Examen du 14 janvier 2022

*Seul document autorisé: une feuille A4 manuscrite. Justifier soigneusement vos réponses. Ne répondre aux questions * que si vous n'avez que cela à faire. On note* $[n] := \{1, \ldots, n\}$.

- Exercice 1 - NP-complétude. On suppose admis dans cette question que 3-SAT est NP-complet.

1. Montrer que décider s'il existe un stable (ensemble de sommets deux à deux non-adjacents) de taille k dans un graphe G est NP-complet.

2. Déduire de la question précédente que décider s'il existe un vertex-cover (ensemble de sommets qui intersecte toutes les arêtes) de taille k dans un graphe G est NP-complet.

3. L'entrée du problème de *couverture* est une collection de n sous-ensembles $X_1, \ldots, X_n$ de $[n]$ et un entier k. Déduire de la question précédente que décider s'il existe une famille de k parties choisies dans $X_1, \ldots, X_n$ dont l'union est $[n]$ est NP-complet.

- Exercice 2 - Point Bonus. Les cinq exercices suivants correspondent à cinq paradigmes: Glouton, diviser pour régner, algorithme randomisé, programmation dynamique, recherche locale (mais ils ne sont pas cités dans le bon ordre...). Retrouver la bonne bijection.

- Exercice 3 - Paradigme 1. On dit qu'un tableau $C[1, \ldots, m]$ est un *sous-tableau* d'un tableau $A[1, \ldots, n]$ s'il existe une fonction strictement croissante f de $[m]$ dans $[n]$ telle que $C[i] = A[f(i)]$ pour tout $i = 1, \ldots, m$. Si de plus f est de la forme $f(i) = i + k$ pour une constante k, on dit que C est un *facteur* de A.

1. Proposer un algorithme en $O(n^2)$ qui admet en entrée deux tableaux A et B de taille n et retourne la longueur d'un plus grand facteur commun.

2. Même question pour sous-tableau commun. En déduire un algorithme qui calcule une plus longue sous-suite croissante dans un tableau d'entiers.

3. * La généralisation directe à deux matrices A et B conduit à deux problèmes: plus grande sous-matrice carrée commune (lignes et colonnes pas forcément consécutifs) et plus grand bloc carré commun (lignes et colonnes consécutifs). Quelle est la complexité de ces deux problèmes?

- Exercice 4 - Paradigme 2. Une *inversion* dans un tableau $A[1, \ldots, n]$ d'entiers deux à deux distincts est un couple $i < j$ tel que $A[i] > A[j]$.

1. Proposer un algorithme en $O(n. \log n)$ (ou au pire en $O(n. \log^2 n)$) qui calcule le nombre d'inversions d'un tableau A en entrée.

2. * Lorsque les valeurs de $A[1, \ldots, n]$ sont au plus n^3, montrer que l'on peut calculer la parité du nombre d'inversions en $O(n)$.

- Exercice 5 - Paradigme 3. On se donne n sous-ensembles $X_1, \ldots, X_n$ de $[n]$. Une *couverture* est une collection de X_i dont l'union est $[n]$. La *densité* d'un ensemble non vide $Y \subseteq [n]$ est le maximum de $|Y \cap X_i|/|Y|$ pour $i = 1, \ldots, n$.

1. Montrer que s'il existe un ensemble de densité $\varepsilon > 0$, alors toute couverture a taille au moins $1/\varepsilon$.
2. A présent, si tous les ensembles Y ont densité au moins $\varepsilon > 0$, montrer que l'on peut en temps polynomial trouver une couverture de taille $(\ln n)/\varepsilon$. (On rappelle que $\ln(1 + x) \leq x$).
3. Que peut-on dire du problème de couverture?
4. * Quelle est la complexité de calculer un sous-ensemble de densité minimale?

- Exercice 6 - Paradigme 4. Un *réseau de neurones de Hopfield* est une pondération entière w de toutes les paires d'entiers distincts de $[n]$. Ainsi, pour tout $i \neq j$, $w_{ij} = w_{ji}$ est un entier positif ou négatif. Une *configuration* c est une fonction c de $[n]$ dans $\{-1, 1\}$ (il y a donc 2^n configurations). On dit que c est *stable* si pour tout $i \in [n]$ on a $\sum_{j \in [n] \setminus \{i\}} c(i)c(j)w_{ij} \geq 0$.

1. Pour tout $[n]$, proposer un réseau de Hopfield avec tous les w_{ij} négatifs et admettant au moins n configurations stables.
2. Proposer un algorithme faiblement polynomial (i.e. polynomial en $\sum |w_{ij}|$ et pas en $\sum \log |w_{ij}|$) qui calcule une configuration stable.
3. * Est-ce que votre algorithme est fortement polynomial?

- Exercice 7 - Paradigme 5. Votre directeur de stage vient encore d'inventer un nouveau générateur de nombres entiers. Un entier n produit par cette machine est:

- soit premier. **Cas 1**.
- soit admet un entier a premier avec n tel que $a^{n-1} \neq 1 \mod n$. **Cas 2**.

Il vous charge de trouver un algorithme qui admet en entrée un nombre n issu de son générateur, et qui décide entre ces deux cas. Il exige de plus que le nombre total d'opérations effectuées (multiplications et modulo n) soit au plus $c. \log n$ pour une constante c de votre choix.

1. Montrer que dans le cas 2, alors pour tout entier b, au moins un élément x parmi $\{b, ab\}$ vérifie $x^{n-1} \neq 1 \mod n$.
2. On rappelle que si un entier a est premier avec n, alors la multiplication par a est une bijection dans les entiers modulo n. De plus, si n est premier et n ne divise pas a, alors $a^{n-1} = 1 \mod n$. Proposer un algorithme qui satisfait les exigences de votre directeur (et qu'il ne pourra très certainement jamais prendre en défaut).