IFT 2125 Introduction à l'algorithmique

Professeure: Sylvie Hamel

FINAL, Mardi 21 avril 2015

NOM: ________________________________

MATRICULE: ______________________

QUESTION 1:		10 pts
QUESTION 2:		10 pts
QUESTION 3:		15 pts
QUESTION 4:		25 pts
QUESTION 5:		15 pts
QUESTION 6:		10 pts
QUESTION 7:		15 pts
QUESTION 8 - BONUS:		10 pts
TOTAL		/100

Directives

1) Répondez directement sur le questionnaire
2) Examen à livre fermé i.e. aucune documentation permise.
(Cela comprend tout appareil électronique)
3) Vérifiez que ce document comporte bien 14 pages.
4) Utilisez les versos des pages comme brouillons.
5) Les meilleures réponses n'utilisent pas tout l'espace alloué.
6) Une réponse sans justification ou preuve ne vaut rien, sauf indication contraire.
Certaines conventions et définitions se trouvent à la page 14.

Question 1) [10 points]
Démontrez formellement, en utilisant la définition de $\mathcal{O}$, que $\mathcal{O}(f(n)) = \mathcal{O}(g(n)) \Leftrightarrow f(n) \in \mathcal{O}(g(n))$ et $g(n) \in \mathcal{O}(f(n))$.

Question 2) [10 points]

(**10 points**) Pour chacun des algorithmes suivants, trouvez ce qu'il calcule, faites une analyse du temps d'exécution et exprimez cette complexité en utilisant la notation $\mathcal{O}$.

a) Algorithme Un($a > 0$, $n \geq 0$ deux entiers)
   – retourne ?
      $k \leftarrow 0$
      $b \leftarrow 1$
      TANT QUE $k < n$ FAIRE
         $k \leftarrow k + 1$
         $b \leftarrow b * a$
      FIN TANT QUE
      RETOURNER b

b) Algorithme Deux(a: vecteur de taille n)

– retourne ?.

$p \leftarrow 0$
$i \leftarrow 0$
$k \leftarrow 0$
TANT QUE $k < n$ FAIRE
SI $\lfloor \frac{a[k]}{2} \rfloor = \frac{a[k]}{2}$
ALORS $p \leftarrow p + 1$
SINON
$i \leftarrow i + 1$
FIN SI
$k \leftarrow k + 1$
FIN TANT QUE
RETOURNER(p, i)

Question 3) [15 points]
Soit $G = (V, E)$ un graphe orienté connexe ou non. Donnez le pseudocode d'un algorithme qui, étant donné un sommet s, retourne tous les sommets accessibles à partir de s. Donnez et expliquez la complexité en temps de votre algorithme. (Si vous utilisez un algorithme vu en classe, vous devez quand même décrire formellement le pseudocode de cet algorithme).

Question 4) [25 points]
Soit le problème du retour de la monnaie suivant: On veut faire la monnaie exacte pour un montant n et on a à notre disposition une infinité de pièces de 1¢ , 3¢ , 5¢ et 6¢.

a) Donnez une stratégie vorace pour ce problème. Quelle serait une solution de cette stratégie pour le montant $n = 10¢$?

b) Donnez un algorithme de programmation dynamique pour résoudre ce problème (décrire les conditions initiales, les cas non définis et la récurrence). Démontrez la complexité de cet algorithme.

c) Calculez la table de programmation dynamique de votre algorithme en b) pour le montant $n = 10$¢:

	0	1	2	3	4	5	6	7	8	9	10
$v_1 = 1$											
$v_2 = 3$											
$v_3 = 5$											
$v_4 = 6$											

d) À partir de la table calculée en c), donnez l'ensemble des pièces représentant une solution optimale. Indiquez aussi dans la table les pointeurs menant à cette solution.

e) Est-ce que cette solution est la même que celle trouvée en a)? Pourquoi?

Question 5) [15 points]
La table de programmation dynamique donnée ci-bas a été construite pour un problème d'alignements entre les deux séquences $AGACATTAGA$ et $ATGA$. Elle utilise la distance d'édition, i.e. une pénalité de suppression, d'insertion et de substitution de 1.

	-	A	G	A	C	A	T	T	A	G	A
-				0							
A		0	1								
T											
G											
A	4										

a) Quel type d'alignement cherchons-nous à faire ici? Comment le savez-vous ?

b) Complétez la table de programmation dynamique et donnez un alignement optimal ainsi que le score de cet alignement. Indiquez aussi dans la table les pointeurs menant à cette solution.

c) Nommez un autre type (que celui en a)) d'alignements possible entre les deux séquences. Quels sont les conditions initiales et la récurrence pour ce type d'alignement si la distance utilisée est la distance d'édition?

