Examen du 9 décembre 2025

Consignes.
L'examen dure 2 heures. Aucun document n'est autorisé.
Le sujet est constitué de trois exercices indépendants. Le barème total est sur 30 points. Toute question non résolue peut être admise dans la suite. ***Les réponses doivent être justifiées et correctement rédigées.***

Exercice 1 (sur 7 pts). *Questions de cours*

1. Décrire en une phrase chacune les deux grandes familles d'algorithmes probabilistes, en donnant leur nom.

2. On tire une fonction de hachage $h : \{0, \dots, N-1\} \to \{0, \dots, m-1\}$ uniformément dans une famille universelle $\mathcal{H}$. Étant donné deux clés $x \neq y$, quelle est la probabilité que $h(x) \neq h(y)$?

3. On réalise le TAD File de priorité par un tas stocké dans un tableau dynamique : on représente le tas dans un tableau dynamique, dont on augmente ou réduit la taille en cas de besoin. On note n le nombre d'éléments de la file de priorité.

- **i.** Quel est le coût dans le pire cas de l'opération EXTRAIRE ?
- **ii.** Quel est le coût amorti d'une série d'appels à INSÉRER et EXTRAIRE ?

4.
- **i.** Dessiner l'arbre binaire de recherche $\mathcal{A}$ obtenu en insérant dans un arbre initialement vide les entiers 5, 9, 7, 3, 6, 4, 8, 2 puis 1 dans cet ordre. *On ne veut que le résultat final.*
- **ii.** On supprime la racine de $\mathcal{A}$ grâce à l'algorithme SUPPRIMER vu en cours. Dessiner l'arbre obtenu.
- **iii.** Existe-t-il un autre arbre binaire de recherche, contenant les mêmes éléments que $\mathcal{A}$ après suppression de sa racine, et de hauteur strictement inférieure ? Justifier.

Exercice 2 (sur 8 pts). *Matrices de Toeplitz*

Une matrice $M \in \mathbb{K}^{n\times n}$ sur un corps $\mathbb{K}$ est une *matrice de Toeplitz* si $M_{i+1,j+1} = M_{i,j}$ pour $0 \le i, j < n-1$ comme dans l'exemple suivant :

$$\begin{bmatrix} 1 & 2 & -4 & 0 \\ -6 & 1 & 2 & -4 \\ 3 & -6 & 1 & 2 \\ 4 & 3 & -6 & 1 \end{bmatrix}.$$

Remarques : les indices des matrices et vecteurs vont de 0 *à* $n-1$ *; les complexités sont exprimées en comptant uniquement le nombre d'opérations dans le corps* $\mathbb{K}$ *(en particulier on ignore le coût des copies de données).*

1. Montrer que la somme de deux matrices de Toeplitz est une matrice de Toeplitz, et qu'on peut la calculer en $O(n)$ additions dans $\mathbb{K}$.

On admet que la soustraction de matrices de Toeplitz a le même coût que leur addition. On s'intéresse au calcul du produit $\vec{w} = M \cdot \vec{v}$ où $\vec{v} \in \mathbb{K}^n$ est un vecteur.

2. Écrire en pseudo-code un algorithme de complexité quadratique pour ce calcul, *qui n'utilise pas la structure Toeplitz de la matrice.*

On suppose maintenant que $n = 2^k$ pour un certain $k > 0$. On coupe M en quatre sous-matrices et $\vec{v}$ en deux sous-vecteurs, tous de dimensions $n/2$:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ et } \vec{v} = \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix}.$$

3. **i.** Montrer que M étant une matrice de Toeplitz, alors $A = D$.

ii. Exprimer $\vec{w} = M \cdot \vec{v}$ en fonction de A, B, C, $\vec{x}$ et $\vec{y}$.

iii. Montrer qu'on peut calculer $\vec{w}$ à l'aide de trois produits matrice-vecteur de dimensions $n/2$. *L'un des produits à calculer est* $A \cdot (\vec{x} + \vec{y})$.

iv. Écrire en pseudo-code un algorithme de type « diviser pour régner » pour le calcul de $M \cdot \vec{v}$ et analyser sa complexité.

Exercice 3 (sur 15 pts). *Le sac-à-dos*

On rappelle le problème du sac-à-dos :

— *Entrée :* une liste de couples $(t_0, v_0), \ldots, (t_{n-1}, v_{n-1})$ et un entier S, où $t_i \leq S$ pour tout i ;
— *Sortie :* la *valeur* maximale $\sum_{i \in I} v_i$ où $I \subset \{0, \ldots, n-1\}$ est un sous-ensemble tel que $\sum_{i \in I} t_i \leq S$.

