L3 Algorithmique et Programmation, Examen 1

Pas de documents autorisés

Lundi 17 novembre, 15h30-18h30

1 Analyse d'algorithme glouton

Nous avons vu en cours l'algorithme de Kruskal pour le problème de l'arbre couvrant de poids minimum. Le problème est le suivant :

Entrée : graphe non orienté connexe $G = (V, E)$, et poids sur les arêtes : $e \in E \mapsto \ell_e > 0$. On suppose que toutes les arêtes ont des poids distincts.
Sortie : sous-ensemble T de E tel que le graphe (V, T) soit un arbre et dont le poids total, $\sum_{e \in T} \ell_e$ soit minimum parmi les arbres.

L'algorithme est le suivant :

Trier les arêtes par ordre poids croissant. Quitte à renuméroter les arêtes, on a $\ell_{e_1} < \ell_{e_2} < \cdots < \ell_{e_m}$. $T \leftarrow \emptyset$ Pour i allant de 1 à m si $T \cup \{e_i\}$ est acyclique alors $T \leftarrow T \cup \{e_i\}$. Résultat : T.

Démontrer que cet algorithme est correct.

2 Suppression dans les arbres binaires de recherche

Donner le résultat de l'algorithme de suppression dans les arbres binaires de recherche, appliqué pour supprimer l'élément xxx de l'arbre binaire de recherche suivant : yyyy

3 Diviser pour régner

On a deux bases de données B_1 et B_2 qui contiennent chacune n valeurs. On peut y accéder par une requête de la forme $R(k, B_i)$ (avec $1 \leq k \leq n$ et $i \in \{1, 2\}$), qui donne comme résultat la valeur du $k$ième élément de la base de données B_i. Le problème est de trouver la valeur médiane de $B_1 \cup B_2$ avec $O(\log n)$ requêtes.

4 Programmation dynamique

On considère le jeu suivant à deux personnes. Une suite de n cartes sont disposée sur la table, face visible. La carte i a la valeur v_i. On suppose ces valeurs toutes distinctes. Les joueur ramassent tour à tour une carte à l'extrémité gauche

ou droite de l'alignement, jusqu'à ce que toutes les cartes aient été ramassées. Le but pour un joueur est de ramasser des cartes de valeur totale (somme des valeurs) maximum.

1. Montrer que la stratégie gloutonne qui choisit toujours, parmi les deux cartes possibles, celle qui a la valeur la plus élevée, ne maximise pas nécessairement pas la valeur totale des cartes ramassées par le premier joueur.

2. Donner un algorithme en $O(n^2)$ pour calculer la stratégie optimale du premier joueur, c'est-à-dire celle qui lui garantit valeur totale maximale même si le deuxième joueur joue de façon optimale.

5 Structure de données

Concevoir une structure de données permettant de stocker une suite σ de valeurs et de faire les opérations suivantes sur σ, avec complexité $O(\log n)$ par opération si σ a taille n:

1. `inserer(i,x)` : insérer la valeur x en position i, si bien que la suite $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ se transforme en $(\sigma_1, \sigma_2, \ldots, \sigma_{i-1}, x, \sigma_i, \sigma_{i+1} \ldots, \sigma_n)$, suite de longueur $n+1$.

2. `supprimer(i)` : supprimer la valeur en position i

3. `recherche(i)` : retourner comme résultat la valeur x qui se trouve en position i dans σ

4. `remplacer(i,x)` : remplace l'élément de σ en position i par la valeur x

5. `taille` : retourner la taille de σ

6 Graphes

Soit G un graphe non orienté à n sommets et tel que tout sommet ait degré supérieur ou égal à $n/2$. Montrer que G est connexe.

7 Graphes

On considère le problème suivant.

Entrée : graphe non orienté $G = (V, E)$, entier k Sortie : décider s'il existe un sous-ensemble S de V de cardinal au plus k et tel que toute arête de E ait au moins une extrémité dans S.

Démontrer que si v est un sommet de G de degré 0 et $(G \backslash \{v\}, k)$ une instance positive du problème, alors (G, k) est une instance positive du problème.

Démontrer que si v est un sommet de G de degré supérieur ou égal à $k+1$ et $(G \setminus \{v\}, k-1)$ une instance positive du problème, alors (G, k) est une instance positive du problème.

Démontrer que si tous les sommets de G ont degré entre 1 et k et si (G, k) est une instance positive du problmème, alors V a cardinal au plus k^2.

Donner un algorithme qui résoud le problème en temps $O(n^2 f(k))$, où c est une constante indépendante de k (mais l'algorithme n'est pas nécessairement polynomial en k).