TP8 : Quelques algorithmes

Intersection

1. Dans une classe `BibCollections`, écrivez une méthode `intersection` qui prend en paramètre deux tableaux (List) d'entiers **triés dans l'ordre croissant** et qui renvoie la liste composée de l'intersection des deux tableaux.

Le profil de cette méthode est :
`public static List<Integer> intersection(List<Integer> liste1, List<Integer> liste2)`

Par exemple :

- sur l'entrée `[1, 3, 6, 7, 9, 10], [3, 5, 6, 8, 11]` votre fonction doit renvoyer `[3, 6]`.
- sur l'entrée `[0, 10, 12, 13], [2, 3, 4]` votre fonction doit renvoyer `[]`.
- sur l'entrée `[0, 2, 4, 6, 8, ..., 200 000], [1, 3, 5, 7, ..., 200 001]` votre fonction doit renvoyer `[]` (en un temps raisonnable).
- sur l'entrée `[0, 2, 4, 6, 8, ..., 200 000], [1, 2, 3, 4, ..., 100 000]` votre fonction doit renvoyer `[0, 2, 4, 6, 8, ... 100 000]` (en un temps raisonnable).

2. Ecrivez un exécutable pour tester votre méthode, y compris sur des listes de grande taille (les deux derniers exemples).
3. Une fois que vous êtes certain que votre code est correct, testez-le ci-dessous :

```
1 Votre classe..
```

Envoyer

Intersection ++

1. Dans une classe `BibCollections`, écrivez une méthode `intersection` qui prend en paramètre trois tableaux (List) d'entiers **triés dans l'ordre croissant** et qui renvoie la liste composée de l'intersection des trois tableaux.

Le profil de cette méthode est :

```
public static List<Integer> intersection(List<Integer> liste1, List<Integer> liste2,  List<Integer> liste3
```

Par exemple :

- sur l'entrée `[1, 3, 6, 7, 9, 10], [3, 5, 6, 8, 11], [-6, -3, -2, -1, 0, 1, 3, 5, 6, 7, 14, 17, 22]`, votre fonction doit renvoyer `[3, 6]` (en un temps raisonnable).

2. Ecrivez un exécutable pour tester votre méthode.
3. Une fois que vous êtes certain que votre code est correct, tester le ci-dessous :

```
1 Votre classe..
```

Envoyer

Élément majoritaire

1. Dans une classe `BibCollections`, écrivez une méthode statique `elementMajoritaire` prenant en entrée une liste d'entiers et renvoyant l'élément le plus représenté dans la liste.

Indices : vous pouvez écrire une/des méthodes intermédiares ; pensez à utilisez une Map.

Le profil de cette méthode est :

```
public static Integer elementMajoritaire(List<Integer> liste)
```

Par exemple :

- sur l'entrée `[1, 4, 3, 7, 1, 4, 3, 3, 9, 3, 5, 3]` votre fonction doit renvoyer `3`.
- sur l'entrée `[42, 1, 2, 3, 4, ..., 1 000 000]` votre fonction doit renvoyer `42` (en un temps raisonnable).

2. Testez votre méthode sur des listes de taille d'un million d'éléments.
3. Une fois que vous êtes certain que votre code est correct, tester le ci-dessous :

```
1 Votre classe..
```

Envoyer

Indice égal à la valeur

Écrivez une méthode statique `indiceEgalElement` prenant en entrée une liste triée d'entiers, `T`, et renvoyant vrai si et seulement si il existe un indice `i` tel que `T[i] == i`. Attention, vous devez écrire ici un code qui fonctionne en temps logarithmique.

Par exemple :

- sur l'entrée `[7,9,11,12]` votre méthode doit renvoyer faux.
- sur l'entrée `[-2,1,7,9]` votre méthode doit renvoyer vrai.

Élément unique

Supposons qu'on vous donne une liste d'entiers pour laquelle chaque valeur apparaît exactement deux fois excepté une valeur qui n'apparaît qu'une fois. Vous devez écrire un algorithme qui trouve cet élément.

Note: pouvez vous trouver un algorithme en temps O(n) ? (indice: utilisez le xor bit à bit, réalisé avec l'opérateur `^`)

Bonus

Implémentez tous les exercices du TD8