Feuille d'exercices n°10 - Révisions d'algorithmique du texte

Exercice 1 : Plus long sous-mot/facteur commun

Q. 1 Rappeler les définitions de **facteur** d'un mot et **sous-mot** d'un mot. Donner un exemple illustrant la différence entre les deux.

Q. 2 Définir le problème du plus long sous-mot commun.

Q. 3 Définir le problème du plus long facteur commun.

Q. 4 Proposer un algorithme de programmation dynamique résolvant le problème du plus long sous-mot commun. Dire d'abord comment obtenir la valeur optimale puis comment calculer une solution optimale.

Q. 5 Proposer un algorithme de programmation dynamique résolvant le problème du plus long facteur commun. Dire d'abord comment obtenir la valeur optimale puis comment calculer une solution optimale.

Q. 6 Quel lien peut-on faire entre ces deux algorithmes et l'algorithme de calcul d'alignement (aussi appelé calcul de la ditance d'édition ou distance de Levenstein).

Éléments pour établir un algorithme de programmation dynamique

→ définir une quantité paramétrée qui représente la valeur optimale du sous-problème défini par ces paramètres ;
→ justifier que cette famille de quantités est intéressante à calculer, en quoi cela permet de résoudre le problème initial ;
→ dire comment calculer l'une de ces quantités à partir des valeurs pour des paramètres plus petits ;
→ dire comment calculer les cas de base, *i.e.* les quantités pour les paramètres minimaux ;
→ si on opte pour de l'impératif, le pseudo-code qui explique dans quel ordre on va calculer ces valeurs, remplir le tableau.

Exercice 2 : Algorithme de Lempel-Ziv-Welsh (LZW)

Q. 1 Rappeler quel est le principe de la compression selon LZW. Que peut-on dire des longueurs des motifs ajoutés au dictionnaire au cours de la compression ? Que peut-on dire des motifs ajoutés au dictionnaire lors de la décompression par rapport à ceux ajoutés lors de la compression ?

Q. 2 Exécuter à la main la compression de LZW sur le mot gogogo. *On pourra prendre comme dictionnaire de base celui qui numérote les lettres de l'alphabet minsucule de 0 à 25.*

Q. 3 Exécuter à la main la décompression de LZW sur la chaîne obtenue à la question précédente, sans utiliser le dictionnaire construit précédemment.

On cherche maintenant à implémenter la compression et la décompression selon LZW en OCaml. On utilise pour implémenter les dictionnaires (et les ensembles si besoin) le module `Hashtbl`. On fournit dans le fichier `compagnon_lzw.ml` plusieurs fonctions utiles, notamment de manipulation élémentaires des dictionnaires mis en jeu lors de la compression ou la décompression.

Q. 4 Coder en Ocaml la compression et la décompression selon LZW. Tester ces procédures sur des textes, et calculer les taux de compression observés.

Q. 5 Citer un autre algorithme de compression (au programmme). En quoi ces deux méthodes de compression diffèrent-elles ?

Exercice 3 : Algorithme de Boyer-Moore (simplifié)

Q. 1 Quel problème résoud l'algorithme de Boyer-Moore ?

Q. 2 Rappeler le principe de l'algorithme de Boyer-Moore ?

Q. 3 Appliquer à la main l'algorithme de Boyer-Moore sur le texte bababobo et pour le motifbobo ?

Q. 4 Identifier quelles données relatives au motif seul peuvent être pré-calculées afin d'améliorer la complexité de l'algorithme.

Q. 5 Proposer le pseudo-code d'un algorithme calculant ces données. Quelle est la complexité de cet algorithme en fonction de la taille du motif ?

Q. 6 Proposer le pseudo-code de l'algorithme de Boyer-Moore. Quelle est la complexité de cet algorithme en fonction de la taille du motif et de celle du texte ?

Revoir le DS n°2 pour un algo de recherche des motifs à l'aide d'un automate pré-calculé pour le motif.