Exercices sur les entiers/tableaux

Exercice 1. Fusion (tri) de deux tableaux triés

On propose d'écrire une fonction renvoyant un tableau d'entiers triés par ordre croissant correspondant à la fusion de deux tableaux triés :

```
int [] fusionTabTries (int [] t1, int [] t2) {
// Pre-requis :  t1 et t2 sont triés par ordre croissant
int [] tabRes = new int [t1.length+t2.length];
fusionTabTriesAux (t1, 0, t2, 0, tabRes, 0);
return tabRes;
}
```

Question 1.1.

Ecrire l'algorithme récursif :

```
void fusionTabTriesAux (int [] t1, int i1, int [] t2, int i2, int []
tabRes, int iRes) {
// Pre-requis :  tableaux t1 et t2 triés à partir de i1 et i2
// respectivement,
// 0 ≤ i1 ≤ t1.length, 0 ≤ i2 ≤ t2.length, 0 ≤ iRes ≤ tRes.length,
// tabRes.length - iRes = (t1.length - i1) + (t2.length - i2)
//
// Action :  remplit tabRes à partir de l'indice iRes, avec les
// valeurs des sous-tableaux t1[i1..(t1.length-1)] et
// t2[i2..(t2.length-1)] triées par ordre croissant }
```

Exercice 2. Tri d'un tableau à 2 valeurs

Question 2.1.

Ecrire le code de la méthode suivante :

```
int drapeauBicolorAux(int[] t, int i) {
//prérequis :
// t ne contient que des 0 et des 1
// 0 <= i <= t.length
//action :
// modifie t[i..t.length-1] pour placer d'abord tous les 0 puis tous
les 1, et
// retourne l'indice du premier 1 dans le nouveau sous tableau
t[i..length-1] (et retourne t.length si il n'y a pas de 1)

}
```

Par exemple :

- drapeauBicolorAux([1,0,1,1,1,0,0],2) transforme t en [1,0,0,0,1,1,1] et retourne 4
- drapeauBicolorAux([1,0,1,1,1,0,0],5) transforme t en [1,0,1,1,1,0,0] et retourne 7

Question 2.2.
En utilisant la méthode précédente, écrire

```
void drapeauBicolor(int[] t){
//prérequis :
// t ne contient que des 0 et des 1
//action :
// modifie t pour placer d'abord tous les 0 puis tous les 1
}
```

Exercice 3. Tri de pancake On considère une pile de n pancakes de diamètres deux à deux distincts. On souhaite trier cette pile avec les pancakes les plus larges au fond. Pour ce faire, la seule opération dont on dispose est le **flip**, consistant à insérer une spatule où l'on souhaite dans la pile de pancake, et à retourner (en un mouvement très rapide, pour ne pas tout faire tomber!) tous les pancakes au dessus de la spatule. Un exemple de flip est donné sur l'image ci-dessous[1].

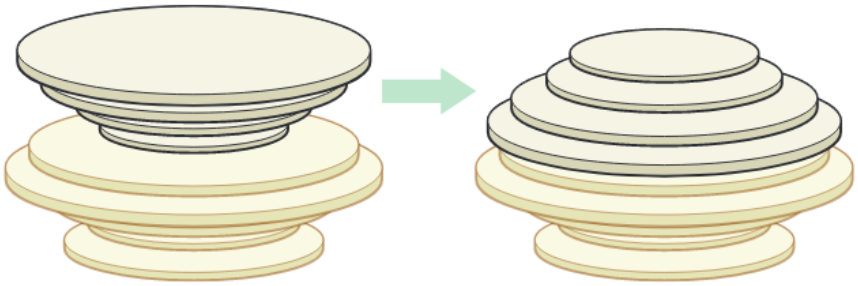

Figure 1: Une opération de flip des 4 pancakes du dessus

.

On modélise ce problème de la façon suivante. La pile de n pancakes est représentée par un tableau t de n entiers, où $t[i]$ représente le diamètre du i-ème pancake, et où la case 0 represente le bas de la pile. On a donc $t[i] \neq t[j]$ pour $i \neq j$, et l'on souhaite trier t par ordre décroissant.

