Exercices sur les vecteurs

28 avril 2023

1 Exercices simples sur des vecteurs

Chaque exercice consiste à réaliser une fonction qui permettra par la suite d'être appelée et également être testée via un test unitaire. Ainsi aucun de ces exercices nécessite d'imprimer ou de lire des informations via la console.

Pour chaque exercice, je vous demande d'essayer de réaliser:

1. hypothèses de départ
2. détail des variables utilisées
3. le pseudo-code
4. l'écriture du code dans un langage de programmation, de préférence Java car celui-ci permet d'utiliser des vecteurs avec des types primitifs et éventuellement Python en utilisant les listes.

1.1 Renvoyez la valeur minimum d'un vecteur d'entiers

Écrivez la fonction « calculMinV(int[] v) » qui devra retourner l'entier le plus petit du vecteur.

1.2 Renvoyer les valeurs minimum et maximum d'un vecteur

Cet exercice est proche du précédent, mais il devra retourner 2 valeurs, les valeurs minimum et maximum. En Java, je vous demande de retourner un vecteur de 2 éléments, le premier élément sera la valeur minimum et le deuxième la valeur maximum:

« calculMinMaxV(int[] v) »

Exemple:

Si à l'appel de la fonction, j'envoie le vecteur | 3 | 1 | 8 | 2 | 22 |,

en retour je dois avoir un résultat avec un vecteur de 2 cases: | 1 | 22 |

1.3 Écrivez une fonction qui calcule le nombre de bosses

« calculNbBosses(int[] v) »

- En entrée, vous avez un vecteur d'entiers
- En sortie, un nombre entier qui indique le nombre de bosses.

Pour avoir une bosse, il faut qu'il y ai une montée suivi d'une descente.

exemples:

1	2	3	==> 0 bosses				
2	2	2	==> 0 bosses				
1	1	2	-1	5	=> 1 bosses		
1	2	1	2	1	2	0	=> 3 bosses

1.4 Écrivez une fonction qui mélange les éléments d'un vecteur

La fonction « melangeV(int[] v) » doit mélanger les éléments du vecteur. Dans votre algorithme, vous pouvez utiliser la fonction

- pseudo-code: générer un nombre entre a et b
- java:

```
Random rand = new Random();
int nbr = rand.nextInt(b);// génère un nombre n : 0<= n <b
```

- Python:

```
import random
random.randint(a,b)# génère un nombre n entre a et b: a<= n <b
```

1.5 Écrivez une fonction qui calcule le nombre de lettres identiques et à la même place entre 2 vecteurs de même taille

Soit la fonction « lettreEnPlace(char[] v1, char[] v2) »:

- Entrée:
  - v1 un vecteur de caractères de taille n
  - v2 un vecteur de caractères de taille n
- Sortie: un entier qui désigne le nombre de lettres du vecteur v2 qui sont à la même place dans le vecteur v1

Exemple:

v1: | I | S | F | C | E |
v2: | O | s | F | E | M |
==>
Réponse: 2 car le s et le F sont à la bonne place (on ne fait pas de différence entre une majuscule et une minuscule)

1.6 Écrivez une fonction qui vérifie qu'il n'existe pas de doublon dans un vecteur

Soit la fonction « sansDoublon(int[] v) » qui doit renvoyer « true » si le vecteur ne contient pas de doublon.
Exemples:

v: | 1 | 7 | 2 | 3 | ==> *true*
v: | 1 | 2 | 7 | 1 | ==> *false* car le 1 existe 2 fois

1.7 Écrivez une fonction qui effectue une rotation d'une case, des éléments du vecteur

Soit la fonction « rotationV1(int[] v, boolean droite) » qui effectue une rotation d'une case vers la droite (droite=*true*) ou vers la gauche (droite=*false*):

v: | 1 | 7 | 2 | 3 | et droite=true ⇒ v: | 3 | 1 | 7 | 2 |

v: | 1 | 7 | 2 | 3 | et droite=false ⇒ v: | 7 | 2 | 3 | 1 |

2 Exercices plus difficiles sur les vecteurs:

2.1 Écrivez une fonction qui retourne la fusion de 2 listes triées:

Considérons 2 listes d'entiers triées par ordre croissant. On vous demande d'écrire le pseudo-code d'une fonction qui retourne une nouvelle liste d'entiers triée qui sera la fusion des 2 listes reçues en entrée. Tenez compte, dans votre algorithme, que les 2 listes initiales sont déjà triées.

Exemple :

L1:	2	5	6	8
L1:	4	6		

Le retour de la fonction devra renvoyer un nouveau vecteur contenant:

L3:	2	4	5	6	6	8

2.2 Écrivez une fonction qui calcule le nombre de lettres en place et à la mauvaise place:

Cet exercice est une poursuite de l'exercice (1.5), cependant dans cet exercice, on recherche également les lettres du vecteur V2 qui existent dans le vecteur V1 mais pas à la bonne place.

Le premier vecteur « v1 » représente une chaîne de caractères de taille n et le $2^{ème}$ vecteur « v2 » représente une chaîne de caractères de même taille. Cette $2^{ème}$chaîne doit être vue comme une proposition d'un joueur pour essayer de trouver la $1^{ère}$chaîne. La fonction va donc renvoyer des informations pour savoir combien de lettres sont à la bonne place et combien sont à la mauvaise place.

Soit la fonction « int[] lettresEnPlaceHorsPlace(char[] v1, char[] v2) » qui doit retourner deux données, soit un tuple (Python) soit un vecteur de 2 entiers en Java (ou un Record) qui contiendra:

- le nombre de lettres en place
- le nombre de lettres à la mauvaise place

Remarques:

- une lettre du vecteur V1 ne peut être utilisée qu'une seule fois

v1:	I	S	F	C	E
v2:	O	E	F	E	I

  ==> Une lettre à sa place et 2 lettres à la mauvaise place

- on vérifie d'abord les lettres en place avant de traiter les lettres à la mauvaise place ainsi le premier « F » de v2 n'est pas considéré comme à la mauvaise place car le « F » de v1 est déjà pris

v1:	I	S	F	I	E
v2:	O	F	F	E	I

  ==> Une lettre à sa place et 2 lettres à la mauvaise place

2.3 Écrivez une fonction qui effectue une rotation de r cases, des éléments du vecteur

Soit la fonction « rotationVr(int[] v, int r) » qui effectue une rotation de « r » cases des éléments du vecteur. Un « r » négatif correspond à une rotation vers la gauche. Le nombre « r » peut être plus grand que la taille du vecteur.

Votre solution **ne doit pas** revenir à effectuer plusieurs rotations d'une case mais bien de mettre chaque élément dans sa case de destination directement.

Exemples:

v:	1	7	2	3	et r=2 ⇒ v:	2	3	1	7
v:	1	7	2	3	et r=-3 ⇒ v:	3	1	7	2
v:	1	7	2	3	et r=6 ⇒ v:	2	3	1	7

Réfléchissez bien avant de vous lancer dans le code!