TD d'algorithmique numéro 2 : Recherche de l'élément majoritaire
Daniel Hirschkoff et Natacha Portier
semaine du 25 au 29 septembre 2000

Soit $E$ une liste de $n$ éléments rangés dans un tableau numéroté de 1 à $n$. On suppose que la seule opération qu'on sait effectuer sur les éléments est de vérifier si deux éléments sont égaux ou non. On dit qu'un élément $x \in E$ est majoritaire si l'ensemble $E_x = \{y \in E | y = x\}$ a strictement plus de $n/2$ éléments. Sauf avis contraire, on supposera que $n$ est une puissance de 2. On s'intéressera à la complexité dans le pire des cas.

**Question 1** ⋆ algorithme naïf ⋆

Écrire un algorithme calculant le cardinal de $c_x$ de $E_x$ pour un $x$ donné. En déduire un algorithme pour vérifier si $E$ possède un élément majoritaire. Quelle est la complexité de cet algorithme?

**Question 2.1** ⋆ diviser pour régner ⋆

Donner un autre algorithme récursif basé sur un découpage de $E$ en deux listes de même taille. Quelle est sa complexité?

**Question 2.2**

Donner un algorithme similaire (en précisant sa complexité) dans le cas général où $n$ n'est pas une puissance de 2.

**Question 3** ⋆ on améliore encore la complexité ⋆

Pour améliore l'algorithme précédent, on va se contenter dans un premier temps de mettre au point un algorithme 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 $E$.

**Question 3.1**

Donner un algorithme récursif possédant cette propriété. Quelle est sa complexité?

**Question 3.2**

Même question quand $n$ n'est pas une puissance de 2.

**Question 3.3**

En déduire un algorithme efficace vérifiant si $E$ possède un élément majoritaire.