L3 Informatique. Session 1 - Algorithmique

mardi 18 décembre 2012. 10h00-12h00. T' 301.

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

Exercice 1. [2pts] Le tableau $T = [3, 10, 9, 8, 4, 2, 1, 4, 1]$ est-il un tas ? Justifiez votre réponse. Si oui, échangez le premier terme avec le dernier terme puis appliquez l'algorithme descendre(T, n, k) du tri par tas qui rétablit la propriété APO du tableau, si non appliquez directement l'algorithme. Dessinez le parcours dans l'arbre de la valeur à la racine.

Exercice 2. [1pt] Écrivez l'expression arithmétique suivante sous forme postfixée :

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

Exercice 3. [2pts] Dressez la table des longueurs des plus longues sous-séquences communes entre les mots `slaloms` et `escalopes`.

Exercice 4. [10pts] On se propose d'écrire un algorithme de tri plus performant que le tri insertion, le tri *shell* qui en est une variante. Vous trouverez l'algorithme du tri par insertion en figure 1.

Si T désigne un tableau de n valeurs à trier, le tri Shell consiste à se fixer un entier $k < n$ et appliquer dans une première passe l'algorithme du tri par insertion sur chacun des k sous-tableaux $(T_i)_{i\in[1:k]}$ constitués des termes d'indices $i + \lambda.k,\ \lambda \in \mathbf{N}$ du tableau T. Par exemple pour un tableau de 12 éléments avec $k = 5$, le tableau T_1 contient les termes d'indices $1, 6, 11$, T_2 ceux d'indices $2, 7, 12$, T_3 ceux d'indices $3, 8$, etc.

Une fois le tri insertion réalisé sur tous les tableaux T_i, on décrémente le pas k et on recommence le processus jusqu'à $k = 1$ (ce qui revient à faire le tri insertion classique sur la dernière passe).

1. [2pt] Appliquez l'algorithme du tri-shell sur le tableau $T = [7, 3, 1, 2, 5, 6, 4]$ avec un pas initial $k = 3$. Donnez le contenu du tableau T après chaque tri insertion sur chaque sous-tableau T_i et ceci pour toutes les valeurs de k de 3 à 1.
2. [4pts] Ecrivez un algorithme tri-pas(T,n,k) sur la base de l'algorithme du tri par insertion qui fait le tri par insertion sur chacun des T_i avec un pas de k.
3. [1pt] Combien y-a-t-il de comparaisons entre la carte à insérer et la valeur courante $T[i]$ (ligne 5 de l'agorithme de tri par insertion ci-dessus) effectuées dans le pire des cas par l'algorithme tri-pas(T,n,k) en fonction de n et de k ?
4. [1pt] Ecrivez un algorithme tri-shell(T,n,k) qui s'appuie sur l'algorithme tri-pas pour réaliser le tri shell.

```
    Algorithme Tri-Insertion(T, n)
    données
        T : tableau de n valeurs;
    variables
        i, j, carte : entiers;
1   j ← 2;
2   tantque (j ≤ n) faire
3       carte ← T[j];
4       i ← j − 1;
5       tantque (carte < T[i] et i > 0) faire
6           T[i + 1] ← T[i];
7           i ← i − 1;
8       ftq
9       T[i + 1] ← carte;
10      j ← j + 1;
11  ftq
```

FIGURE 1. Le tri par insertion.

5. [2pts] Combien y-a-t-il de comparaisons globalement effectuées par le tri-shell en fonction de n et de k ?

Exercice 5. [5pts] Écrivez un algorithme val2(x) qui détermine la valuation dyadique $\nu_2(x)$ d'un entier x, c'est-à-dire qui détermine l'indice du bit le plus à droite dans l'écriture binaire $x_n x_{n-1} \dots x_1 x_0$ de x en indexant les bits dans l'ordre croissant des puissances de 2 :

$$x = \sum_{i=0}^{n} x_i 2^i$$

Exemple, la valuation dyadique de 84 est 2 car $84 = 1010\underline{1}00$. Vous utiliserez librement les fonctions DG(x, k), DD(x, k) qui renvoient respectivement l'entier dont l'écriture binaire est celle de x décalée à gauche (resp. à droite) de k bits et les opérateurs binaires bit-à-bit ET, OU, NON, XOU.

Implantez cet algorithme en langage C sous forme de fonction `uchar val2(ullong x)` où `uchar` est le type `unsigned char` et `ullong` le type `unsigned long long`. On rappelle l'équivalent des opérateurs bit-à-bit en C : DG (`<<`), DD (`>>`), ET (`&`), OU (`|`), NON (`~`) et XOU (`^`).