

## Examen 2023-2024 – Session 1 – Durée 2 h 30

Une feuille A4 manuscrite recto-verso autorisée. Les calculatrices et les téléphones (incluant smartphone, tablettes,... tout ce qui contient une interface réseau) sont interdits.

Les dictionnaires pour les personnes de langue étrangère sont autorisés.

**Le barème est indicatif et le total est sur 21 points.**

Comme d’habitude, vous êtes vivement encouragés à illustrer vos recherches et vos réponses par des schémas.

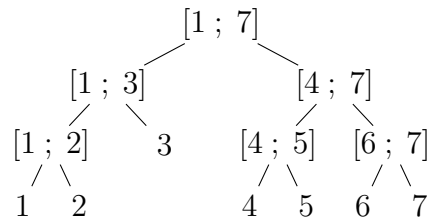
### 1 Algorithme des noeuds chapeaux (10 points)

On propose d’utiliser un arbre binaire pour gérer un calendrier de réservations sur une période de  $n$  jours :

- chaque feuille représente une journée, et les feuilles portent toutes les valeurs de 1 à  $n$  en ordre croissant de la gauche vers la droite ;
- chaque noeud interne est étiqueté par l’intervalle qui contient tous ses descendants.

L’arbre n’est pas forcément complet ni tassé mais il est équilibré : toutes ses feuilles sont au dernier ou à l’avant-dernier niveau.

Par exemple, un arbre possible pour une période d’une semaine est :



Pour effectuer la réservation d’un intervalle  $[x ; y]$ , on marque certains noeuds de l’arbre dont la réunion recouvre toutes les dates de  $[x ; y]$ .

Par exemple, dans l’arbre ci-dessus, pour réserver l’intervalle  $[3 ; 4]$ , on marque les feuilles 3 et 4. Mais pour réserver l’intervalle  $[4 ; 7]$ , il suffit de marquer directement le noeud  $[4 ; 7]$ , sans avoir à marquer aucun des descendants de ce noeud.

Dans tout cet exercice, on suppose que l’intervalle  $[x ; y]$  est inclus dans la période couverte par l’arbre, pour éviter des cas particuliers sans intérêt.

**Rappels et conventions** On manipule les arbres binaires à l’aide des primitives suivantes :

- L’appel de fonction  $estVide(A)$  renvoie vrai ou faux selon que l’arbre  $A$  est vide ou non.
- Pour un arbre non vide  $A$ , les appels de fonctions  $FG(A)$  et  $FD(A)$  renvoient respectivement le fils gauche et le fils droit de  $A$ .
- Pour un arbre non vide  $A$ , les appels de fonctions  $Inf(A)$  et  $Sup(A)$  renvoient respectivement les bornes inférieure et supérieure de l’intervalle qui étiquette la racine de  $A$ .
- Si  $A$  est une feuille,  $Inf(A)$  et  $Sup(A)$  renvoient toutes les deux la valeur du jour représenté par la feuille.

**Question 1 :** Méthode naïve (1 point)

Écrivez un algorithme qui marque chaque feuille correspondant à la période de réservation  $[x ; y]$ .

**Question 2 :** Complexité de la méthode naïve (1 point)

Quelle est la complexité de cette méthode dans le pire des cas ?

**Question 3 :** Un essai pour faire mieux (2 points)

On cherche à marquer moins de nœuds pour effectuer les réservations.

- Écrivez un algorithme qui prend en argument un arbre  $A$  et un intervalle  $[x ; y]$ , et qui détermine le nœud de  $A$  le plus profond qui contient entièrement  $[x ; y]$ .
- Pourquoi cet algorithme ne résout-il pas de façon satisfaisante le problème de la réservation d'un intervalle de temps ?

On va mettre au point une méthode qui permet de marquer un minimum de **nœuds chapeaux** représentant exactement  $[x ; y]$ , en les choisissant les plus hauts possibles dans l'arbre :

- si un nœud chapeau est marqué, tous ses descendants doivent être contenus dans  $[x ; y]$ ;
- si un nœud chapeau est marqué, tous ses nœuds parents ont au moins un descendant (une feuille) en dehors de  $[x ; y]$ .

Par exemple, pour réserver la période  $[1 ; 4]$  dans l'arbre ci-dessus, au lieu de marquer les feuilles 1, 2, 3 et 4 il suffit de marquer les deux nœuds  $[1 ; 3]$  et 4. On ne peut pas marquer moins de nœuds car tous les nœuds situés plus haut dans l'arbre contiennent les jours 5, 6 ou 7 qui ne sont pas dans la période voulue.

