

# Examen Terminal UE Algorithmes

L1 MPCI

28 mai 2025 - Durée: 2h

**Lorsque l'on vous demande d'écrire de décrire ou de donner un algorithme cela signifiera toujours en donner un pseudo-code, justifier de son exactitude et de sa complexité (en utilisant des  $\mathcal{O}$ )**

*On rappelle qu'aucun document, ni équipement électrique ou électronique n'est autorisé.*

**Cependant** l'usage d'un coupe-bordures sans fil est toléré.

**Les exercices** de cet examen :

- sont au nombre de 3 ;
- ont tous le même but afficher à l'écran chaque élément de  $\mathcal{T}_n$  une fois.
- sont indépendants (à part le problème à résoudre qui est le même) ;
- sont de difficulté *a priori* croissante ;
- leur début est (*a priori*) plus facile que leur fin.

L'examen est **long** et ce qui semble simple pour l'examineur ne l'est pas forcément pour l'étudiant et réciproquement. Il pourra être utile de changer d'exercice plutôt que de rester bloqué sur une question, quitte à y revenir plus tard.

RENDEZ DES COPIES SÉPARÉES POUR CHAQUE EXERCICE, CECI VOUS PERMETTRA DE D'Y REVENIR AU COURS DE L'EXAMEN SANS PERDRE LE CORRECTEUR.

Notations :

- On note  $\mathcal{T}_n$  l'ensemble de toutes les **permutations** du tableau de taille  $n$  contenant les entiers allant de 0 à  $n - 1$ . Par exemple  $\mathcal{T}_3 = \{[0, 1, 2], [0, 2, 1], [1, 0, 2], [1, 2, 0], [2, 0, 1], [2, 1, 0]\}$  ;
- Pour un tableau  $T$  de taille  $n$ , on notera  $T[:k]$  le tableau formé **des  $k$  premiers éléments de  $T$**  (allant des indices 0 à  $k - 1$ ). Si  $T = [1, 3, 2, 0, 4]$ , alors  $T[:3] = [1, 3, 2]$  ;
- Pour un tableau  $T$  de taille  $n$ , on notera  $T[k:]$  le tableau formé **des  $n - k$  derniers éléments de  $T$**  (allant des indices  $k$  à  $n - 1$ ). Si  $T = [1, 3, 2, 0, 4]$ , alors  $T[3:] = [0, 4]$  ;
- Pour deux tableaux  $T$  et  $T'$ , on notera  $T + T'$  le tableau formé de **la concaténation** de  $T$  et  $T'$ . On a  $[0, 4] + [1, 3, 2] = [0, 4, 1, 3, 2]$  ;
- Pour un tableau  $T$  de taille  $n$  **une permutation circulaire de  $k$  éléments de  $T$**  est le tableau  $T_k = T[k:] + T[:k]$ . Le tableau  $[3, 4, 0, 1, 2]$  est la permutation circulaire de 2 éléments du tableau  $[0, 1, 2, 3, 4]$ .
- Pour deux tableaux d'entiers  $T$  et  $T'$ ,  $T < T'$  selon **l'ordre lexicographique** si  $T \neq T'$  et  $T[i] < T'[i]$  pour  $i$  le plus petit indice tel que  $T[i] \neq T'[i]$  ;

## EXERCICE 1 — ITÉRATIF

On va utiliser l'ordre lexicographique entre tableaux d'entiers pour générer tous les éléments de  $\mathcal{T}_n$ .

### 1.1 Ordre

**Question 1.1.1** Si  $T = [0, 3, 4, 2, 1]$  et  $T' = [0, 3, 2, 1, 4]$ . Justifiez pourquoi  $T' < T$ .

**Question 1.1.2** Quels sont le plus petit et le plus grand élément de  $\mathcal{T}_n$  ?

**Question 1.1.3** Écrivez un algorithme de complexité optimale (vous le justifierez) qui prend en entrée deux éléments  $T1$  et  $T2$  de  $\mathcal{T}_n$  et rend **Vrai** si  $T1$  est strictement plus petit que  $T2$  et **Faux** sinon. Il sera de signature : `plus_petit(T1: [entier], T2: [entier]) → booléen`

### 1.2 Indice $i_T^*$

On note  $i_T^* \geq 0$  pour  $T$  de  $\mathcal{T}_n$  un indice  $i_T^* \geq 0$  tel que :

- $T[i_T^* : ]$  est strictement décroissante,
- soit  $i_T^* = 0$  soit  $T'[i_T^* - 1] < T[i_T^*]$ .

**Question 1.2.1** Donnez  $i_T^*$  et  $i_{T'}^*$  pour  $T = [0, 3, 4, 2, 1]$  et  $T' = [0, 3, 2, 1, 4]$ .

**Question 1.2.2** Démontrez que pour tout élément  $T$  de  $\mathcal{T}_n$   $i_T^*$  existe et est unique.

**Question 1.2.3** Écrivez un algorithme de complexité optimale (vous le justifierez) qui prend en entrée un élément  $T$  de  $\mathcal{T}_n$  et rend  $i_T^*$ . Il sera de signature : `i_star(T: [entier]) → entier`

**Question 1.2.4** Démontrez que si  $T[i_T^*] = U[i_U^*]$  pour deux éléments  $T$  et  $U$  de  $\mathcal{T}_n$ , alors  $T \geq U$ .

