UNIVERSITÉ DE MONTPELLIER

UE : HLIN301

Session : 1 — Durée de l'épreuve : 2 heures

Date : Janvier 2019 — Documents autorisés : 1 feuille A4 recto verso

Licence $2^{ème}$ année, parcours Informatique et Math-Informatique — Matériel utilisé : aucun

1 Tableaux

On cherche à vérifier si un tableau d'entiers T contient au moins un entier e qui est égal à la somme des éléments de T inférieurs strictement à e.

On supposera que tous les éléments du tableau ont des valeurs différentes.

Par exemple le tableau | 22 | 26 | 2 | 30 | 4 | 7 | 1 | 8 | contient une telle valeur ; en effet 7 est égal à la somme des éléments du tableau de valeur inférieure strictement à 7 : 1, 2 et 4. C'est également le cas de 22 qui est égal à la somme de 1, 2, 4, 7 et 8.

Question 1. (2,5 points) Écrivez un algorithme pour ce problème dans le cas où le tableau n'est pas trié. Votre algorithme ne doit pas trier le tableau. Quelle est la complexité de votre algorithme en fonction de la taille n du tableau ?

Question 2. (2,5 points) Écrivez un second algorithme pour ce problème en considérant cette fois-ci que la donnée est un tableau trié. Votre algorithme doit avoir une meilleur complexité que celui de la question précédente. Quelle est cette complexité en fonction de la taille n du tableau ?

Question 3. (1 point) Est-il intéressant du point de vue de la complexité de trier le tableau pour traiter ce problème ? Justifiez.

2 Listes chaînées

Question 4. Écrivez deux algorithmes qui fusionnent deux listes triées :

Algorithme : fusion(**d** L1 : Liste d'entiers, **d** L2 : Liste d'entiers) : Liste d'entiers
Données : $L1$ et $L2$ deux listes d'entiers triées par ordre croissant
Résultat : renvoie la liste d'entiers triée qui contient tous les éléments de $L1$ et tous les éléments de $L2$. L'algorithme recopie les éléments de $L1$ et de $L2$.

Par exemple si les listes sont $L1 = (1, 4, 5, 9)$ et $L2 = (2, 4, 7)$, `fusion(L1,L2)` renvoie la liste $(1, 2, 4, 4, 5, 7, 9)$. L'algorithme doit dupliquer les éléments des listes $L1$ et $L2$. Vous pouvez utiliser l'opération `créerListe` ainsi que les opérations d'insertion, en début ou fin de liste.

1. **(2,5 points)** Écrivez une version itérative de l'algorithme `fusion`. Quelle est sa complexité en fonction de n, nombre total d'éléments des deux listes ?
2. **(2,5 points)** Écrivez une version récursive de l'algorithme `fusion`. Quelle est sa complexité en fonction de n, nombre total d'éléments des deux listes ?

Question 5. (3 points) Écrivez un algorithme qui scinde une liste d'entiers en deux listes.

Algorithme : scinder(**dr** L : Liste d'entiers, **r** Li : Liste d'entiers, **r** Lp : Liste d'entiers)
Données : L est une liste d'entiers
Résultat : Li est la liste des éléments de L de rang impair (le 1^{er}, le $3^{ème}$, le $5^{ème}$, ...)
Lp est la liste des éléments de L de rang pair (le $2^{ème}$, le $4^{ème}$, ...)
L'algorithme ne crée aucune cellule. Il opère en modifiant le chaînage des éléments de L.
En fin d'algorithme, la valeur de L est quelconque.

Par exemple, si la donnée est la liste $L = (1, 3, 4, 5, 6, 8, 2)$, en fin d'algorithme les listes résultats sont $Li = (1, 4, 6, 2)$, $Lp = (3, 5, 8)$ et L quelconque.

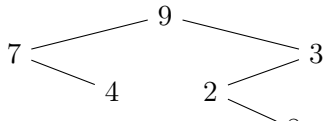
Votre algorithme ne doit recopier aucun élément de L, il doit modifier le chaînage des éléments de L. Vous ne devez donc utiliser ni l'opération `créerListe`, ni aucune opération d'insertion.

Quelle est la complexité de votre algorithme en fonction de la taille n de la liste L ?

3 Arbres Binaires

Question 6. (2 points) Complétez l'algorithme suivant qui calcule la plus petite valeur des noeuds d'un arbre :

Algorithme : valMin(**d** A : Arbre Binaire)
Données : A un arbre binaire **non vide**
Résultat : renvoie la plus petite valeur des noeuds de l'arbre A.

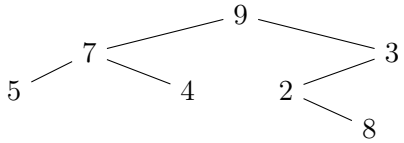


Par exemple si A est l'arbre ci-dessus l'algorithme renvoie 2.

Question 7. (2 points) Écrivez un algorithme qui calcule la liste des noeuds d'un arbre d'une profondeur donnée.

Algorithme : liValProf(**d** A : Arbre Binaire, **d** p : entiers) : Liste d'entiers
Données : A un arbre binaire ; p un entier naturel
Résultat : renvoie la liste des valeurs des noeuds de A qui sont à profondeur p

---



Par exemple si A est l'arbre ci-dessus
`liValProf(A,2)` renvoie la liste $(5, 4, 2)$ et `liValProf(A,5)` renvoie la liste vide.
Vous pouvez pour cette question utiliser une opération de concaténation. Il est inutile d'écrire l'algorithme de concaténation.

Question 8. (2 points) Pour répondre à cette question, vous pouvez utiliser l'algorithme `scinder` de la question 5 même si vous n'avez pas répondu à celle-ci.
Écrivez un algorithme construisant un arbre de hauteur minimum composé d'un ensemble de valeurs donné.

Algorithme : arbreMin(**d** L : Liste d'entiers) : Arbre Binaire
Données : L une liste d'entiers
Résultat : renvoie un arbre de hauteur minimum dont les noeuds sont étiquetés par les entiers de L

L'arbre résultat doit être de hauteur minimum, son nombre de noeuds doit être égal à la taille de la liste. L'ensemble des valeurs des noeuds de l'arbre doit être égal à l'ensemble des valeurs de la liste. Par contre leur répartition dans l'arbre est quelconque.

Par exemple si la liste $L = (4, 7, 3, 2, 8)$, `arbreMin(L)` renvoie un arbre de 5 noeuds, de hauteur 2, dont l'ensemble des valeurs est $\{2, 3, 4, 7, 8\}$. Le résultat peut être l'un des trois arbres ci–dessous :