Question 3.1.
Ecrire la méthode suivante (dont l'utilité apparaîtra plus tard) :

```
public static int indiceMax(int[]t, int i){
//prérequis :  0<= i < t.length
//action :  retourne l'indice du plus grand élément de
t[i..t.length-1]
}
```

Par exemple, avec $t = [8, 5, 7, 14, 11, 12, 9, 6, 3]$,

- `indiceMax(t,4)` retourne 5
- `indiceMax(t,1)` retourne 3

Le flip est modélisé de la façon suivante. Pour $0 \leq i < t.length$, on appelle **effectuer un flip en** i l'action de retourner le sous tableau $t[i..(t.length - 1)]$, correspondant au fait de placer la spatule sous le pancake numéro i et d'effectuer le retournement. Sur l'exemple de la figure, en partant de $t = [8, 5, 7, 14, 11, 3, 6, 9, 12]$ (représenté à gauche), et en effectuant un flip en 5, t est modifié pour devenir $t = [8, 5, 7, 14, 11, 12, 9, 6, 3]$.

[1] image tirée du livre "Algorithms", de Jeff Erikson

Question 3.2.
Ecrire la méthode suivante :

```
public static void flipAux(int[]t, int i, int j){
//prérequis :  0<= i < t.length
// et 0<= j < t.length
// (mais on impose pas i<=j)
//action :  retourne le sous tableau t[i..j]
}
```

Par exemple, avec $t = [8, 5, 7, 14, 11, 12, 9, 6, 3]$,

- `flipAux(t,2,5)` transforme t en $[8, 5, 12, 11, 14, 7, 9, 6, 3]$

Question 3.3.
En déduire la méthode suivante :

```
public static void flip(int[]t, int i){
//prérequis :  0<= i < t.length
//action :  effectue un flip en i
}
```

Nous avons maintenant tout les outils pour écrire l'algorithme principal de tri.

Question 3.4.
En utilisant `flip` et `indiceMax`, écrire la méthode suivante :

```
public static void triPancake(int[]t, int i){
//prérequis :  0<= i < t.length
//action :  trie t[i..t.length-1] par ordre décroissant
}
```

Par exemple, avec $t = [8, 5, 7, 14, 11, 12, 9, 6, 3]$,

- `triPancake(t,3)` transforme t en $[8, 5, 7, 14, 12, 11, , 9, 6, 3]$

Exercices de dénombrement

Exercice 4. Nombre de chemins dans une grille Pour un $n > 0$, on considère une grille carrée de $n \times n$ sommets. On oriente le plan de façon usuelle : le sommet en haut à gauche de la grille est aux coordonnées $(0, n-1)$, et le sommet en bas à droite est aux coordonnées $(n-1, 0)$. On appelle un chemin dans une grille une succession de mouvements qui part du sommet $(0, 0)$ pour arriver au sommet $(n-1, n-1)$, chaque mouvement ne pouvant être qu'un pas vers le haut, ou un pas vers la droite. Par exemple, pour $n = 3$, $[(0,0)(1,0)(1,1)(1,2)(2,2)]$ est un chemin, mais $[(0,0)(1,0)(1,1)(0,1)(0,2)(1,2)(2,2)]$ n'est pas un chemin. On dit que deux chemins sont égaux ssi ils empruntent exactement la même succession de positions. Ecrire un algorithme récursif `int nbChemins(int n)` qui pour tout $n \in \mathbb{N}^*$ calcule le nombre de chemins dans une grille $n \times n$. Par exemple, on a nbChemins(3)=6.

Indications : on pourra d'abord écrire un algorithme `int nbChemins(int x,int y, ..)` ... cela est surprenant : il est plus facile de répondre à un problème plus général!.