Chaque couple représente un *objet* de *taille* t_i et de *valeur* v_i, et S est la *taille du sac-à-dos*. **Les trois questions de cet exercice sont indépendantes.**

1. **Recherche exhaustive.** On souhaite calculer une solution optimale en essayant tous les sous-ensembles I possibles.
   - **i.** Comment représenter un sous-ensemble $I \subset \{0, \ldots, n-1\}$ par un mot binaire ? *Préciser la taille du mot binaire.*
   - **ii.** Quel algorithme permet de passer d'un sous-ensemble au suivant ? Quelle est sa complexité en pire cas et sa complexité amortie ? *Pas de justification demandée.*
   - **iii.** En déduire le coût d'un algorithme qui résout le problème du sac-à-dos par recherche exhaustive.

2. **Programmation dynamique.** Pour $0 \leq m \leq n$ et $u \geq 0$, on note $S(m,u)$ la taille minimale d'un sac-à-dos dont la valeur totale est $\geq u$ et qui ne contient que des objets parmi les m premiers (dont les indices vont de 0 à $m-1$). On pose $S(m,u) = +\infty$ si la somme des valeurs des m premiers objets est $< u$. *Formellement*,

$$S(m,u) = \begin{cases} +\infty \text{ si } \sum_{i=0}^{m-1} v_i < u \\ \min\left\{\sum_{i \in I} t_i : I \subset \{0, \ldots, m-1\} \wedge \sum_{i \in I} v_i \geq u\right\} \text{ sinon} \end{cases}$$

   - **i.** Que vaut $S(m,0)$ pour tout m ?
   - **ii.** Écrire une formule récursive pour $S(m,u)$. *On peut distinguer les cas* $u < v_{m-1}$ *et* $u \geq v_{m-1}$.
   - **iii.** Exprimer la solution au problème du sac-à-dos à l'aide des $S(n,u)$, c'est-à-dire la valeur maximale étant donné une taille S de sac-à-dos.
   - **iv.** Écrire en pseudo-code un algorithme de programmation dynamique pour le problème du sac-à-dos, et analyser sa complexité.

3. **Approximation.** On décrit un algorithme d'approximation : on trie les objets par *ratio* v_i/t_i décroissant, on calcule l'indice maximal k tel que $\sum_{i=0}^{k-1} t_i \leq S$ et on renvoie $\max\left(\sum_{i=0}^{k-1} v_i, v_k\right)$.
   - **i.** Appliquer l'algorithme avec les objets $(1,10)$, $(2,19)$, $(4,35)$ et $(3,26)$, et une taille de sac-à-dos $S = 4$. Le résultat est-il optimal ?

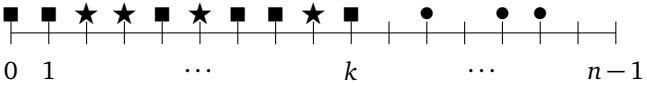

FIGURE 1 – Illustration où I_0 est représenté par les ★, I_1 par les ■ et I_2 par les ●.

Soit OPT la valeur optimale et $I^* \subset \{0,\dots,n-1\}$ tel que $\sum_{i\in I^*} v_i = \text{OPT}$ et $\sum_{i\in I^*} t_i \leq S$. On définit $I_0 = I^* \cap \{0,\dots,k\}$, $I_1 = \{0,\dots,k\} \setminus I_0$ et $I_2 = I^* \setminus I_0$ tels que $I_0 \sqcup I_1 = \{0,\dots,k\}$ et $I_0 \sqcup I_2 = I^*$. (*cf.* Figure 1).
On note $V_j = \sum_{i\in I_j} v_i$ et $S_j = \sum_{i\in I_j} t_i$ pour $0 \leq j \leq 2$, et $r = v_{k+1}/t_{k+1}$.

ii. Montrer que $V_1 \geq rS_1 > rS_2 \geq V_2$.
iii. Exprimer OPT en fonction des V_j, et en déduire que $V_0 + V_1 \geq \text{OPT}$.
iv. En déduire le facteur d'approximation de l'algorithme proposé. *Indication. Pour x, $y > 0$,* $\max(x,y) \geq \frac{1}{2}(x+y)$.