Informatique tronc commun
Tableaux et éléments

Sujet

21 octobre 2007

1 Recherche du $k^{\text{ème}}$ élément d'un ensemble

On se donne un tableau de nombres, non trié, dont l'on veut déterminer le $k^{\text{ème}}$ plus petit élément (dans la suite, le « plus petit » sera sous entendu). On va voir un algorithme linéaire en la taille du tableau.

L'algorithme procède récursivement, en éliminant des éléments à chaque étape. Pour ce faire, il commence par déterminer un pivot qui serait suffisement proche du milieu du tableau si il était trié (on verra comment faire plus loin). Puis, il découpe le tableau en deux tas : $T_<$ les éléments strictement plus petits que le pivot, et $T_>$ ceux qui sont plus grands ou égaux. Soit $n_{\leq}T$. Si $n \geq k$, il faut rechercher le $k^{\text{ème}}$ élément de T et sinon le $(k-n)^{\text{ème}}$ élément de T (via un appel récursif).

Maintenant voyons comment déterminer le pivot. On commence par découper le tableau en paquets de 5, dont l'on détermine trivialement le $3^{\text{ème}}$ élément. Cela donne un tableau T'. Maintenant, on recherche via un appel récursif le $|T'|/2^{\text{ème}}$ élément de T', et c'est cet élément que l'on prend comme pivot.

1. Implémenter.
2. Montrer que la complexité est linéaire.
3. Montrer que l'on ne peut pas faire mieux que linéaire, si l'on se restreint à des algorithmes pour lesquels le déroulement ne dépend que de la suite des résultats des comparaisons effectuées.

2 Recherche du $k^{\text{ème}}$ élément de la réunion de deux tableaux triés

On se donne S et T deux tableaux de nombres, triés. On veut déterminer le $k^{\text{ème}}$ élément de la réunion disojointe des deux tableaux.

4. Trouver un algorithme qui tourne en $\Theta(|S| + |T|)$.

Indication : procéder par dichotomie sur les deux tableaux à la fois.

5. Implémenter.

3 Sous-suite commune maximale

On s'intéresse à des suites finies de nombres (par exemple, on pourrait faire ça avec des suites de n'importe quoi en fait).

Étant donné deux suites L_1 et L_2, on veut déterminer une sous-suite comm une aux deux de longueur maximale, que l'on notera $L_1 \cap L_2$ (il peut y en avoir plusieurs de longueur maximale, dans ce cas, on en choisit une arbitrairement).

4 Énoncé

6. Déterminer une relation de récurrence reliant $L_1 \cap L_2$, aux $L'_1 \cap L'_2$, L'_1 et L'_2 étant certaines s ous-listes de L_1 et L_2 respectivement.
7. Cette relation induit un algorithme. Quelle est sa complexité ?
8. Trouver comment implémenter cette récurrence en temps $\mathcal{O}(n^2)$ et en mémoir e $\mathcal{O}(n^2)$, et implémenter.

En fait, avec seulement des tableaux, il n'est pas possible d'atteindre cette ef ficacité. Il faut utiliser des listes, sauf que ça n'est pas au programme, et que les listes de maple, personne ne sait trop si ce sont vraiment des listes. Donc vous pouvez vous contenter de calculer simplement la longeur de la sous-suite co mmune maximale, au lieu de la suite elle-même. Et une fois ce calcul fait, il est en fait possible de recalculer la sous-suite commune maximale sans perte de complexité.