TD3 : Algorithmes sur les textes

Exercice 1 : Chercher un mot

Soit une chaîne de caractères `t` de longueur n et un mot w de longueur `m<n`.

1. Ecrire un algorithme retournant la position de la première occurrence du mot w dans la chaîne `t` (ou -1 s'il est absent). Quelle est sa complexité?
2. Ecrire un un algorithme retournant la position de toutes les occurrences du mot w dans la chaîne `t` (ou une liste vide s'il est absent). Quelle est sa complexité?

Exercice 2 : Compter les mots

Écrire un algorithme qui compte le nombre de mots dans un texte.

Remarque : On considère comme caractère d'espacement tout caractère qui n'est pas alphanumérique (alphabétique accentué ou non et chiffres).

Exercice 3 : Palindrome

Écrire un algorithme récursif permettant de savoir si un tableau de caractères est un palindrome (un palindrome se lit “à l'endroit” et “à l'envers” de la même façon, comme par exemple “à l'étape, épate-la!”).

Remarque : on ne considère ni la ponctuation, ni les espaces, ni les accents.

Quelle est sa complexité?

Exercice 4 : Distances entre chaînes de caractères

On cherche à exprimer une distance entre deux chaînes de caractères.

Une distance entre 2 textes d1 et d2 est telle que :

- dist(d1,d2) = dist(d2,d1)
- dist(d1,d2) ≥ 0
- dist(d1,d2) + dist(d2,d3) ≥ dist(d1,d3)

Distance de Hamming

La distance de Hamming entre deux chaînes de même taille est définie comme le nombre de caractères non appariés. Ainsi la distance de Hamming entre “passoire” et “pastèque” est égale à 4. Peut-on généraliser cette distance à des chaînes de taille différente?

Distance d'édition

La distance d’édition est définie, pour deux chaînes de longueur quelconque, comme le nombre minimal d’opérations permettent de transformer d1 en d2, avec les opérations suivantes :

- **ins**(a) ⇢ insertion du caractère a
- **perm** (a, b) ⇢ remplacement de a par b
- **del** (a) ⇢ suppression du caractère a

auxquelles on ajoute l'identité **id**(a,a) de coût nul.

Il existe différentes manières de transformer la chaîne d1 en d2. On peut par exemple supprimer tous les caractères de d1 et insérer tous les caractères de d2, mais c'est rarement le nombre d'opérations optimal (|d1|+|d2|).

La résolution de ce problème repose sur les principes de la programmation dynamique.

1. Essayez de calculer “à la main” la distance d'édition
   - entre “robe”, “arbre”, et “porte”.
   - entre “cloche”, “hochet” et “louche”
   - Vérifiez sur ces exemples que l'inégalité triangulaire est bien respectée.
2. Résoudre au tableau l'algorithme de calcul pour “cloche” et “hochet”
3. Écrire l'algorithme général permettant de calculer la distance d'édition entre deux chaînes quelconques d1 et d2. Quelle est sa complexité?
4. Est-il possible de réduire cette complexité? Si oui, dans quels cas?