Distances d'édition : Distance de Levenshtein et Distance de Damerau-Levenshtein.

Charlotte Perlant

14 November 2018

1 Question 1

Soit $s = c_1...c_i...c_n$ et $s' = c'_1...c'_j...c'_m$ deux chaines. On désire calculer la distance de Levenstein entre les chaînes s et s' à l'aide de la programmation dynamique.
On souhaite transformer le mot s en le mot s' en un minimum d'opérations. A une étape donnée de la récursion, on suppose que les deux mots sont identiques au delà de la lettre de rang k (respectivement de rang k'). Cette hypothèse est en particulier vraie au début, car les deux mots vides sont identiques. On considère alors les préfixes de ces deux mots (les lettres précédant respectivement le rang k et le rang k'). Alors trois types d'opérations sont possibles à l'étape (k, k') : à l'étape précédente, on a soit supprimé une lettre, soit inséré une lettre, soit remplacé une lettre soit rien du tout si $c_k = c'_{k'}$.
Donc :

$$d_L(k, k') = min \begin{cases} d_L(k-1, k') \\ d_L(k, k'-1) \\ d_L(k-1, k'-1) + 1 & \text{si } c_k \neq c'_{k'} \\ d_L(k-1, k'-1) & \text{si } c_k = c'_{k'} \end{cases}$$

Avec $d_L(0, s_k) = d_L(s_k, 0) = k$.

L'algorithme se termine forcément car à chaque étapeon retire une lettre dans au moins un mot. On finit donc toujours par arriver à un cas trivial avec au moins un préfixe de longueur 0. En conclusion, on transforme donc un mot en l'autre en partant de la fin et en choisissant à chaque étape l'opération qui permet de diminuer au maximum le nombre d'étapes nécessaires.

2 Question 4

La fonction récursive naïve (c'est-à-dire sans mémoïsation) est de complexité temporelle exponentielle (en $O(3^n)$ car on fait le minimum de trois valeurs dans chaque boucle). En revanche, l'algorithme récursif avec mémoïsation (qui utilise

le tableau memo pour stocker les valeurs des distances calculées au fur et à mesure) est de complexité polynomiale (en O(mn) si m et n représentent les logueurs respectives des deux chaînes).

3 Question 5

La mémoïsation permet de ne pas répéter à chaque itération tous les calculs. En effet, avec la mémoïsation on stocke dans un tableau au fur et à mesure les valeurs des distances calculées. Ainsi, la fonction récursive avec mémoïsation est beaucoup plus rapide que celle naïve. La mémoïsation permet également de réduire le nombre de cases mémoire utilisé.

4 Question 7

La complexité en espace et en temps de la version itérative du calcul de la distance de Levenshtein est en O(mn) où m et n représentent respectivement les longueurs des chaînes de caractères s et s'. Ces complexités sont donc polynomiales.

5 Question 8

Globalement, les temps d'exécution affichés par le programme sont de 0. Ceci est sûrement dû au fait que les mots traités en exemple sont très courts et donc leur durée de traitement est particulièrement brève.

6 Question 11

Afin d'optimiser les complexités, il serait possible d'effectuer le calcul en ne gardant que la ligne actuelle et la ligne précédente en mémoire. Il est possible d'expliciter également les calculs effectués pour passer d'une chaîne à une autre et on obtient des suites.