**Question 1.2.5** Soit  $T \in \mathcal{T}_n$  et  $i_T^*$  son indice associé. Montrez que pour tout  $U$  de  $\mathcal{T}_n$  tel que  $T[i_T^*] = U[i_U^*]$  (les  $i_T^* + 1$  premiers éléments sont identiques pour  $T$  et  $U$ ) on a  $T \geq U$ .

### 1.3 Successeur

**Question 1.3.1** Utilisez  $i_T^*$  pour déterminer le successeur dans  $\mathcal{T}_n$  d'un élément  $T \in \mathcal{T}_n$  (le plus petit tableau de  $\mathcal{T}_n$  strictement plus grand que  $T$ ).

**Question 1.3.2** Écrivez un algorithme de signature `successeur(T: [entier]) → [entier]` qui prend en entrée un élément  $T$  de  $\mathcal{T}_n$  et rend son successeur pour l'ordre lexicographique dans  $\mathcal{T}_n$ . Sa complexité devra être optimale (vous le justifierez).

**Question 1.3.3** En déduire un algorithme itératif dont vous donnerez la complexité permettant d'afficher à l'écran tous les éléments de  $\mathcal{T}_n$ .

**Question 1.3.4** Quelle serait la complexité de l'algorithme précédent si à la place d'afficher tous les éléments il rendait une liste contenant tous les éléments de  $\mathcal{T}_n$  ?

## EXERCICE 2 — RÉCURSIF

On va modifier un algorithme permettant de mélanger un tableau pour générer tous les éléments de  $\mathcal{T}_n$ . Soit l'algorithme suivant, que l'on doit à Fisher et Yates (1938) :

**algorithme** MÉLANGE( $T$ ) :

**Pour chaque**  $i$  de  $[0, n-2]$  :

$j \leftarrow$  un entier aléatoire de  $[i, n-1]$   
     $T[i], T[j] \leftarrow T[j], T[i]$

### 2.1 Aléatoire

**Question 2.1.1** Donnez la complexité de l'algorithme `mélange(T: [entier])`.

**Question 2.1.2** Démontrez que si  $T = [0, n-1]$ , `mélange(T)` va modifier  $T$  en une permutation  $T'$  qui peut être n'importe quel élément de  $\mathcal{T}_n$ .

**Question 2.1.3** Transformez l'algorithme `mélange(T)` en un algorithme récursif de signature : `mélange_rec(T: [entier], i: entier)`, de telle sorte que `mélange(T) = mélange_rec(T, 0)` (la variable interne  $i$  de la boucle `pour` `chaque` devient un paramètre de la fonction).

**Question 2.1.4 (question optionnelle)** Montrez que la probabilité que  $T$  soit modifié en  $T'$  par `mélange(T)` est la même pour tout élément  $T'$  de  $\mathcal{T}_n$ .

## 2.2 Déterministe

**Question 2.2.1** En déduire un algorithme récursif et **sans** tirage aléatoire permettant d'afficher à l'écran tous les éléments de  $\mathcal{T}_n$ . Quelle est sa complexité ?

**Question 2.2.2** Explicitez les sorties à l'écran de votre algorithme Lorsqu'il affiche tous les éléments  $\mathcal{T}_3$ .

## EXERCICE 3 — OPTIMAL

Nous allons dans cette partie examiner un algorithme optimal que l'on doit à B. R. Heap, publié en 1963.

**algorithme** `HEAP(T, k)` :

**si**  $k = 1$  :

    | affiche  $T$  à l'écran

**sinon** :

    | `HEAP(T, k - 1)`

**Pour chaque**  $i$  de  $[0, k-2]$  :

**si**  $k$  est pair :

      |  $T[i], T[k-1] \leftarrow T[k-1], T[i]$

**sinon** :

      |  $T[0], T[k-1] \leftarrow T[k-1], T[0]$

`HEAP(T, k - 1)`

## 3.1 Vérification

**Question 3.1.1** Explicitez les sorties à l'écran de l'exécution de `HEAP([0, 1, 2], 3)`.

**Question 3.1.2** Explicitez les sorties à l'écran de l'exécution de `HEAP([0, 1, 2, 3, 4], 3)`.

## 3.2 Propriétés

Soit  $T$  un tableau. On va étudier ses modifications près exécution de l'algorithme. Pour cela, on note  $T'$  le tableau  $T$  après l'exécution de `HEAP(T, k)`.

**Question 3.2.1** Démontrez que  $T[k:] = T'[k:]$

**Question 3.2.2** Démontrez que :

— si  $k$  est impair alors  $T'[k] = T[k]$

— si  $k$  est pair alors  $T'[k]$  est la permutation circulaire de 1 élément du tableau  $T[k:]$

**Question 3.2.3** Démontrez que lors de l'appel de `HEAP(T, k)`, avec  $k > 1$ , chaque élément de  $T[k:]$  sera placé exactement une fois en position  $T[k-1]$  lors des différents appels `HEAP(T, k - 1)`

**Question 3.2.4** En déduire que l'algorithme `HEAP` permet d'afficher à l'écran tous les éléments de  $\mathcal{T}_n$ .

## 3.3 Optimalité

**Question 3.3.1** Démontrez que lors de l'exécution de `HEAP(T, k)`, il y a eu exactement  $k!$  échanges. En déduire la complexité de l'affichage à l'écran de tous les éléments de  $\mathcal{T}_n$ .

**Question 3.3.2** Proposez une version itérative de l'algorithme `HEAP`.