COMPLEX - TD 06
2025-2026

Énoncé

Exercice 1 : Arbre de majorité

Un *arbre de majorité* est un arbre ternaire complet où chaque feuille est étiquetée soit 0, soit 1. La valeur d'une feuille est son étiquette et la valeur d'un nœud interne est la majorité de la valeur de ces trois fils. Nous considérons le problème du calcul de la valeur de la racine d'un arbre de majorité étant donnée la suite des 3^n étiquettes de ses feuilles.

1.a] Montrer que tout algorithme déterministe qui retourne la valeur d'un arbre de majorité doit examiner chaque feuille (et a une complexité en $\Omega(3^n)$).
Indication : Considérer le cas $n = 1$ puis utiliser un raisonnement par récurrence.

1.b] Proposer et analyser un algorithme probabibiliste (de type Las Vegas) qui retourne la valeur d'un arbre de majorité de profondeur n en temps espéré $O(c^n)$ pour une constante $c < 3$.
Indication : Considérer le cas $n = 1$ puis utiliser un raisonnement par récurrence.

Exercice 2 : Sélection rapide - Pire cas

Considérons l'algorithme de sélection rapide vu en cours et l'ensemble de ses exécutions possibles sur un tableau de longueur n ne contenant que des valeurs distinctes. Notons E l'événement qui correspond à une exécution qui demande le plus de comparaisons et notons E_k (pour $k \in \{1, \dots, n\}$) l'événement qui consiste à tirer comme pivot le plus petit élément ou le plus grand élément du sous-tableau lorsqu'il reste k éléments.

2.a] Justifier que $E \subseteq \bigcap_{k=1}^{n} E_k$.

2.b] Montrer que $\mathbb{P}(E) \leq \dfrac{2^{n-1}}{n!}$.

Exercice 3 : Sélection rapide - Cas moyen

Considérons l'algorithme de sélection rapide vu en cours et l'ensemble de ses exécutions possibles sur un tableau de longueur n ne contenant que des valeurs distinctes. Notons $T(n)$ le nombre moyen de comparaisons effectuées par l'algorithme dans le pire des cas pour déterminer un élément de rang donné et $T(n, j)$ le nombre moyen de comparaisons effectuées par l'algorithme dans le pire des cas pour déterminer un élément de rang $j \in \{1, \dots, n\}$ (de sorte que $T(n) = \max_{j \in \{1,\dots,n\}} T(n, j)$). Nous allons montrer que $T(n) \leq 4n$.

3.a] Montrer que

$$T(n, j) \leq n + \frac{1}{n}\left(\sum_{i=1}^{j-1} T(n-i) + \sum_{i=j}^{n} T(i-1)\right).$$

3.b] Supposons qu'il existe une constante $\alpha > 0$ telle que $T(k) \leq \alpha \cdot k$ pour tout entier $k \in \{1, \dots, n-1\}$. Montrer que

$$T(n, j) \leq n + \frac{\alpha}{2n}\left(-2j^2 + (2n+4)j + (n^2 - 3n - 2)\right).$$

3.c] En déduire que

$$T(n) \leq n + \frac{\alpha}{2n}\left(3n^2/2 - n\right)$$

3.d] Conclure.

Compléments

Exercice 4 : Élément majoritaire avec comparaison

4.a] Proposer un algorithme de complexité $O(n \log n)$ pour trouver un élément majoritaire (s'il existe) dans un tableau de longueur n en supposant que l'algorithme puisse comparer les éléments (pour un ordre total) et pas seulement tester leur égalité comme dans les algorithmes vus en cours.

Exercice 5 : Élément majoritaire - Algorithme de Boyer-Moore

Considérer l'algorithme suivant (de complexité $O(n)$ sur un tableau de longueur n).

