DU ISN - Algorithmique

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Travaux dirigés, séance 3

Le majoritaire

Définition : un élément x est dit *majoritaire* dans un multiensemble E de n éléments, si et seulement si le nombre d'occurences de x dans E est strictement supérieur à $n/2$.
Propriétés :

— il y a **au plus** un élément majoritaire (s'en convaincre).

Si $E = E_1 \cup E_2$ avec E_1 et E_2 de taille n' (si $n = 2 \times n'$) ou de tailles n' et $n'+1$ si $n = 2 \times n' + 1$, alors :

— si x est majoritaire dans E, alors x est majoritaire dans E_1 ou dans E_2.

— si x est majoritaire dans E_1 et dans E_2, alors il est majoritaire dans E.

(le démontrer).
Dans la suite E est un tableau indicé de 1 à n.

Algorithme naïf

```
existe_majoritaire <- faux
pour i de 1 à n
   c <- 0
   j <- 1
   tant que j <=n et non(existe_majoritaire) faire
        si E[j]=E[i] alors c <- c+1
        si (c> n/2) alors
              existe_majoritaire <- vrai
              majo <- E[i]
        j <- j+1
```

Quel est le coût au mieux et au pire de cet algorithme ?

Diviser pour régner

En explorant récursivement les deux moitiés de tableau, écrivez une procédure `majoritaire(i,j)` qui cherche s'il existe un élément majoritaire dans `E[i..j]`. Elle retourne :

— `(-, 0)` s'il n'y a pas de majoritaire dans `E[i..j]`.

— `(x, c)` si `x` est majoritaire avec un nombre d'occurrences égal à `c`.

Etudiez sa complexité.

Pour aller (un tout petit peu) plus loin

Démontrez les propriétés suivantes :

- si une seule des "moitiés" fournit un majoritaire, seul cet élément peut être majoritaire dans le tableau complet.
- si les deux appels récursifs fournissent des majoritaires, qu'ils sont différents, avec un nombre d'occurences différent : le seul qui puisse être majoriatire est celui qui a la plus grand nombre d'occurences.
- dans le cas où n est une puissance de 2, et donc les moitiés toujours de taille égale, s'il y a deux majoritaires différents avec le mêmne nombre d'occurences, alors il ne peut y avoir de majoritaire dans le tableau complet.

En vous plaçant dans le cas où la taille du tableau est une puissance de 2, écrivez une nouvelle version de la fonction `majoritaire` récursive qui tienne compte de ces propriétés.

Encore mieux

Pour améliorer l'algorithme précédent, on suppose qu'on dispose d'un algorithme $\mathcal{A}$ possédant la propriété suivante :

- soit l'algorithme garantit que E ne possède pas d'élément majoritaire ;
- soit l'algorithme fournit un entier $p > n/2$ et un élément x tels que x apparaisse au plus p fois dans E et tout élément autre que x apparaît au plus $n - p$ fois dans E.

1. En supposant cet algorithme $\mathcal{A}$ fourni, en construire un qui vérifie si E possède un élément majoritaire.
2. Donner un algorithme récursif pour $\mathcal{A}$. Quelle est sa complexité ?
   Quelle est celle de l'algorithme qui vérifie si E possède un élément majoritaire ?