Retem

جامعة هواري بومدين للعلوم و التكنولوجيا
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
Département d'Informatique

Concours d'accès au Doctorat 3ième Cycle Informatique 2016 – 2017

Le 26/10/2016

Matière 2 : Algorithmique Avancée et Complexité
Coefficient **1**, durée **1 Heure.**
(Spécialité : RSI)

Exercice 1 : (12 points)

Le but de cet exercice est de donner un algorithme rapide pour compter le nombre d'inversions présentes dans une liste sans doublons. Pour fixer les idées, on travaille sur des permutations des n ($n > 1$) premiers entiers positifs, donc sur l'intervalle 1..n. Une liste de valeurs sans doublons est rangée dans le tableau T[1..n], et l'on dit que les nombres i et j, tels que $1 \leq i < j \leq n$, forment une inversion si $T[i] > T[j]$.

Par exemple, pour n = 8, le nombre d'inversions de la liste suivante vaut 13 :

i	1	2	3	4	5	6	7	8
T[i]	3	5	2	8	6	4	1	7

Écrites en tant que liste des couples d'indices (i, j) tels que i<j et T[i] >T[j], les inversions sont les suivantes :

(1, 3), (1, 7), (2, 3), (2, 6), (2, 7), (3, 7), (4, 5), (4, 6), (4, 7), (4, 8), (5, 6), (5, 7), (6, 7)

1. Ecrire un algorithme itératif naïf qui calcule le nombre d'inversions dans une liste sans doublons. Donner la complexité de cet algorithme.

On va supposer maintenant que $n = 2^k$, pour k entier supérieur ou égal à 1.

Pour k>1, on peut partitionner les inversions en trois catégories :

– celles dont les deux termes sont dans la première moitié de T, par exemple : (2, 3) dans l'exemple ci-dessus,

– celles dont les deux termes sont dans la seconde moitié de T, par exemple : (6, 7),

– celles qui ont le premier terme dans la première moitié de T et le second dans la seconde moitié, par exemple : (2, 6).

2. Ecrire la fonction *NbInversion2(bi, bs) :entier ;* telle que l'appel *NbInversion2(1,n)* calcule, selon une approche Diviser pour Régner, le nombre d'inversions présentes dans le tableau T[1..n]. Donner sa complexité. Que peut-on en conclure par rapport à la première question ?
3. Pour améliorer l'efficacité de la solution, il faut trouver un moyen de ne pas comparer tous les couples possibles. Ecrire un algorithme qui non seulement donne le nombre d'inversions, mais également effectue le tri du tableau T. Donner sa complexité.

Exercice 2 : (8 points)

1) Ecrire un algorithme en **Θ(n*log n)** qui étant donné un ensemble **S** de **n** nombres réels et un nombre réel **x** donné, détermine s'il existe ou non dans **S** deux éléments dont la somme est égale à **x**. Justifiez votre réponse.

2) Démontrer que $\mathbf{T(n) = (n-3)*2^n + 4}$ est solution de l'équation suivante :

$$T(n) = \begin{cases} 1 & si\ n = 0 \\ 0 & si\ n = 1\ ou\ n = 2 \\ 5 * T(n-1) - 8 * T(n-2) + 4 * T(n-3) & sinon \end{cases}$$

3) La fusion de deux listes triées en une liste triée de **m** et **n** éléments respectivement en une liste triée fait **(n+m-1)** comparaisons dans le cas pire.
Qu'en est-il de la fusion de deux arbres binaires de recherche ayant respectivement **m** et **n** nœuds ? Justifiez votre réponse.