

Partiel du 10 Novembre 2023

*Document autorisé : une feuille A4 manuscrite. Les questions (Pa) seront rédigées en détail et pour les autres, un petit paragraphe suffit. Les questions *** sont hors barème.*

- Exercice 1 - Questions de cours

1. Rappeler brièvement l'idée de l'algorithme de Strassen (sans les détails!) et sa complexité.
2. On se donne des intervalles $[x_i, y_i]$ de réels avec $i \in [n]$. Trouver en temps polynomial un ensemble maximal d'intervalles deux à deux disjoints.
3. Qu'implique l'existence d'un algorithme polynomial calculant une clique de taille maximale dans un graphe?

- Exercice 2 - Points Bonus. Parmi les quatre questions (Pa), l'une est un problème NP-difficile et les trois autres utilisent les paradigmes : diviser pour régner, programmation dynamique, et algorithme glouton. Trouver la bijection entre les questions (Pa) et les quatre cas possibles.

- Exercice 3 - Algorithme Primal-Dual. Votre laboratoire se lance dans un nouveau concept de "brasserie du parallélisme" : Les verres ont des capacités très diverses, et chaque robinet de la tireuse peut remplir (de façon égale) un sous-ensemble de verres (un même verre pouvant être rempli par plusieurs robinets). Ainsi, une *commande* est un vecteur d'entiers positifs $(c_1, \dots, c_n)$ (avec c_i la capacité du verre i) et à chaque robinet R_j , avec $j = 1, \dots, m$, est associé un ensemble $I_j \subseteq [n]$ (où R_j peut remplir de façon égale les verres i vérifiant $i \in I_j$).

1. (Pa) Un *service* est une affectation à chaque R_j d'une quantité $q_j \geq 0$ vérifiant qu'aucun verre ne dépasse sa capacité (i.e. pour tout $i \in [n]$, $\sum_{j:i \in I_j} q_j \leq c_i$). Un service est *maximal* si aucune valeur q_j ne peut être augmentée sans qu'un verre ne déborde. Existe-t-il un algorithme polynomial qui calcule un service maximal? L'entrée est codée en binaire, i.e. de taille $O(n.m + \sum_{i=1, \dots, n} \log c_i)$.
2. Dans un service maximal $S = (q_1, \dots, q_m)$, un verre i est *plein* si $\sum_{j:i \in I_j} q_j = c_i$. On note c la somme des capacités des verres pleins de S . On note p_j le nombre de verres pleins sous le robinet R_j (i.e. le nombre de verres pleins i tels que $i \in I_j$). Montrer que $c = \sum_{j=1, \dots, m} p_j \cdot q_j$.
3. On se donne un sous-ensemble $I \subseteq [n]$ tel que $I \cap I_j \neq \emptyset$ pour tout $j \in [m]$. Montrer que pour tout service $(q_1, \dots, q_m)$, on a $\sum_{j=1, \dots, m} q_j \leq \sum_{i \in I} c_i$.
4. * Le problème $TRANSVERSAL_{10}$ admet en entrée un ensemble X dont chaque élément a un coût positif et un ensemble E de parties de taille 10 de X et retourne un sous-ensemble T de X de coût total minimal qui intersecte tous les éléments de E . Proposer un algorithme polynomial pour $TRANSVERSAL_{10}$ qui retourne une solution au pire 10 fois supérieure à l'optimum.

- **Exercice 4 - Tableau à compléter.** On se donne un tableau $T[1 \dots n, 1 \dots m]$ dont les entrées sont $-1, 1$ ou bien $*$.

1. (Pa) Existe-t-il un algorithme polynomial (en nm) qui décide si l'on peut transformer des entrées 1 ou -1 en entrées $*$ de telle sorte qu'aucune ligne ne soit que des $*$ et qu'aucune colonne ne possède à la fois un 1 et un -1 ?
2. Proposer un algorithme polynomial décidant si on peut remplacer chaque entrée $*$ par un réel de sorte que la somme des valeurs sur toute ligne et toute colonne soit 0 .
3. *** Résoudre la question précédente lorsque $n = m$ en temps $O(n^2)$.

- **Exercice 5 - Élément Majoritaire** Dans un tableau $T[1 \dots n]$ d'entiers, un élément $T[i]$ est *majoritaire* s'il apparaît plus de $n/2$ fois dans T . Le problème *MAJORITAIRE* admet en entrée T et retourne un élément majoritaire s'il existe, et FAUX sinon.

1. Montrer que l'on peut résoudre en temps $O(n \log n)$ le problème plus général de décider si une entrée apparaît au moins k fois.
2. Utiliser un algorithme du cours pour résoudre *MAJORITAIRE* en $O(n)$ (on commencera par trouver un bon candidat). Adapter votre algorithme pour trouver toutes les entrées qui apparaissent au moins $n/10$ fois.
3. (Pa) Pour la suite de l'exercice, on suppose que l'on a seulement accès à une boîte noire répondant à " $T[i] = T[j]$?". Dans cette question, n est une puissance de 2. Proposer un algorithme en $O(n)$ pour *MAJORITAIRE*. On pourra utiliser un problème auxiliaire *CANDIDATMAJORITAIRE* retournant x s'il apparaît au plus $k > n/2$ fois dans T et que les autres valeurs apparaissent au plus $n - k$ fois, ou bien retourne FAUX.
4. On rappelle que la probabilité de tirer moins de $n/3$ fois "pile" en effectuant n tirages avec une pièce non biaisée est au plus $1/2^{cn}$ pour une constante $c > 0$. Proposer un algorithme randomisé de résolution de *MAJORITAIRE* en temps $O(n)$ qui est valide lorsqu'il retourne une valeur majoritaire et se trompe pour les réponses FAUX avec probabilité au plus $1/2^{c\sqrt{n}}$.
5. *** Proposer un algorithme linéaire déterministe pour la version boîte noire qui trouve toutes les entrées qui apparaissent au moins $n/10$ fois.

- **Exercice 6 - Collision Maximale** Votre directeur de stage a (encore) inventé une fonction de hachage qui calcule pour tout entier n un tableau d'entiers positifs $H[1 \dots n]$ dont les valeurs sont au plus n^{10} et tel que pour tout $I \neq J \subseteq [n]$, la somme S_I des $H[i]$ pour $i \in I$ est différente de la somme S_J des $H[i]$ pour $i \in J$.

1. Indiquez lui pourquoi une telle fonction ne peut pas exister pour tout n .
2. (Pa) Existe-t-il un algorithme en temps polynomial prenant en entrée un tableau d'entiers positifs $H[1 \dots n]$ dont les valeurs sont au plus n^{10} et retourne $I \neq J \subseteq [n]$ tels que $S_I = S_J$ avec S_I maximal lorsque deux tels sous-ensembles existent ?