UNIVERSITÉ DE MONTPELLIER

UE : HLIN301	
Session : 2	Durée de l'épreuve : 2 heures
Date : Avril 2017	Documents autorisés : 1 feuille A4 recto verso
Licence $2^{ème}$ année, parcours Informatique et Math-Informatique	Matériel utilisé : aucun

1 Tableaux

On souhaite calculer la valeur la plus fréquente dans un tableau d'entiers. En cas d'égalité on choisit comme résultat la plus petite des valeurs les plus fréquentes.

Exemples : La valeur la plus fréquente dans le tableau | 1 | 3 | 3 | 1 | 3 | 4 | 3 | est 3 qui apparaît quatre fois. Dans le tableau | 8 | 1 | 4 | 1 | 9 | 4 | 0 | 3 | 4 | 1 | les valeurs 1 et 4 apparaissent trois fois. Le résultat attendu pour ce tableau est l'entier 1.

On étudie ce problème dans deux cas.

Question 1. (**3,5 points**) Dans un premier temps, les valeurs sont des entiers quelconques.

Écrivez un algorihme `plusFrequente` :

Algorithme 1 : plusFréquente(**d** T : tableau de n entiers) : entier

Données : T est un tableau de n entiers quelconques
Résultat : Renvoie la valeur la plus fréquente dans T

On demande un algorithme de complexité dans le pire des cas optimale. Vous pouvez, si vous le souhaitez, utiliser un algorithme de tri. Dans ce cas, sans redonner l'algorithme, vous indiquerez le tri que vous utilisez et sa complexité. Quelle est la complexité dans le pire des cas de votre algorithme `plusFréquente` ?

Question 2. (**3,5 points**) On suppose à présent que les valeurs du tableau appartiennent à un intervalle d'entiers restreint et connu a priori. On prendra comme exemple le résultat d'un vote pour élire une personne parmi onze candidats en supposant que :

— chaque candidat est représenté par un numéro de 1 à 11
— il n'y a ni vote blanc, ni vote nul

Écrivez un algorihme `vainqueurElection` :

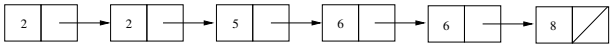
Algorithme 2 : vainqueurElection(**d** T : tableau de n entiers compris entre 1 et 11) : entier

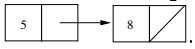
Données : T est un tableau de n entiers de l'intervalle $[1, 11]$, résultat d'un vote pour élire l'un des 11 candidats
Résultat : Renvoie la valeur la plus fréquente dans T, le numéro du candidat élu.

On demande un algorithme de complexité dans le pire des cas optimale. Vous pouvez, si vous le souhaitez, utiliser un algorithme de tri. Dans ce cas, sans redonner l'algorithme, vous indiquerez le tri que vous utilisez et sa complexité. Quelle est la complexité dans le pire des cas de votre algorithme `vainqueurElection` ?

2 Listes chaînées

Soit L une liste d'entiers triée. On veut obtenir la liste (triée) des éléments qui apparaissent une et une seule fois dans L.

Par exemple pour la liste 

, le résultat doit être la liste 

.

Vous devez écrire deux algorithmes réalisant cette opération. Vous pouvez pour cela utiliser des opérations sur les listes chaînées telles que `supprimer`, `insérerDébut`, `insérerFin`, `insérerAprès`, ...

Question 3. (**3,5 points**) Écrivez un algorithme qui ne modifie pas la liste donnée.

Algorithme 3 : liOccUnique(**d** L : ListeSC) : ListeSC

Données : L est une liste simplement chaînée triée d'entiers
Résultat : Renvoie une nouvelle liste, en recopiant les éléments apparaissant une seule fois dans L.

Quelle est la complexité de `liOccUnique` dans le pire des cas ?

Question 4. (**3,5 point**) Écrivez un algorithme qui modifie la liste donnée.

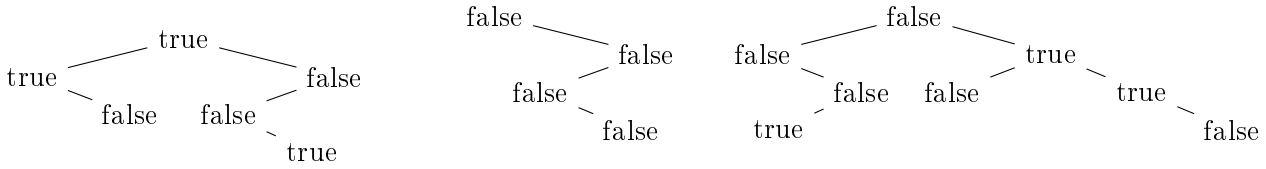
Algorithme 4 : oterOccMultiple(**dr** L : ListeSC)

Données : L une liste simplement chaînée d'entiers triée
Résultat : Supprime de la liste L les éléments appraissant plusieurs fois dans L.

`oterOccMultiple` ne crée aucune liste, il modifie la liste donnée. Quelle est la complexité de `oterOccMultiple` dans le pire des cas ?

3 Arbres Binaires

On considère la représentation d'un dictionnaire par un arbre binaire étiqueté par des booléens, vue en cours et TD. La figure ci–dessous donne 3 exemples d'arbres et les dictionnaires associés.



1) dico = $\{mot\ vide, a, bab\}$ 2) dico = $\emptyset$ 3) dico = $\{aba, b, bb\}$

Question 5. (**3 points**) Écrivez un algorithme `plusLgB` qui étant donné un arbre binaire représentant un dictionnaire renvoie la longueur du plus long mot composé exclusivement de b. Si aucun mot ne contient que des b, l'algorithme renvoie 0.

Exemples : Pour l'arbre 3) `lgMinDico` renvoie 2 car bb est le plus long mot composé que de b. Pour les arbres 1) et 2) l'algorithme renvoie 0 car les dictionnaires ne contiennent pas de mots composés que de b.

Question 6. (**1 point**) On souhaite ajouter la lettre b au début de chaque mot d'un dictionnaire. Écrivez un algorithme `ajoutBDebut` qui réalise cette opération en modifiant l'arbre représentant le dictionnaire.

Exemples : Appliquer l'algorithme `ajoutBDebut` sur l'arbre 3), modifie l'arbre pour qu'il représente le dictionnaire $\{baba, bb, bbb\}$. Appliquer l'algorithme `ajoutBDebut` sur l'arbre 1), modifie l'arbre pour qu'il représente le dictionnaire $\{b, ba, bbab\}$.

Question 7. (**2 points**) Écrivez un algorithme `ajoutBFin` qui modifie l'arbre en ajoutant la lettre b à la fin de chaque mot de son dictionnaire.

Exemples : Appliquer l'algorithme `ajoutBFin` sur l'arbre 3), modifie l'arbre pour qu'il représente le dictionnaire $\{abab, bb, bbb\}$. Appliquer l'algorithme `ajoutBFin` sur l'arbre 1), modifie l'arbre pour qu'il représente le dictionnaire $\{b, ab, babb\}$. Appliquer l'algorithme `ajoutBFin` sur l'arbre 2) donne en résultat un arbre qui représente le dictionnaire vide.