TD 3 : Diviser pour régner

27 Septembre 2018

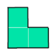
1 k-ième plus petit élément

QUESTION 1 – Étant donné deux tableaux triés A et B de taille respective n et m, écrire une fonction qui retourne le k-ième plus petit élément de l'union des deux tableaux et de complexité $O(log\ n + log\ m)$.

2 Pavage avec des triominos

On considère le problème suivant :

— **Entrée :** un carré de $2^n \times 2^n$ cases et une case particulière L ;

— **Sortie :** le pavage du carré privé de la case L à l'aide de triominos de la forme  .

QUESTION 2 – Proposer un algorithme de type « diviser pour régner » résolvant le problème. Quelle est sa complexité en fonction de n ?

3 Silhouette d'une ville

On cherche à calculer la silhouette d'une ville constituée d'un groupe d'immeubles, vue (en 2 dimensions) par un observateur distant. Chaque immeuble i est représenté par un triplet (h_i, g_i, d_i) où h_i représente la hauteur, g_i l'abscisse gauche et d_i l'abscisse droite. On cherche à écrire un algorithme SILHOUETTE(l) qui, étant donnée une liste de tels triplets, renvoie une liste de paires (x_j, y_j), triés par x_j croissants, et correspondant aux changements de niveau de la silhouette de gauche à droite.

QUESTION 3 – Donner la liste de triplets représentant l'ensemble d'immeubles de la figure 1, ainsi que la liste de paires représentant la silhouette.

QUESTION 4 – Étant données deux silhouettes s_1 et s_2 représentées par une liste de paires, écrire une fonction FUSION-SILHOUETTE(s_1, s_2) qui renvoit une nouvelle liste de paires s, représentant la fusion des deux silhouettes.

QUESTION 5 – En déduire l'écriture de la fonction SILHOUETTE(l), et calculer la complexité associée.

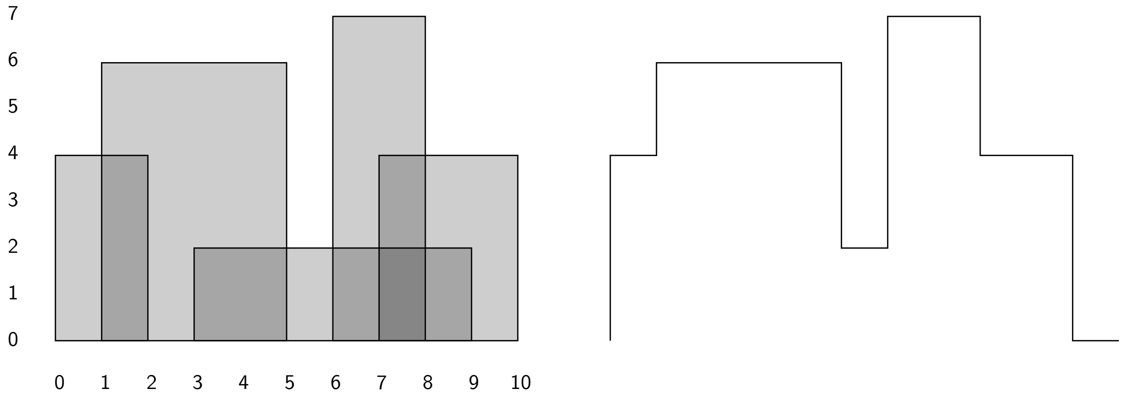


FIGURE 1 – Un ensemble d'immeubles (à gauche) et la silhouette correspondante (à droite).

4 Vote majoritaire

Le but du problème est de trouver dans un tableau T de taille $n = |T|$ l'élément majoritaire, c'est-à-dire apparaissant strictement plus de $\lfloor n/2 \rfloor$ fois, si il en existe un.

QUESTION 6 – Proposer une solution naive (sans diviser). Quelle en est la complexité ?

QUESTION 7 – Proposer une solution utilisant une approche « diviser pour régner » de complexité asymptotique optimale.

QUESTION 8 – Proposer un algorithme de complexité asymptotique optimale, qui fonctionne sans modifier le tableau et à coût mémoire constant.

5 Bonus : Trouver un puits dans un graphe

Rappel : Un graphe orienté $G = (V, E)$ comportant n sommets notés de $v_1 \ldots, v_n$ peut etre représenté par une matrice A telle que

$$a_{ij} = \begin{cases} 1 & \text{si } (v_i \to v_j) \in E \\ 0 & \text{sinon.} \end{cases}$$

Cette matrice est appelée *matrice d'adjacence* du graphe G.

Pour cet exercice, nous supposerons que pour tout $i \in \{1 \ldots, n\}$, $a_{ii} = 0$ et nous considérerons qu'un noeud v_i est un *puits* dans le graphe G ssi :

— pour tout $j \neq i$, $(v_j \to v_i) \in E$
— pour tout $j \in \{1 \ldots, n\}$, $(v_i \to v_j) \notin E$

QUESTION 9 – Étant donné un graphe G représenté par sa matrice d'adjacence A écrire une fonction PUITS(A) qui retourne, s'il existe, un puits de G en temps linéaire (en le nombre de sommets de G).

HINT : nous pourrons considérer une fonction auxiliaire prenant comme paramètres la matrice A ainsi qu'une liste de sommets L. Nous nous intéresserons alors aux puits contenus dans le sous-graphe de G restreint aux sommets de L.