Recherche de facteurs (séance 2). Exercices

Algorithme de Knuth, Morris, Pratt

1. Rappeler le principe de l'algorithme KMP.
2. Construire le tableau de pré-traitement de l'algorithme pour le facteur suivant : ABCDABD
3. Faire tourner l'algorithme KMP sur le mot-texte suivant :
   ABC ABCDAB ABCDABCDABDE
   en recherchant le facteur de la question précédente.
4. Donner une version de l'algorithme de Knuth-Morris et Pratt qui compte les occurrences d'un motif dans un texte.

Recherche avec Automate

1. On suppose donné un automate déterministe complet qui reconnaît le langage A*.{x}.A* (A étant l'alphabet) des mots contenant le motif x ainsi que les méthodes `Etat initial ()` (qui renvoie l'état initial de l'automate), `Boolean final(Etat s)` (qui dit si l'état s est un état final) et `Etat delta (Etat s, char c)` (qui implémente la fonction de transition de l'automate). En utilisant un tel automate :
   - Écrire un algorithme qui détecte la présence du motif x dans un mot donné.
   - Écrire un algorithme qui compte le nombre d'occurences de x dans un mot donné et qui affiche leurs positions.
   - Écrire un algorithme qui affiche tous les mots reconnus par l'automate dans un mot donné (On supposera que l'on dispose d'une méthode `affiche(String mot, int i, int j)` qui affiche la sous-chaîne d'un mot contenu entre les indices i et j).

   Quelle est la compléxité de ces algorithmes ?(On négligera le coût des affichages)
2. On dispose d'une chaînes de cractères contenant une suite de caractères A, C, G et T correspondant à un fragment d'ADN.
   ACGAGCATTACGATAGTAGATCGATTAGAGATTAAGCGCATAGAG
   On veut écrire un algorithme qui étant donné une telle chaîne :
   - détermine le nombre de codons GAG apparaissant dans ce fragment de génôme ;
   - réécrive le fragment dans une autre chaîne en remplaçant la seconde occurrence trouve du codon GAG par CAT.

   (a) Donner un automate qui permet de reconnaître le codon GAG.
   (b) Écrire la méthode `Etat initial (Etat s)` qui renvoie l'état initial de l'automate.
   (c) Écrire une méthode `Boolean final()` retournant vrai si l'automate est dans un état final et faux sinon.
   (d) Écrire une méthode `Etat delta (Etat s, char x)` qui implémente la fonction de transition de l'automate.
   (e) En utilisant les méthodes ci-dessus écrire une méthode `int reecrit-fragment(String fragment, String nouveau-fragment)` qui implémente l'algorithme voulu.