Travaux pratiques #2 — Éléments majoritaires

Élément majoritaire *Majority report.*

Étant donné un tableau de n entiers, l'objectif est de déterminer l'élément **majoritaire** (s'il existe). Un élément est **majoritaire** s'il apparaît strictement plus de $n/2$ fois.

1	8	1	1	6	5	1

1	2	1	2	3	4	5

FIGURE 1 – L'élément majoritaire du premier tableau est 1. Le second tableau ne contient pas d'élément majoritaire.

Le tableau est représenté par un objet de la classe `vector`. L'ensemble des méthodes de la classe peuvent être utilisées pour répondre aux questions suivantes.

$\mathcal{Q}_1$. Écrire une fonction permettant de trouver un tel élément en $O(n^2)$ opérations.

$\mathcal{Q}_2$. Proposer une modification de cette fonction pour trouver un tel élément en $O(n \log n)$ opérations.
Indice. La méthode `sort` *de la STL permet de trier un conteneur en* $O(n \log n)$ *opérations.*

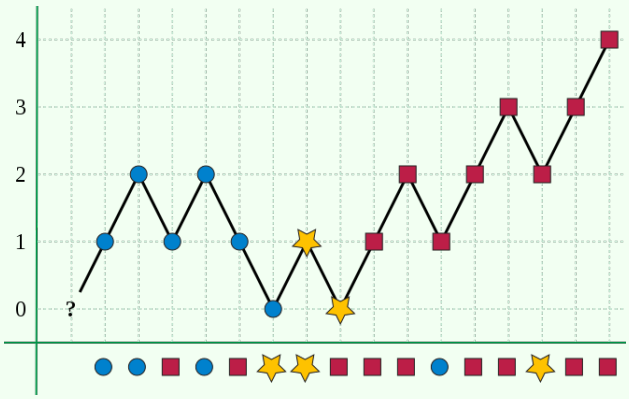
$\mathcal{Q}_3$. En utilisant un type de conteneur supplémentaire, écrire une fonction permettant de résoudre le problème en $O(n)$ opérations. Quel est l'*espace* supplémentaire utilisé par cette fonction ?

Il est possible de trouver un élément majoritaire dans un tableau en temps $O(n)$ et en espace **constant** (il n'est donc pas nécessaire d'utiliser une structure intermédiaire pour résoudre ce problème).

Algorithme de Boyer-Moore. Cet algorithme utilise un unique parcours du tableau pour obtenir la réponse. Le principe est le suivant : l'algorithme met à jour un **compteur** c (initialisé à 0) et un **candidat** m (initialement le premier élément du tableau).

Pour chaque élément t du tableau, plusieurs actions sont possibles : si c vaut 0, alors le candidat m devient t. Sinon, le compteur est incrémenté si m est égal à t, et décrémenté sinon.

Illustration de l'algorithme de Boyer-Moore.



L'ordonnée représente le compteur c, l'abscisse le tableau.

$\mathcal{Q}_1$. Implémenter l'algorithme de Boyer-Moore et l'exécuter sur plusieurs tableaux. Que peut-on constater ?

Élement majoritaire dans un anneau

Lord of the ring.

Un réseau en forme d'**anneau unidirectionnel** est un réseau circulaire où chaque **noeud** peut recevoir des messages de son prédécesseur **uniquement**. Les noeuds possèdent tous un identifiant unique, de type `char` [1].

On se propose de résoudre le problème suivant : chaque noeud choisit aléatoirement un bit (0 ou 1), et l'objectif est de déterminer le bit majoritaire en envoyant des messages **uniquement**. Les messages sont envoyés **simultanément** par tous les noeuds.

$\mathcal{Q}_1$. Implémenter une classe `Anneau` permettant de représenter un anneau unidirectionnel. La classe `list` de la STL est imposée.

Un `Anneau` sera donc constitué d'une liste de `Noeud`, qui sont des objets à définir. Il n'existe pas d'implémentation standard de liste circulaire en C++. Il sera donc nécessaire d'ajouter une méthode à la classe `Anneau`.

$\mathcal{Q}_1$. Implémenter un algorithme permettant de déterminer le bit majoritaire dans un anneau de **taille impaire**.

L'utilisation d'un compteur associé à chaque noeud permet de résoudre assez simplement le problème.

Dans la méthode précédente, le nombre de messages envoyés est relativement élevé puisque chaque noeud doit recevoir autant de messages que l'anneau ne comporte de bits pour déterminer le bit majoritaire.

Il existe un algorithme nécessitant moins d'échanges de messages, basé sur l'observation suivante : si les noeuds agissent comme des **filtres** (selon un critère à déterminer), il est possible de ne conserver que les messages réellement utiles à chaque étape.

$\mathcal{Q}_1$. Implémenter un algorithme permettant de déterminer le bit majoritaire dans un anneau de **taille impaire** sans utiliser de compteur.

1. La taille des anneaux sera donc -très- limitée.