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

On rappelle qu'aucun document, ni équipement électrique ou électronique n'est autorisé.
Cependant *l'usage d'une scie-sauteuse sans fil est tolérée.*

Les exercices :
- sont au nombre de 3 ;
- sont de difficulté croissante (le premier est le plus facile et vaudra moins de points que le troisième) ;
- sont indépendants (à part le problème à résoudre qui est le même) ;
- leur début est plus facile que leur fin (qui vaudra donc plus de points).

Rendez des copies séparées pour chaque exercice, ceci vous permettra de reprendre les exercices 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 $n-1$;
- 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 $k-1$) ;
- Pour un tableau T n, on notera $T[k:]$ le tableau formé des $n-k$ derniers éléments de T (allant des indices k à $n-1$) ;
- Pour deux tableaux T et T', on notera $T+T'$ le tableau formé de la concaténation de T et T'.

Les trois exercices de cet examen ont pour but d'afficher à l'écran chaque élément de $\mathcal{T}_n$ une seule fois.

Exercice 1 – Itératif

On va utiliser l'**ordre lexicographique** (c'est l'ordre des mots dans un dictionnaire) entre tableaux pour générer $\mathcal{T}_n$. On rappelle que pour cet ordre $T < T'$ si $T \neq T'$ et $T[i] < T'[i]$ pour i le plus petit indice tel que $T[i] \neq T'[i]$.

1.1 Ordre

Question 1.1.1 Quels sont le plus petit et le plus grand élément de $\mathcal{T}_n$?

Question 1.1.2 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 grand que $T2$ et `Faux` sinon. Il sera de signature : `plus_grand(T1: [entier], T2: [entier]) → booléen`

1.2 Indice $i^\star_T$

Question 1.2.1 Démontrez que pour tout élément T de $\mathcal{T}_n$, il existe un indice $i^\star_T \geq 0$ tel que :
- $T[i^\star_T :]$ est strictement décroissante,
- soit $i^\star_T = 0$ soit $T'[i^\star_T - 1] < T[i^\star_T]$.

Question 1.2.2 Écrivez un algorithme de complexité optimale (vous le justifierez) qui prend en entrée un élément T de $\mathcal{T}_n$ et rend $i^\star_T$. Il sera de signature : `i_star(T: [entier]) → entier`

Question 1.2.3 Démontrez que si $T[: i^\star_T] = U[: i^\star_T]$ pour deux éléments T et U de $\mathcal{T}_n$, alors $T \geq U$.

Question 1.2.4 Soit $T \in \mathcal{T}_n$. Quel est le plus petit élément U de $\mathcal{T}_n$ tel que $T[: i^\star_T] = U[: i^\star_T]$?.

1.3 Successeur

Question 1.3.1 Utilisez $i^\star_T$ pour déterminer le successeur (pour l'ordre lexicographique) dans $\mathcal{T}_n$ d'un élément $T \in \mathcal{T}_n$.

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

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

Exercice 2 – Récursif

2.1 Aléatoire

Soit l'algorithme suivant de Fisher et Yates (1938), aussi appelé *mélange de Knuth*.

algorithme MÉLANGE(T) **:**
  Pour chaque i de [0, n-2] **:**
    $j \leftarrow$ un entier aléatoire de [i, n-1]
    $T[i]$, $T[j]$ $\leftarrow T[j]$, $T[i]$

Question 2.1.1 Donnez la complexité de cet algorithme.

Question 2.1.2 Démontrez que `mélange(T)` avec $T = [0, n-1]$ va modifier T en une permutation T' de $\mathcal{T}_n$ de manière uniforme (la probabilité que T soit modifiée en T' est la même pour tout élément T' 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 En déduire un algorithme récursif permettant d'afficher à l'écran tous les éléments de $\mathcal{T}_n$.

Question 2.1.5 Explicitez les sorties à l'écran de votre algorithme Lorsqu'il génère $\mathcal{T}_3$.

Exercice 3 – Optimal

Nous allons dans cette partie examiner un algorithme optimal que l'on doit à Heap.

```
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] ←T[k − 1], T[i]
        | sinon :
        |     | T[0], T[k − 1] ←T[k − 1], T[0]
        | HEAP(T, k − 1)
```

3.1 Vérification

Question 3.1.1 Quels paramètres utiliser pour que l'algorithme HEAP affiche à l'écran les éléments de $\mathcal{T}_n$?

Question 3.1.2 Explicitez les sorties à l'écran de votre algorithme Lorsqu'il génère $\mathcal{T}_3$.

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 $\textsc{heap}(T, k)$.

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 $T'[:k] = [T[k-1]] + T[1:k-1]$ ($T'[:k]$ est une permutation circulaire de un élément du tableau initial)

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

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 $\textsc{heap}(T, k)$, 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.