TD ALGO n°7
Algorithmes sur les textes

Exercice 1 : Chercher un mot

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

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

Exercice 2 : Algorithme de Boyer-Moore

Soit une chaîne de caractères `d` de longueur `n` et un mot `t` de longueur `m`<`n`. Le but est de retourner la position de toutes les occurrences du mot `t` dans la chaîne `d.` L'algorithme de Boyer-Moore examine d'abord la chaîne `t` et en déduit des informations permettant de ne pas comparer chaque caractère plus d'une fois.

Hypothèse : On suppose qu'il est possible de tester si un caractère `c` appartient au motif `t` en temps constant

Algo : Le but est de calculer un décalage permettant de ne pas inspecter les positions où il n'y a aucune chance de trouver le motif `t`.

- On commence par chercher la position `i = m - 1`
- Soit `c` = `d[i]` le dernier caractère
- Si `c` n'est pas dans `t`, le décalage vaut `m`
- Sinon on note `k` la position de la dernière occurrence de `c` dans `t`
    - si `k` vaut `m-1` (dernier caractère), le décalage vaut `m`
    - Sinon le décalage est égal à `m - 1 - k`

1) Exécuter l'algorithme (à la main) sur le problème suivant : `d="CODAGE DE CHAINES EN MACHINE"` et `t="CHINE"`
2) Ecrire l'algorithme.
3) Quelle est sa complexité (pire cas)?

* Exercice 3 : Algorithme de Knuth-Morris-Pratt (KMP)

Soit une chaîne de caractères `d` de longueur `n` et un mot `t` de longueur `m<n`. Le but est de retourner la position de toutes les occurrences du mot `t` dans la chaîne `d.` L'algorithme KMP commence par construire un tableau de préfixes qui permet d'éviter de recommencer la recherche depuis le début lorsque des correspondances partielles échouent. *

Le tableau des préfixes associe à chaque position du mot la longueur du plus long préfixe qui est également un suffixe valide, à l'exclusion du mot complet.

- **Préfixe** : Une sous-chaîne qui commence au début du mot.
- **Suffixe** : Une sous-chaîne qui se termine à la fin du mot.

1) Calculez le tableau des préfixes pour le mot **"ABABC"**

2) On souhaite comparer le mot **"ABABC"** et la chaîne **"ABABABCABABC"**. Effectuez la recherche *à la main* en comparant chaque caractère du motif avec les caractères correspondants de la chaîne, et en utilisant le tableau de préfixes pour décider où continuer la comparaison après une correspondance partielle ou un échec.

3) Le calcul du tableau des préfixes s'effectue en parcourant le motif tout en comparant les caractères pour déterminer les correspondances partielles entre les préfixes et les suffixes :

1. Initialisez le tableau avec `0` pour la première position.
2. Parcourez le motif avec deux indices :
    - `i` : Index actuel du caractère en cours d'analyse.
    - `j` : Longueur du préfixe qui correspond aussi à un suffixe.

4) Ecrivez maintenant l'algorithme KMP, en utilisant judicieusement le tableau de préfixes calculé dans la partie précédente. Quelle est sa complexité? Comparez avec l'algorithme Boyer-Moore.

Exercice 4 : Compter les mots

1. 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).

2. Dessiner l'automate fini correspondant.

Exercice 5 : Expressions régulières

Les langages réguliers sont des types de langages formels qui peuvent être reconnus par un automate fini. Le langage des expressions régulières est un langage régulier qui permet de décrire des motifs (c'est à dire des classes de mots) dans une chaîne de caractère.

1) On s'intéresse à l'expression régulière `(a|b)*c`
    a) Donner un ensemble de mots reconnus par ce langage
    b) Dessiner l'automate fini correspondant
    c) Décomposer les étapes correspondant à la reconnaissance d'un des mots proposés précédemment
    d) Soit `G` le graphe orienté décrivant cet automate, chaque arête étant indexée par un caractère. Donner l'algorithme qui indique si oui ou non l'expression est reconnue dans une chaîne `s` à partir de son automate fini.
    e) Quelle est sa complexité?

2) * On s'intéresse à l'expression régulière `(a|b)*ab`
    a) Donner un ensemble de mots reconnus par ce langage
    b) Dessiner l'automate fini correspondant
    c) Décomposer les étapes correspondant à la reconnaissance d'un des mots proposés précédemment.
    d) Soit `G` le graphe orienté décrivant cet automate, chaque arête étant indexée par un caractère. Donner l'algorithme qui indique si oui ou non l'expression est reconnue dans une chaîne `s` à partir de son automate fini.
    e) A quel parcours de graphe correspond cet algorithme? Quelle est sa complexité?

Exercice 6 : Syntaxe des expressions régulières

Le langage des expressions régulières est un langage régulier qui permet de décrire des motifs (c'est à dire des classes de mots) dans une chaîne de caractère.

1. Donnez l'expression régulière permettant de reconnaître les entiers relatifs et dessiner l'automate fini correspondant.

2. Donnez l'expression régulière permettant de reconnaître les nombres décimaux (par exemple `-3`, `12.3`, `-12.34`, `+3`, `0.`) et dessiner l'automate fini correspondant.

3. Donnez l'expression régulière expression régulière qui valide les noms de fichiers se terminant par l'une des extensions spécifiées : `.jpg`, `.png`, ou `.gif` et dessiner l'automate fini correspondant.

4. Donnez une expression régulière pour reconnaître les URL commençant par `https://...`

5. Écrivez une expression régulière pour reconnaître la date et l'heure au format `YYYY-MM-DD (HH:MM)`

Exercice 7 (facultatif) :

En vous inspirant du cours, dessinez l'automate fini qui effectue l'addition *décimale.*