Devoir à la maison

Ce devoir est à rendre le mercredi 2 décembre en cours. Une attention particulière sera prêtée à la clareté du pseudocode et des explications. **Une justification succinte du fonctionnement de vos algorithmes et de leurs complexités est attendue.**

Polynômes

Un polynôme s'écrit $P = a_0 + a_1X + a_2X^2 \cdots + a_nX^n$. On veut implémenter un type abstrait pour les polynômes. Il s'agit d'implémenter des fonctions qui vont permettre, de sommer, de multiplier, de dériver, d'intégrer des polynômes pour pouvoir ensuite oublier l'implémentation concrète des polynômes et utiliser uniquement ces fonctions.

(a) Proposer un type pour un polynôme, sous forme de tableau de ses monômes (un monôme est un terme de la forme a_iX^i).

(b) Donner une fonction qui affiche un polynôme.

(c) Donner une fonction qui renvoie la somme de deux polynômes.

(d) Proposer maintenant un type pour un polynôme, sous forme de liste de ses monômes. Attention les monômes de coefficient 0 ne doivent pas être représentés.

(e) Pour cette nouvelle représentation, donner une fonction qui affiche un polynôme et une autre qui fait la somme de deux polynômes.

(f) Comparer les complexités de vos deux fonctions de somme.

(g) Donner une fonction qui réalise la dérivation d'un polynôme.

Puissance 4

Dans cet exercice on va travailler sur le jeu puissance 4. Le but du jeu est d'aligner 4 pions sur une grille comptant n lignes et m colonnes. Chaque joueur dispose de n*m pions d'une couleur (jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion descend alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

On utilise comme type de donnée pour représenter le jeu un tableau bidimensionnel d'entiers de n par m. Ce type est noté JEU. Si on a un JEU J, J[a,b] correspond à l'état de la case à la ligne numéro a et à la colonne numéro b. Sa valeur est -1 si la case est vide, 0 si c'est un pion du premier joueur (disons jaune) et 1 si c'est un pion du second joueur (rouge).

1. Donner une procédure avec pour entrée un JEU J, un entier c et un entier t. La procédure ajoute un pion de type t (qui vaut 0 ou 1) à la colonne c dans J. Si l'ajout est possible la fonction renvoie 1, 0 sinon. Quelle est la complexité de votre procédure ?
2. Donner une fonction avec comme entrée un JEU J et qui renvoie -1 si personne n'a gagné, 0 si c'est le joueur 0 et 1 si c'est le joueur 1. Quelle est sa complexité ?
3. Donner une structure de données qui permettrait de représenter une partie de puissance 4, c'est à dire une suite de coups.

4. On veut être capable de jouer une stratégie optimale au puissance 4. Pour cela il faut représenter tous les coups possibles. Proposer une structure de données arborescente qui permette de représenter l'ensemble des parties possibles. Il faut que les fils d'un élément de l'arbre soient les états du jeu accessibles en jouant un coup à partir de l'élément. À quoi correspondent les feuilles de cet arbre ?
5. Expliquer en quelques mots comment on peut créer l'arbre des coups et comment faire pour calculer pour chaque position si elle est gagnante pour le joueur jaune quoique fasse le joueur rouge.

Il n'y a pas de gène

On manipule des listes dont chaque élément contient un caractère, elles représentent donc des mots.

1. Étant donné en entrée les listes l_1 et l_2, donner un algorithme qui décide si la liste l_1 est contenue dans l_2. Par exemple *oba* est contenu dans *baobab* mais pas dans *oababa*. Quelle est la complexité de votre algorithme ?
2. Si la liste l_1 n'est pas contenue dans l_2, on voudrait savoir quel est le plus long préfixe de l_1 dans l_2. Par exemple le plus long préfixe de *superficiel* qu'on peut trouver dans *superfétatoire* est *superf*. Donner un algorithme pour résoudre ce problème et sa complexité.
3. La plus longue sous-séquence commune (PLSSC) à deux mots, est la suite de lettres consécutives la plus grande qu'on peut retrouver dans les deux mots. Par exemple *ananas* et *name* ont une PLSSC de taille 2 : *na*. Donner un algorithme qui étant donné deux listes l_1 et l_2 trouve la PLSSC à ces deux listes. Quelle est sa complexité ?
4. On peut faire un algorithme plus efficace en utilisant de la programmation dynamique. Soit deux listes l_1 et l_2 de taille n et m, on veut construire un tableau T de dimension $(n+1)*(m+1)$. Soit l_1^i le préfixe de taille i de l_1 et l_2^j le préfixe de taille j de l_2, $T[i,j]$ contient la taille de la PLSSC de l_1^i et l_2^j. Si on considère $T[2,3]$ pour les mots *ananas* et *name* c'est la taille de la PLSSC de *an* et de *nam*, c'est à dire 1. Quelle est la valeur de $T[0,i]$ et de $T[i,0]$ pour tout i ?
5. Remarquez et expliquez pourquoi la valeur de $T[i,j]$ est le maximum de la valeur de $T[i-1,j]$ et du plus long suffixe de l_1^i qui est dans l_2^j. Donner un algorithme pour calculer $T[i,j]$ étant donné $T[i-1,j]$, l_1 et l_2. Quelle est sa complexité ?
6. Donner un algorithme pour remplir le tableau T à partir des listes l_1 et l_2, en utilisant l'algorithme précédent. Où peut-on lire la taille de la PLSSC de l_1 et l_2 dans ce tableau ?