IFT2015 :: hiver 2020

Cours de structures de données, Université de Montréal

FÉVRIER 11, 2020FÉVRIER 11, 2020 *by* CSUROSM

Algorithmes de tri interne: exercices

t.1 Tri bébé

a. Quel est le minimum de comparaisons pour trier trois éléments? Donner un algorithme pour trier trois éléments a, b, c avec ce nombre minimum de comparaisons.
b. Quel est le minimum de comparaisons pour trier quatre éléments? Donner un algorithme pour trier quatre éléments a, b, c, d avec ce nombre minimum de comparaisons.

t.2 Séquence bitonique

Le tableau $A[0 \ldots n-1]$ est appellé **bitonique** s'il existe un indice $i \in \{0, 1 \ldots, n-1\}$ tel que $A[0] \leq A[1] \leq \cdots \leq A[i]$ et $A[i] \geq A[i+1] \geq A[i+2] \geq \cdots \geq A[n-1]$. Donner un algorithme pour trier A dans l'ordre croissant en temps linéaire au pire.

Indice. Utiliser un tableau auxiliaire.

t.3 Tri local

On veut trier un tableau $A[0..n-1]$. Le tri correspond à une permutation π des indices $0, \ldots, n-1$ telle que $A[\pi(0)] \leq A[\pi(1)] \leq \cdots \leq A[\pi(n-1)]$. Supposons qu'on sait que le tableau est «presque trié» dans le sens qu'il existe une constante Δ avec $|k - \pi(k)| \leq \Delta$ pour tout k. Dans d'autres mots, le tri déplace chaque élément par tout au plus Δ position. Il n'est pas difficile de voir que le tri par insertion prend $O(n\Delta)$ temps sur un tel tableau presque trié.
a. Modifier le tri par sélection pour qu'il prenne $O(n\Delta)$ temps quand Δ est fourni comme argument.
b. Développer un algorithme de tri qui prend le «désordre» Δ comme argument pour trier le tableau en $O(n \log \Delta)$ temps. L'algorithme ne doit utiliser que $O(\Delta)$ espace de travail.
Indice. Employez une file à priorités de Δ éléments au plus, en liaison avec une fenêtre de taille Δ glissant au long du tableau.

t.4 Vis-écrou

On a un ensemble de n vis et n écrous. On sait que les vis sont de tailles différentes, et qu'il existe exactement un écrou pour chaque vis. On ne peut comparer qu'un écrou E à une vis V à la fois (donc pas de comparaisons écrou-écrou ou vis-vis), pour vérifier que soit E<V, soit E=V, ou bien E>V.

a. Donner un algorithme pour ranger n=2, et un autre pour n=3 paires de vis-écrou.

b. Donner un algorithme pour trouver l'écrou à chaque vis avec O(n log n) comparaisons *en moyenne*.

Indice. Aadaptez l'astuce de partition par pivotage dans Quicksort.

c. Quel est le nombre minimal de comparaisons vis-écrou qu'un algorithme de tri déterministe doit faire?

Indice. Analysez la hauteur de l'arbre de décision: attention, le résultat d'une comparaison entre vis et écrou est ternaire.

Remarque. *Il est très difficile de trouver un algorithme déterministe pour ce problème avec temps linéarithmique. En fait, c'est un problème ouvert de recherche!*

t.6 Megafusion

Dans le tri par fusion, on fusionne deux listes en un temps linéaire. Donner un algorithme pour fusionner k listes triées dans un temps O(n log k) quand la longueur totale des listes est n (donc n est la longueur du résultat).

Indice. Utilisez un tas de taille k, ou développez une solution récursive (par k)

Publicités