TP n°3 - Permutations

Soit n un entier naturel non nul. Une *permutation* σ de l'ensemble $[\![0, n-1]\!]$ des entiers compris entre 0 et $n-1$ est une bijection de cet ensemble dans lui-même.

1 Représentation d'une permutation

Nous représenterons une permutation σ de $[\![0, n-1]\!]$ par un tableau d'entiers t de longueur n, tel que pour tout $i \in [\![0, n-1]\!]$, l'entier $t.(i)$ soit égal à $\sigma(i)$.

Q1 À quelle condition un tableau t de longueur n représente-t-il une permutation de $[\![0, n-1]\!]$?
En déduire une fonction `est_permutation` qui vérifie si un tableau d'entiers représente une permutation.

Q2 Le support d'une permutation σ est l'ensemble des points non fixes de σ, c'est-à-dire l'ensemble des entiers i tels que $\sigma(i) \neq i$.
Écrire une fonction `support` qui prend en argument un tableau représentant une permutation et qui renvoie son support.

Q3 La composée de deux permutations σ et σ' de $[\![0, n-1]\!]$ est leur composée $\sigma \circ \sigma'$ au sens habituel des applications. Il s'agit d'une permutation de $[\![0, n-1]\!]$.
Écrire une fonction `compose` qui prend en argument deux tableaux représentant des permutations et qui renvoie le tableau représentant leur composée.

2 Énumération des permutations

2.1 Ordre sur les permutations

On ordonne les permutations de $[\![0, n-1]\!]$ selon l'ordre lexicographique. Nous allons écrire une fonction qui étant donnée une permutation représentée par un tableau t calcule son successeur immédiat. Pour cela, on applique l'algorithme suivant :

1. On détermine le plus grand entier i tel que $t.(i) < t.(i+1)$;
2. On permute $t.(i)$ avec le plus petit élément qui lui soit plus grand parmi $t.(i+1)$,..., $t.(n-1)$;
3. On trie la portion du tableau située à partir de l'indice $i+1$ compris.

Q4 Écrire une fonction `indice_max_croit` qui prend en argument un tableau t et renvoie le plus grand indice i tel que $t.(i) < t.(i+1)$.

Q5 Écrire une fonction `permute_max` qui prend en argument un tableau t et un entier i et qui permute dans le tableau t l'élément d'indice i avec le plus petit élément qui lui soit plus grand parmi $t.(i+1)$,..., $t.(n-1)$ (ou n est la longueur de t).

Q6 Écrire une fonction `trie_droite` qui prend en argument un tableau t et un entier k et qui trie en place la portion du tableau située à partir de l'indice k.

Q7 En déduire une fonction `suivant` qui prend en argument un tableau t représentant une permutation et modifie le tableau t en place de manière à obtenir la permutation suivante.

2.2 Affichage des permutations

Q8 Écrire une fonction `print_perm` prenant en argument une permutation et l'affiche sous la forme de votre choix.

Q9 En déduire une fonction `enumere` qui prend en argument un entier n et qui affiche toutes les permutations de $[\![0, n-1]\!]$. Combien y en a-t-il ?

3 Cycles

On appelle *cycle* une permutation σ tel qu'il existe k éléments distincts $x_0, \dots, x_{k-1}$ tels que le support de σ est $\{x_0, \dots, x_{k-1}\}$, et tels que $\sigma(x_0) = x_1$, $\sigma(x_1) = x_2$, $\dots$, $\sigma(x_{k-2}) = x_{k-1}$ et $\sigma(x_{k-1}) = x_0$.

Un tel cycle peut être représenté par la liste $[x_0; \dots; x_{k-1}]$.

On peut démontrer que toute permutation se décompose de manière unique comme un produit (commutatif) de cycles de supports disjoints. On représentera un produit de cycles sous la forme d'une liste de listes d'entiers.

Q10 Décomposer (à la main) la permutation `[|4; 6; 7; 1; 2; 5; 3; 0|]` en un produit de cycles disjoints.

Q11 Écrire une fonction `compose_cycle` prenant en argument un cycle c et un tableau t représentant une permutation pour laquelle on suppose que les éléments du cycle sont des points fixes, et qui modifie en place le tableau t de sorte que le tableau obtenu représente la composée du cycle et de la permutation initiale.

Q12 En déduire une fonction `produit_cycles` qui prend pour argument un produit de cycles et l'entier n et qui renvoie la permutation de $[\![0, n-1]\!]$ représentée par le produit de cycles.

Q13 On souhaite maintenant procéder à l'opération inverse. Écrire une fonction `cycles` qui prend en argument une permutation et qui décompose cette permutation en un produit de cycles de supports disjoints.