L2 Informatique - Algorithmique - Session 1

mardi 17 décembre 2013. 14h00-16h00. T' 301.

La précision et la clarté de votre rédaction sont *fondamentales*. Documents interdits. Durée 2h00. Le barème indiqué est *approximatif*.

Exercice 1. [2pts] Quelle est la valeur de la somme des n premiers entiers naturels non-nuls ? Plus généralement, soit (u_n) une suite arithmétique de raison r et de premier terme u_0. Démontrez que

$$\sum_{i=0}^{n} u_i = (n+1)\left(u_0 + \frac{rn}{2}\right).$$

Exercice 2. [3pts] Rappelez la définition d'un APO et celle d'un TAS. Montrez que le tableau $T = [3, 2, 5, 1, 2, 7]$ n'est pas un TAS. Justifiez votre réponse en indiquant les *indices* des nœuds pères qui violent la propriété APO et dessinez l'arbre binaire associé à T. Appliquez l'algorithme étudié en cours/TD/TP qui transforme ce tableau en TAS, dessinez l'arbre binaire associé *à chaque étape* de l'algorithme.

Exercice 3. [2.5pts] Écrivez l'expression arithmétique suivante sous forme postfixée :

$$32 - 2 \times \big(12 - (7-2) \times 3\big).$$

Évaluez cette expression à l'aide d'une pile. On rappelle qu'on lit les termes de l'expression postfixée et que, selon qu'il s'agisse d'un opérande x ou d'un opérateur $\star$, on empile x ou on dépile les deux valeurs a et b au sommet de la pile et on empile $a \star b$.

Faites un tableau contenant les différents états de la pile au fur et à mesure de l'évaluation de l'expression postfixée que vous aurez trouvée. Quel est le résultat final dans la pile ?

Exercice 4. [2.5pts] Dressez la table des longueurs des plus longues sous-séquences communes entre les mots `slaloms` et `escalopes`.

Exercice 5. [4pts] Écrivez un algorithme `EstSousSeq(u,v):booleen` qui renvoie `vrai` si u est une sous-séquence de v et `faux` sinon. Estimez la complexité de cet algorithme dans le meilleur des cas et dans le pire des cas en fonction des longueurs des séquences u et v.

Exercice 6. [6pts] Écrivez un algorithme `Dichotomie(L,x):entier` qui recherche la valeur x dans une liste L supposée triée dans l'ordre croissant avec la méthode de la dichotomie. On supposera que la liste est indexée à partir de 1 et que $\sharp L$ désigne le nombre de termes de la liste. L'algorithme devra renvoyer le premier indice de x où x apparaît dans la liste L ou 0 si x n'apparaît pas dans la liste L.

Estimez la complexité de votre algorithme dans le meilleur des cas et dans le pire des cas en fonction du nombre n de termes dans la liste L.

Exercice 7. [2pts] Écrivez un algorithme `MaxMax(L):entier` qui renvoie les indices des deux termes les plus grands de la liste L indexée à partir de 1. Par exemple pour la liste $[2, 1, 4, 3, 7, 2, 1, 9, 3]$, l'algorithme renvoie les valeurs 5 et 8. On veut que l'algorithme trouve ces deux indices *en une seule passe* sur la liste. Quelle est la complexité de cet algorithme en fonction du nombre n de termes dans la liste L ?