Recherche de facteurs (séance 1). Exercices

Algorithme linéaire

1. Écrire un algorithme qui vérifie que deux mots sont des anagrammes (mots de même longueur, constitués du même jeux de lettres ordonnées différemment). Par exemple, “marie” est un anagramme de “aimer”. Existe-t-il une solution au problème qui n'utilise pas de tableau suppléméntaire et qui ne modifie pas les mots donnés en entrée ?
2. Écrire un algorithme qui recherche un facteur dans un mot et qui retourne vrai s'il le trouve et faux sinon. Existe-t-il un moyen simple d'améliorer l'algo ?
3. Écrire un algorithme qui recherche un sous-mot dans un mot et qui retourne vrai s'il le trouve et faux sinon.
4. Écrire un algorithme qui affiche tous les “tokens” d'un texte. Un texte est un mot composé avec les lettres de l'alphabet plus les caractères d'espace et de ponctuation. Un “token” est un mot précédé et suivi par le caractère espace.
5. Écrire une fonction nommée `donnePositionFacteur` qui prend en argument deux tableaux de caractères (le mot et le facteur) et un entier (une position dans le mot à partir de laquelle chercher le facteur) et renvoie la position du premier facteur à partir de l'indice. La fonction renvoie `-1` si le facteur n'appartient pas au mot.
6. Écrire une fonction `affichePositionsFacteur` qui prend en argument un mot et un facteur et affiche les différentes positions du facteur dans le mot.

7. Écrire une fonction nommée `donneLongueurPrefSuff` qui prend en argument un mot et renvoie la longueur du plus long préfixe du mot qui est aussi un suffixe.

   Exemples :
   `abcx` $\rightarrow 0$
   `abcxa` $\rightarrow 1$
   `abcxabc` $\rightarrow 3$
8. Écrire un algorithme qui génère les préfixes d'un mot.
9. Écrire un algorithme qui génère les suffixes d'un mot.
10. Écrire un algorithme qui génère tous les facteurs d'un mot donné (sans se préoccuper de proposer plusieurs fois le même).
11. Écrire un algorithme qui génère tous les sous-mots d'un mot donné (sans se préoccuper de proposer plusieurs fois le même).