

---

**TD 05 – Révisions**


---

(ElemMaj) **Exercice 1.***Element majoritaire*

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  $E_x = \{y \in E \mid y = x\}$  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.

**1. Algorithme naïf**

Écrire un algorithme calculant le cardinal  $c_x$  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 ?

**2. Diviser pour régner**

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

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

**3. Encore mieux**

Pour améliorer 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  $E$ .

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

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

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

**4. Encore encore mieux**

On change la manière de voir les choses. On a un ensemble de  $n$  balles et on cherche le cas échéant s'il y a une couleur majoritaire parmi les balles.

(a) Supposons que les balles soient rangées en file sur une étagère, de manière à n'avoir jamais deux balles de la même couleur à côté. Que peut-on en déduire sur le nombre maximal de balles de la même couleur ?

On a un ensemble de  $n$  balles, une étagère vide où on peut les ranger en file et une corbeille vide. Considérons l'algorithme suivant :

- **Phase 1** - Prendre les balles une par une pour les ranger sur l'étagère ou dans la corbeille. Si la balle n'est pas de la même couleur que la dernière balle sur l'étagère, la ranger à côté, et si de plus la corbeille n'est pas vide, prendre une balle dans la corbeille et la ranger à côté sur l'étagère. Sinon, c'est à dire si la balle est de la même couleur que la dernière balle sur l'étagère, la mettre dans la corbeille.

- **Phase 2** - Soit  $C$  la couleur de la dernière balle sur l'étagère à la fin de la phase 1. On compare successivement la couleur de la dernière balle sur l'étagère avec  $C$ . Si la couleur est la même on jette les deux dernières balles sur l'étagère, sauf s'il n'en reste qu'une, auquel cas on la met dans la corbeille. Sinon on la jette et on jette une des balles de la corbeille, sauf si la corbeille est déjà vide auquel cas on s'arrête en décrétant qu'il n'y a pas de couleur majoritaire. Quand on a épuisé toutes les balles sur l'étagère, on regarde le contenu de la corbeille. Si elle est vide alors il n'y a pas de couleur majoritaire, et si elle contient au moins une balle alors  $C$  est la couleur majoritaire.

- (b) (*Correction de l'algorithme*) Montrer qu'à tout moment de la phase 1, toutes les balles éventuellement présentes dans la corbeille ont la couleur de la dernière balle de l'étagère. En déduire que s'il y a une couleur dominante alors c'est C.  
Prouver la correction de l'algorithme.
- (c) (*Complexité*) Donner la complexité dans le pire des cas en nombre de comparaisons de couleurs des balles.

## 5. Optimalité

On considère un algorithme de majorité pour les couleurs de  $n$  balles. Le but est de regarder faire l'algorithme, en choisissant au fur et à mesure les couleurs des balles pour que l'algorithme ait le maximum de travail (tout en restant cohérent dans le choix des couleurs). On obtiendra ainsi une borne inférieure de complexité pour un algorithme de majorité. C'est la technique de l'*adversaire*.

À tout moment de l'algorithme, on aura une partition des balles en deux ensembles : l'*arène* et les *gradins*. L'*arène* contient un certain nombre de composantes connexes de deux sortes : les *binômes* et les *troupeaux*. Un binôme est un ensemble de deux balles pour lesquelles l'algorithme a déjà testé si elles étaient de la même couleur et a répondu non. Un troupeau est un ensemble non vide de balles de la même couleur, connectées par des tests de l'algorithme. Ainsi, un troupeau avec  $k$  éléments a subi au moins  $k - 1$  comparaisons de couleurs entre ses membres. Soient  $B$  le nombre de binômes et  $T$  le nombre de troupeaux. Soient  $g$  le nombre d'éléments dans les gradins et  $t$  le nombre total d'éléments dans tous les troupeaux. Enfin, soit  $m = \lfloor n/2 \rfloor + 1$  le "*seuil de majorité*". Au début de l'algorithme toutes les balles sont des troupeaux à un élément. La stratégie de l'*adversaire* est la suivante. L'algorithme effectue un test  $\text{couleur}(x) = \text{couleur}(y)$ ?

1. Si  $x$  ou  $y$  sont dans les gradins, la réponse est non.
  2. Si  $x$  (resp.  $y$ ) est dans un binôme, la réponse est non et  $x$  (resp.  $y$ ) est envoyé dans les gradins alors que l'élément restant devient un troupeau singleton.
  3. Si  $x$  et  $y$  sont dans le même troupeau la réponse est oui.
  4. Si  $x$  et  $y$  sont dans des troupeaux différents alors cela dépend de  $d = B + t$ .
    - (a)  $d > m$  : cela signifie que les balles sont dans des troupeaux singletons. La réponse est non et les balles deviennent un nouveau binôme.
    - (b)  $d = m$  : la réponse est oui et les troupeaux de  $x$  et  $y$  fusionnent.
- (a) Vérifier que les quatre cas précédents traitent tous les cas possibles.  
Montrer qu'à tout moment  $d \geq m$  et que si  $d > m$  alors tous les troupeaux sont des singletons.
- (b) Montrer qu'à tout moment les deux coloriage suivants sont cohérents avec les réponses de l'*adversaire* :
- (i) Toutes les balles sont de couleurs différentes sauf celles qui sont dans un même troupeau.
  - (ii) Une même couleur est attribuée à toutes les balles de tous les troupeaux et à une balle de chaque binôme. Les balles restantes ont chacune une couleur distincte.
- (c) Montrer que si un algorithme correct s'arrête alors l'*arène* ne contient qu'une seule composante connexe qui est un troupeau de taille  $m$ .
- (d) À tout moment le nombre de comparaisons inégales effectuées par l'algorithme est au moins  $2g + B$  et le nombre de comparaisons égales est au moins  $t - T$ .
- (e) Considérons un algorithme qui résout la majorité. Montrer qu'il existe une donnée pour laquelle l'algorithme effectue au moins  $2(n - m) = 2\lceil n/2 \rceil - 1$  comparaisons d'inégalité et au moins  $\lfloor n/2 \rfloor$  comparaisons d'égalité, et donc au moins au total  $3\lceil n/2 \rceil - 2$  comparaisons.

(plssc) **Exercice 2.**

*Plus longue sous suite croissante*

On considère une suite de  $n$  entiers :  $a_1, a_2, \dots, a_n$ . Une sous séquence est un sous ensemble d'entiers de la suite, pris dans le même ordre que la suite :  $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ , avec  $1 \leq i_1 \leq i_2 \leq \dots \leq i_k \leq n$ . Une sous séquence croissante est une sous séquence où les entiers sont strictement croissants. Par exemple, la plus grande sous séquence croissante de la suite 5, 2, 8, 6, 3, 6, 9, 7 est 2, 3, 6, 9.

1. Donnez une plus longue sous séquence croissante de 11, 6, 2, 24, 25, 12, 21, 41, 34, 30.
2. Donnez un algorithme de programmation dynamique permettant de trouver la plus grande sous séquence croissante d'une suite de  $n$  entiers et donnez sa complexité.
3. On peut cependant faire mieux que par programmation dynamique. Proposez un algorithme qui tourne en  $O(n \log n)$ .