa complexité temporelle est également $O(mn)$ à cause des deux boucles imbriquées.

4 Suite de modifications à effectuer

Pour retrouver la suite de modifications pour passer de s à s', on remonte le tableau d depuis $d_{m,n}$. On prend le voisin de gauche, diagonal gauche ou du haut qui rend la distance minimale. On crée ainsi un chemin dans la matrice d, et en fonction de la direction prise à chaque étape, on peut déterminer quelle action il faut effectuer. Si on choisit l'élément du dessus, cela correspond à un effacement, l'élément de gauche à une insertion et l'élément diagonal gauche à une substitution.

5 Distance de Damerau-Levenshtein

Pour prendre en compte la possibilité de permuter deux caractères adjacents, on rajoute une opération dans la boucle. Si on peut effectivement permuter les caractères, alors $d_{i,j} = d_{i-2,j-2} + c$.

6 Une version linéaire en espace

En fait, on ne se sert que de la ligne précédente pour calculer la nouvelle ligne de d. Il suffit donc de ne stocker que la ligne précédente et la ligne actuelle, qui représentent 2m données, donc une complexité spatiale de $O(m)$.