Université de Versailles Saint-Quentin
M1

Mise à niveau algorithmique
Année 2015-2016

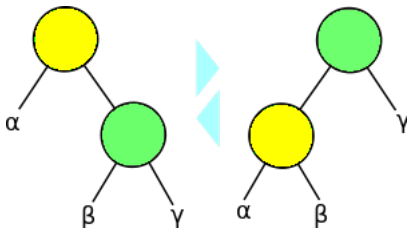
Examen

vendredi 23 octobre

Toute réponse donnée doit être **justifiée** et tout algorithme doit être **expliqué** au moins succinctement. Le barème est indicatif. L'examen dure deux heures et les notes prises en cours et td sont autorisées. Tout autre document, ordinateur, téléphone ... doit être rangé.

Exercice 1 L'arbre qui cache l'examen (5 points)

1. Donnez un algorithme qui prend en entrée un tableau T de n éléments, et un entier x et qui modifie chaque élément de T de façon à ce que $T[i] = T[i] + x + (x \mod i)$.
2. Appliquez votre algorithme au tableau $7, 2, 14, 9, 0, 4, 99, 17, 20, 102$ avec x le numéro inscrit sur votre chaise.
3. Construisez un arbre binaire de recherche en insérant successivement les éléments du tableau modifiés par vos soins. Vous représenterez chacune des étapes.
4. Supprimez les éléments correspondants à 99 puis 9.
5. La rotation est l'opération décrite dans l'image suivante. Les cercles gris clair et gris foncé sont des sommets et α, β et γ sont des sous-arbres.



Donner un algorithme qui prend un arbre en entrée et y applique une rotation, l'élément en gris clair étant la racine de l'arbre.

Exercice 2 Les automates c'est automatique (5 points)

On donne un automate non déterministe sur 4 états, et l'alphabet $\{a, b\}$ par les transitions suivantes :

	a	b
1	2,3	1
2	1	3
3	3,4	4
4	2	2

L'état initial est le 1 et l'état final le 4.

1. Représentez graphiquement cet automate.
2. Déterminisez l'automate.
3. Donner un algorithme qui prend en entrée un automate **déterministe** (la fonction de transition est donnée par un tableau à deux dimensions comme dans l'exemple) et un mot (donné par un tableau de lettres) et qui décide si l'automate accepte le mot ou non.

Exercice 3 Il n'y a pas de gène (10 points)

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. Dans le cas ou la première liste l_1 n'est pas contenue dans la deuxième 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 qui renvoie le plus long préfixe commun à deux mots. Quelle est la complexité de cet algorithme ?
3. On veut calculer la distance d'édition entre deux mots, distance qui sert pour la correction d'orthographe ou la comparaison des séquences d'ADN. La distance entre deux mots u et v est le nombre minimum d'opérations élémentaires qui permettent de transformer u en v. Les opérations sont :
   — la suppression d'un caractère
   — l'ajout d'un caractère
   Par exemple entre master et maître il y a une distance de 4 : master $\rightarrow$ mater $\rightarrow$ matr $\rightarrow$ maîtr $\rightarrow$ maître.

   On note u_i le préfixe de taille i de u, par exemple $(master)_3 = mas$. Étant donné deux mots u et v de taille n et m, on veut créer le tableau à deux dimensions $d[n][m]$. Dans la case $d[i][j]$ du tableau on veut la distance d'édition entre u_i et v_j. Donner la valeur de $d[i][0]$ et de $d[0][i]$ pour tout i.
4. Donner la valeur de $d[i][j]$ en fonction de $d[i-1][j]$, $d[i][j-1]$ et $d[i-1][j-1]$. Attention cette relation dépend des deux mots u et v.
5. Donner un algorithme qui étant donné u et v remplit le tableau d et renvoie la distance d'édition.
6. Quelle est la complexité de votre algorithme ?