**Question 4 :** Exemple (1 point)

Avec cette méthode, quels seraient les nœuds chapeaux à marquer pour la période  $[2 ; 5]$  ?

**Question 5 :** Nœud chapeau gauche pour une réservation (2 points)

Dans un premier temps, écrivez un algorithme qui prend en argument un arbre et un intervalle  $[x ; y]$  et qui détermine le nœud chapeau le plus à gauche pour cette réservation.

**Question 6 :** Complexité de la recherche du nœud chapeau gauche (1 point)

Dans le pire des cas, quel serait la complexité de la recherche de ce nœud chapeau gauche ?

On suppose qu'on a déterminé de façon similaire le nœud chapeau droit pour la réservation  $[x ; y]$ . On cherche maintenant quels autres nœuds chapeaux sont nécessaires. On ne demande pas d'algorithme ici, on cherche juste à comprendre le principe à suivre.

**Question 7 :** Nœuds intermédiaires pour une réservation (2 points)

Dessinez un schéma qui représente, sur un arbre calendrier et pour un intervalle  $[x ; y]$  :

- les nœuds chapeaux gauche et droit de cette réservation ;
- le nœud déterminé à la question 3 ;
- la position de tous les autres nœuds chapeaux à marquer.

Expliquez en quelques lignes où se trouvent les nœuds chapeaux.

Évitez de prendre un exemple particulier, représentez plutôt la forme générale de l'arbre.

## 2 Tri de livres (11 points)

On dispose d'une collection de livres comme celle représentée ci-contre, dont les volumes sont numérotés de 1 à  $n$ . On souhaite trier ces livres par ordre de numéro (le numéro 1 en haut de la pile, et le numéro  $n$  en bas).

Cependant, on dispose de juste assez de place pour poser deux piles l'une à côté de l'autre. La seule opération permise est donc de prendre une partie (supérieure) d'une des deux piles et de la reposer sur l'autre pile.



Il est interdit de faire d'autres manipulations (retourner une partie des livres, garder un ou plusieurs livres en main, etc.). On suppose qu'il est toujours possible de lire les numéros des volumes sur la tranche.

On exprimera les algorithmes en français. Il n'est pas interdit d'utiliser des variables pour faciliter l'écriture, mais par exemple il n'y a aucun sens à utiliser des tableaux. L'utilisation de schémas est chaudement recommandée.

### Question 8 : (1 point)

Est-il possible d'effectuer le tri en déplaçant toujours un seul volume à la fois ?  
Si oui donnez un algorithme, sinon justifiez pourquoi ce n'est pas possible.

### Question 9 : Tri par insertion (2 points)

Écrivez un algorithme qui résout ce problème en s'inspirant du tri par insertion, c'est-à-dire qu'on traitera les volumes dans un ordre quelconque, en les insérant dans une pile (ou une partie de pile) déjà triée.

### Question 10 : Terminaison (1 point)

Donnez un **variant** qui justifie que cet algorithme se termine.

### Question 11 : Invariant (1 point)

Donnez un **invariant** vérifié par votre algorithme qui permettrait de démontrer qu'à la fin de l'algorithme la pile de livres est totalement triée. Vous exprimerez cet invariant :

- par une phrase en français
- **et** sous forme d'un schéma.

### Question 12 : Preuve (2 points)

Démontrez que l'invariant proposé :

- est vérifié à l'initialisation
- est maintenu par chaque itération

et déduisez-en que l'algorithme est correct.

### Question 13 : Coût en déplacements (1 point)

Déterminez, aussi précisément que possible, le coût au pire de cet algorithme en **nombre de déplacements de piles**.

**Question 14 :** Coût en lecture des numéros (*1 point*)

Déterminez, aussi précisément que possible, le coût au pire de cet algorithme **en nombre de fois où il faut lire le numéro d'un volume.**

**Question 15 :** Structure de données (*2 points*)

On souhaite représenter ces deux piles de livres dans un langage de programmation, par exemple en C. Proposez une structure de données qui permette d'effectuer l'opération de déplacement de pile en temps constant.

Vous donnerez :

- a. une description précise de la structure utilisée (tableaux, pointeurs, variables à mémoriser...);
- b. un schéma qui représente le contenu de cette structure;
- c. le pseudo-code de l'opération de déplacement de pile, incluant les arguments dont elle a besoin pour fonctionner.