Question 6) [10 points]
Soit $G = (V, E)$ un graphe non orienté, connexe, où chaque arête à un poids non négatif. Voici une stratégie diviser pour régner pour le problème de trouver un arbre couvrant minimal pour ce graphe:
On partitionne l'ensemble des sommets V en deux ensembles V_1 et V_2 tels que leur cardinalité diffère d'au plus 1. Soit E_1 l'ensemble des arêtes qui ne sont incidentes qu'à des sommets de V_1 et soit E_2 l'ensemble des arêtes qui ne sont incidentes qu'à des sommets de V_2. On résout alors récursivement le problème de recherche d'un arbre couvrant minimal pour chacun des sous-graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$. Finalement, on sélectionne l'arête de poids minimal de E entre un sommet de V_1 et un sommet de V_2 et l'on utilise cette arête pour obtenir l'arbre couvrant minimal pour G.
Prouvez que cet algorithme calcule bien un arbre couvrant minimal pour G ou donnez un contre-exemple.

Question 7) [15 points] Donnez le pseudocode d'un algorithme diviser pour régner qui trouve le plus petit entier y, dans une liste triée $L[1..n]$, telle que $y > x$, pour un entier donné $x < L[n]$. Donnez et expliquez la complexité en temps de votre algorithme. (Si vous utilisez un algorithme vu en classe, vous devez quand même décrire formellement le pseudocode de cet algorithme).

Question 8) Bonus [10 points]
Étant donné un arbre T et un noeud v de T, on appelle "excentricité de v", la longueur du plus long chemin entre v et l'un des autres noeuds de T. Un noeud de T d'excentricité minimum est appelé le centre de T.

a) Donner un algorithme efficace pour trouver le centre d'un arbre T.

b) Est-ce que le centre est unique?

Aide mémoire

Propriétés des $\log$**:**

- $\log_b x = y \Longleftrightarrow b^y = x$
- $\log_a(xy) = \log_a x + \log_a y$
- $\log_a(\frac{x}{y}) = \log_a x - \log_a y$
- $\log_a x^y = y \log_a x$
- $\log_a x = \frac{\log_b x}{\log_b a}$
- $x^{\log_b y} = y^{log_b x}$

Séries, sommations:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=0}^{n} r^i = \frac{(r^{n+1}-1)}{(r-1)}, \text{ pour } r \neq 1$$

Définition de limite: Pour $g : \mathbb{N} \longrightarrow \mathbb{R}^{\geq 0}$, on dit que

$$\lim_{n\to\infty} g(n) = \ell \quad \Longleftrightarrow \quad \forall \epsilon > 0, \exists\, n_0 \in \mathbb{N} \text{ tel que } |g(n) - \ell| \leq \epsilon, \forall n \geq n_0$$

Règle de l'Hôpital: Supposez que $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = 0$ ou que $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = +\infty$ alors

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

Dérivées: Dans les formules suivantes, u et v désignent des fonctions de n; a, c et k sont des constantes et u' désigne la dérivée de u par rapport à n:

- $c' = 0$
- $(cn)' = c$
- $(cn^k)' = kcn^{k-1}$
- $(cu)' = cu'$
- $(uv)' = uv' + vu'$
- $(\frac{u}{v})' = \frac{vu'-uv'}{v^2}$
- $(u^k)' = ku^{k-1}u'$
- $(\ln u)' = \frac{u'}{u}$
- $(a^u)' = a^u u' \ln a$

Théorème: Soient $n_0 \geq 1$, $\ell \geq 1$, $b \geq 2$ et $k \leq 0$ des entiers et soit $c \in \mathbb{R}^+$. Soit $T : \mathbb{N} \longrightarrow \mathbb{R}^+$ une fonction éventuellement non décroissante telle que

$$T(n) = \ell T(\frac{n}{b}) + cn^k, \forall n > n_0,$$

où $\frac{n}{n_0}$ est une puissance de b. Alors,

$$T(n) \quad \in \quad \begin{cases} \Theta(n^k) & \text{si } \ell < b^k \\ \Theta(n^k \log_b n) & \text{si } \ell = b^k \\ \Theta(n^{\log_b \ell}) & \text{si } \ell > b^k \end{cases}$$