Feuille de révisions n°2 - Programmation en OCaml : manipulation de tableaux

1 Pied à l'étrier

R. 2-1 Écrire une fonction calculant l'indice du premier `0` dans un tableau d'entiers parcouru par boucle `while`. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 2-2 Écrire une fonction calculant l'indice du premier `0` dans un tableau d'entiers parcouru par boucle `for`, arrêtée au moyen d'une levée d'exception. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 2-3 Écrire une fonction calculant l'indice du **dernier** `0` dans un tableau d'entiers parcouru par boucle `while`. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 2-4 Écrire une fonction testant l'existence d'un `0` dans un tableau d'entiers avec une boucle `while`, arrêtée dès que possible sans utiliser d'exception.

R. 2-5 Écrire une fonction testant l'existence d'un `0` dans un tableau d'entiers avec une fonctionnelle `Array.iter`, arrêtée par une levée d'exception.

R. 2-6 Écrire une fonction prenant en argument un tableau d'entiers et calculant la somme des éléments du tableau. On fournira une implémentation au moyen d'une boucle `for`.

R. 2-7 Écrire une fonction prenant en argument un tableau d'entiers et calculant la somme des éléments du tableau. On fournira une implémentation au moyen d'un `Array.fold_left`.

R. 2-8 Écrire une fonction calculant l'indice du minimum dans un tableau d'entiers parcouru par une boucle `for`. Cette fonction déclenchera l'exception `Invalid_argument` dans le cas où le tableau est de taille `0`.

R. 2-9 Écrire une fonction calculant l'indice du maximum dans un tableau d'entiers positifs parcouru par une fonctionnelle `Array.fold_left`. Cette fonction déclenchera l'exception `Invalid_argument` dans le cas où le tableau est de taille `0`.

R. 2-10 Écrire une fonction prenant en argument un tableau d'entiers `t` et calculant le tableau des sommes cumulées du tableau `t`. Ainsi la case d'indice `i` du tableau résultat doit contenir `t.(0) + t.(1) + ... + t.(i)`. On fournira une implémentation en $\mathcal{O}(n)$.

R. 2-11 Définir une fonction prenant en argument un tableau de listes d'entiers `t` et le modifiant en ajoutant à chaque liste l'entier indice de la case du tableau dans laquelle se trouve la liste. On itérera sur le tableau au moyen d'une boucle `for`

R. 2-12 Définir une fonction prenant en argument un tableau de listes d'entiers `t` et un entier `x` et retournant un tableau de listes d'entiers obtenu par ajout de l'entier en tête de chacune des listes se trouvant dans le tableau. On itérera sur le tableau au moyen d'un `Array.map`.

R. 2-13 Écrire une fonction prenant en argument un tableau d'entiers t, contenant des valeurs entières dans un intervalle $[\![0, m-1]\!]$ où m vous est passé en argument, et calculant l'entier de $[\![0, m-1]\!]$ ayant le plus d'occurrences dans le tableau. On s'efforcera d'itérer sur les tableaux aux moyens de fonctionnelles et non de boucle `for`/`while`. On fournira une implémentation en $\mathcal{O}(n+m)$.

2 Tris

R. 2-14 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri à bulles♣.

R. 2-15 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri par insertion.

R. 2-16 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri par sélection.

R. 2-17 Écrire une fonction partition prenant en argument un tableau `t`, deux indices `d` (début) et `f` (fin) et `ip` un indice du sous-tableau $\texttt{t}[\texttt{d},\texttt{f}]$ et qui permute $\texttt{t}[\texttt{d},\texttt{f}]$ de manière à placer d'abord les éléments inférieurs à la valeur pivot `t.(ip)`, puis cette valeur pivot, à l'indice `q` qu'elle retournera, et enfin ceux strictement supérieurs à cette valeur.

R. 2-18 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri rapide.

R. 2-19 Écrire une fonction permettant de calculer la médiane d'un tableau d'entiers sans d'abord trier ce tableau. On s'attend à une complexité en $\mathcal{O}(n^2)$.

3 Autres utilisations classiques des tableaux en algorithmique

R. 2-20 On représente une permutation de $[\![0, n-1]\!]$ par un tableau d'entiers `t` tel que l'image de tout entier `i` de $[\![0, n-1]\!]$ par la permutation est `t.(i)`. On représente un cycle (au sens des permutations) par la liste des éléments de son support, dans l'ordre du cycle, ainsi `[0; 1; 2]` est la permutation qui envoie 0 sur 1, 1 sur 2 et 2 sur 0. On rappelle qu'une permutation se décompose de manière unique en un produit de cycles à supports disjoints. On représente un produit de cycles à supports disjoints comme un liste des cycles qui le composent. Leurs supports étant disjoints, ces cycles commutent, ainsi l'ordre dans la liste n'importe pas. Donner une fonction calculant la décomposition d'une permutation en produit de cycles à supports disjoints.

R. 2-21 Écrire une fonction prenant en argument un tableau d'entiers, triés par ordre croissant, de taille n, un entier `p` et testant si l'entier `p` apparaît dans le tableau. L'algorithme devra être récursif et ne pas manipuler de boucles `while`/`for`. On assurera une complexité en $\mathcal{O}(\log(n))$.

R. 2-22 Écrire une fonction prenant en argument un tableau d'entiers, triés par ordre croissant, de taille n, un entier `p` et testant si l'entier `p` apparaît dans le tableau. L'algorithme utilisé ne devra pas être récursif. On assurera une complexité en $\mathcal{O}(\log(n))$.

R. 2-23 Écrire une fonction prenant en argument un tableau de booléens représentant un entier écrit en base 2 (les bits de poids faibles sont à droite) et modifiant le tableau pour qu'il représente l'entier suivant. On lèvera l'exception `Invalid_argument "dernier"` s'il n'est pas possible d'incrémenter l'entier.

R. 2-24 Écrire une fonction prenant en argument un tableau `t` de taille n et une valeur par défaut `x` de même type que celui des éléments du tableau et retournant un tableau de taille $2n$ dont les n premiers éléments sont ceux de `t` et les n derniers contiennent la valeur par défaut.

R. 2-25 Implémenter une structure de table dynamique, c'est-à-dire une structure de tableau telle que l'ajout et la suppression d'éléments en fin de tableau ont un coût amorti constant.

♣. https://fr.wikipedia.org/wiki/Tri_à_bulles