Entrée: Un tableau T de longueur n
Sortie: m élément de T

```
m ← T[1] ; c ← 1
pour i de 2 à n faire
  si c = 0 alors
    m ← T[i] ; c ← 1
  sinon si T[i] = m alors
    c ← c + 1
  sinon
    c ← c − 1
  fin si
fin pour
si Est_Élément_Majoritaire ?(T, m) alors
  retourner m
sinon
  retourner Pas d'élément majoritaire
fin si
```

5.a] Montrer que cet algorithme retourne effectivement un élément majoritaire (s'il existe)

Exercice 6 : Pots et couvercles

Supposons que nous avons en notre possession n pots et n couvercles de tailles différentes. Chaque couvercle correspond à un pot et un seul (et réciproquement). Les pots et les couvercles sont presque de la même taille et il est impossible de dire si un pot est plus grand qu'un autre ou si un couvercle est plus grand qu'un autre. Cependant, si on essaie de faire correspondre un pot à un couvercle, le couvercle sera soit trop grand, soit trop petit, soit de la bonne taille exactement. Nous pouvons donc faire des comparaisons pot-couvercle, et chaque comparaison donne un résultat parmi les trois résultats possibles.

6.a] Proposer un algorithme qui, étant donné un pot, retrouve le couvercle correspondant. Donner sa complexité en nombre de comparaisons pot-couvercle (en fonction de n).

6.b] Proposer un algorithme probabiliste inspiré du tri rapide pour associer correctement chaque pot à son couvercle.

Soit $X_{i,j}$ (pour $i, j \in \{1, \ldots, n\}$) la variable aléatoire définie par $X_{i,j} = 1$ si, à un moment lors de l'exécution de votre algorithme, le pot i est comparé au couvercle j, et $X_{i,j} = 0$ sinon.

6.c] Exprimer en fonction des $X_{i,j}$ (pour $i, j \in \{1, \ldots, n\}$) le nombre $T(n)$ de comparaisons pot-couvercle effectuées par l'algorithme.

6.d] Donner une formule pour $\mathbb{E}(X_{i,j})$ pour $i, j \in \{1, \ldots, n\}$.

6.e] Montrer que $\mathbb{E}(T(n)) = O(n \log n)$.

Exercice 7 : Médian des médians

Nous allons utiliser une variante du tri rapide pour déterminer le k-ième élément d'un ensemble de n éléments. Nous supposons que les n clés sont distinctes deux à deux.

7.a] Décrire une procédure SÉLECTIONNER qui utilise un pivot de façon similaire à l'algorithme de tri rapide qui détermine le k-ième é lément du tableau.

7.b] Montrer que que sans hypothèse particulière sur le choix du pivot, la fonction SÉLECTIONNER peut réaliser $O(n^2)$ comparaisons.

7.c] Montrer que si le choix du pivot garantit que la taille du sous-tableau de l'appel récursif ne dépasse pas αn où $\alpha < 1$, la complexité en nombre de comparaisons de la fonction Sélectionner est $O(n)$.

On considère l'algorithme de choix du pivot suivant :

— Découper le tableau en $n/3$ blocs $\{B_1, \ldots, B_{n/3}\}$ de trois éléments ;
— Déterminer les éléments médians m_k des B_k, $k \in \{1, \ldots, n/3\}$;
— Utiliser l'algorithme récursivement pour déterminer l'élément médian p parmi $m_1, \ldots, m_{n/3}$;
— Déterminer le rang de p dans le tableau, puis utiliser l'algorithme récursivement sur une partie du tableau.

7.d] Montrer que le pivot choisi est strictement supérieur à au moins $n/3 - O(1)$ éléments de t et est inférieur ou égal à au moins $n/3 + O(1)$ éléments de t. En déduire que le sous-tableau de l'appel récursif est de taille au plus égale à $2n/3$.

7.e] Donner la complexité de la fonction SÉLECTIONNER.

7.f] Que devient la récurrence si, pour le choix du pivot, on découpe le tableau dans des blocs de 5 éléments plutôt que de 3 ? Montrer que la complexité est alors meilleure.