Exercice 5. Nombre de façons de rendre la monnaie On considère un ensemble de valeur de pièces de monnaies représentées dans un tableau t (par exemple $t = [5, 2, 1]$ pour représenter les pièces de 1, 2 et 5 euros). Ecrire un algorithme récursif `int nbMonnaie(int n, int []t)` qui pour tout $n \in \mathbb{N}$, et pour tout t (contenant des entiers strictement positifs deux à deux distincts) calcule le nombre de façons de rendre exactement n euros en utilisant des pièces dont les valeurs sont dans t. Par exemple, nbMonnaie(7,[1,2,5]) doit retourner 6, car on peut rendre 7 euros selon les façons suivantes : $5+2$, $5+1+1$, $2+2+2+1$, $2+2+1+1+1$, $2+1+1+1+1+1$, $1+1+1+1+1+1+1$. Remarquez que l'ordre ne compte pas, rendre $5+2$ ou $2+5$ est considéré comme une seule solution.

Bonus : affichez toutes les solutions.

Exercices sur les listes

Exercice 6. Deux à deux distincts Ecrire dans la classe Liste une méthode `boolean distincts()` qui retourne vrai ssi la liste ne contient pas deux fois le même entier. Il est intérdit d'écrire une méthode auxiliaire.

Exercices divers

Exercice 7. Tromino Un tromino est une pièce en forme de "L" formée de trois carrés adjacents de taille 1×1 (cf Figure 2). Etant donné un damier vide de $2^n * 2^n$ cases sur lequel il manque une case, le problème du tromino consiste à couvrir toutes les cases du damier avec des trominos, et ce sans chevauchement. Ecrire un algorithme de type diviser pour régner qui pour tout n, et pour toute position manquante sur le damier, résolve le problème du tromino. Remarque : on pourra se contenter de trouver l'idée du diviser pour régner sans écrire complètement l'algorithme (qui nécessiterait de définir des structures de données pour modéliser le damier, les trominos, etc.).

Questions bonus. On ajoute la contrainte suivante : on souhaite de plus colorier les trominos du pavage de telle sorte que deux trominos qui se touchent soient de couleurs différentes. Combien de couleurs sont nécessaires ?

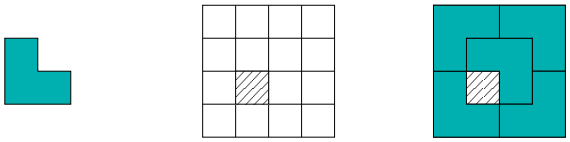

Figure 2: Exemple de damier pour le tromino.

Exercice 8. Retournement On suppose qu'on dispose d'une classe "Pile" qui permet de modéliser des piles (par exemple "(4,1,5)" est une pile d'entiers de hauteur 3, le 4 étant au dessus). On suppose que la classe Pile est munie des méthodes suivantes :

- `public int hauteur()` qui retourne le nombre d'entiers (≥ 0) de la pile
- `public int depile()` qui doit être appelé sur une pile non vide, et qui enlève l'entier du haut de la pile, et le retourne
- `public void empile(int x)` qui ajoute x au sommet de la pile

Ecrire un algorithme récursif "void retourne(Pile p)" qui pour toutr pile p, modifie p pour qu'elle contienne les mêmes entiers, mais dans l'ordre inverse (par exemple (4,1,5) devient (5,1,4)). Attention, il est interdit d'utiliser des boucles, et il est interdit d'écrire une méthode auxiliaire!

Exercice 9. Nono le robot On considère un tableau de 0 et de 1 que l'on veut trier par ordre croissant. Pour modifier ce tableau on dispose d'un robot qui se balade "en dessous" du tableau, qui peut "retirer" du tableau n'importe qu'elle paire d'éléments consécutifs, et emmener cette paire soit tout à gauche, soit tout à droite. Le robot n'a pas le droit de prendre un seul élément au bord, il doit toujours en prendre exactement 2.

Par exemple si le tableau est $[011011]$, le robot peut retirer par exemple la paire $[10]$ en position 2 et 3, le tableau devient $[0111]$, puis placer ce $[10]$ soit tout à gauche, soit tout à droite.

Question : peut on toujours trier ?