

Partiel du 12 novembre 2021

Pas de documents. Justifier vos réponses.

- **Exercice 1 - Découpage électoral.** Le parti des programmeurs dynamiques (PPD) se présente aux élections sur deux circonscriptions qui regroupent à elles deux un total de n zones (avec n pair). Chaque zone a m électeurs. Le nombre de voix du PPD dans une zone $i \in [n]$ (on note $[n] := \{1, \dots, n\}$) est estimé à v_i , avec $0 \leq v_i \leq m$. Le PPD peut choisir à sa guise le regroupement des zones en deux circonscriptions de tailles égales. Un *découpage gagnant* est une partition de $[n]$ en deux parties sur lesquelles le PPD a la majorité absolue.

Précisément, c'est une partie $I \subseteq [n]$ de taille $n/2$ telle que $\sum_{i \in I} v_i > m.n/4$ et $\sum_{i \in [n] \setminus I} v_i > m.n/4$.

Par exemple pour $n = 4$ et $m = 100$, si les estimations de vote sont 46, 60, 55, 44, un découpage gagnant est 46+55 et 60+44 (regroupant donc les zones 1 et 3 et les zones 2 et 4).

1. Proposer une instance ayant 6 zones telle qu'il existe un unique découpage gagnant (on choisira ici $m = 100$).
2. Proposer un algorithme en $O(n^3m)$ qui calcule tous les nombres possibles de voix des sous-ensembles de $n/2$ zones.
3. Comment décider s'il existe un découpage gagnant en temps polynomial?

- **Exercice 2 - Tas.** Un tableau d'entiers $T[1 \dots n]$ vérifie la propriété de tas en $i \in [n]$ si $T[i] \geq T[2i]$ (lorsque $2i \leq n$) et $T[i] \geq T[2i+1]$ (lorsque $2i+1 \leq n$). On dit que $T[1 \dots n]$ est un *tas* s'il vérifie la propriété de tas pour tout $i \in [n]$.

1. Montrer que si T est un tas, on peut calculer la quatrième plus grande valeur de T en un nombre constant de lectures d'entrées de T (à préciser).
2. Ecrire ou décrire un algorithme de complexité $O(\log n)$ qui a pour entrée un tableau T vérifiant la propriété de tas pour $i = 2, \dots, n$ et permute ses valeurs pour transformer T en tas. Indication: l'opération élémentaire est l'échange de $T[i]$ avec l'une des valeurs $T[2i], T[2i+1]$.
3. Ecrire ou décrire un algorithme qui permute les valeurs d'un tableau $T[1 \dots n]$ quelconque pour en faire un tas. On étendra itérativement la propriété de tas de l'ensemble $\{k+1, \dots, n\}$ à l'ensemble $\{k, \dots, n\}$.
4. * Montrer que la complexité de l'algorithme précédent est linéaire.

- **Exercice 3 - Indépendant.** On se donne un graphe $G = (V, E)$ ainsi qu'une fonction de poids ω positive définie sur les arêtes. Un sous-ensemble d'arêtes F est *indépendant* si chaque composante connexe de F possède au plus un cycle (i.e. est un arbre ou un arbre plus une arête).

1. Montrer que si G a n sommets, alors un indépendant a au plus n arêtes.

2. Soit F un indépendant. On dit qu'un sommet $v \in V$ est *saturé dans F* s'il appartient à une composante de F qui possède un cycle. Montrer que si e est une arête de $E \setminus F$ qui ne relie pas deux sommets saturés, alors $F \cup \{e\}$ est indépendant.
3. Montrer que l'ensemble des indépendants forme un matroïde.
4. Proposer un algorithme qui calcule un indépendant de poids maximum (on indiquera surtout comment les composantes connexes sont mises à jour).

- Exercice 4 - Polynômes. L'objet de cet exercice est la multiplication rapide de deux polynômes. On appelle n -poly un polynôme de degré $n - 1$, donc avec n coefficients. Soient $P = \sum_{i=0}^{n-1} a_i X^i$ et $Q = \sum_{i=0}^{n-1} b_i X^i$ deux n -polys. Leur produit $R = P \times Q$ est un $(2n - 1)$ -poly. On note $M(n)$ (resp. $A(n)$) le nombre de multiplications (resp. d'additions) entre coefficients pour multiplier deux n -polys.

1. Montrer que $M(n) = n^2$ et $A(n) = (n - 1)^2$ pour l'algorithme usuel
2. On suppose n pair, $n = 2 \times m$. On écrit $P = P_1 + X^m \times P_2$ et $Q = Q_1 + X^m \times Q_2$, où P_1, P_2, Q_1 , et Q_2 sont des m -polys. On définit $R_1 = P_1 \times Q_1$, $R_2 = P_2 \times Q_2$, et $R_3 = (P_1 + P_2) \times (Q_1 + Q_2)$ (donc R_1, R_2 , et R_3 sont des $(n - 1)$ -polys). Exprimer $R = P \times Q$ en fonction de R_1, R_2 , et R_3
3. Calculer $M(n)$ et $A(n)$, en supposant utiliser l'algorithme usuel pour calculer R_1, R_2 , et R_3 .
4. On suppose maintenant que $n = 2^s$ et on utilise récursivement l'approche précédente. Proposer une formule de récurrence pour $M(n)$ et $A(n)$ avec cet algorithme.
5. Calculer l'ordre de grandeur du $M(n)$ précédent.

- Exercice 5 - Arbre binaire. On se donne un arbre binaire A de racine r ayant n noeuds tel que chaque noeud i (interne ou feuille) possède une valeur $v_i > 0$. On note p la *hauteur* de A , c'est à dire la distance maximale de la racine à une feuille. Pour les problèmes suivants, décrire un algorithme de résolution aussi performant que possible, montrer sa validité, et calculer sa complexité.

1. Calculer un sous-ensemble X de noeuds de valeur totale maximale sans relation de parenté directe (i.e. pour toute paire de noeuds x, y de X le parent de y n'est pas x).
2. Calculer un sous-ensemble X de noeuds de valeur totale maximale sans relation de descendance directe (i.e. tout chemin de la racine à une feuille contient au plus un élément de X).
3. Calculer un minimum local de A (dont la valeur est inférieure ou égale à celles de son enfant gauche, son enfant droit, et de son parent, si applicable).
4. *** Calculer un minimum local dans la grille $n \times n$ en temps $O(n)$.