Boucles et listes

Produit de matrices

Dans cet exercice on code une matrice dans une liste de listes.

1. Écrire une fonction qui prend en entrée deux entiers n et p et retourne la matrice de taille $n \times p$ dont toutes les entrées sont nulles.
   Important : bien vérifier que toutes les entrées de la matrice obtenue sont indépendantes (pour cela, modifier une entrée de la matrice obtenue et s'assurer que les autres entrées de la matrice sont inchangées).
2. Écrire une fonction qui prend en entrée A et B deux matrices de même taille et retourne la somme de A et B. Faire de même pour le produit (on supposera dans ce cas que le nombre de colonnes de A est égal au nombre de lignes de B).

Opérations sur les sous-ensembles finis de $\mathbb{N}$

On veut manipuler des sous-ensembles finis de $\mathbb{N}$. La *première représentation* d'un ensemble fini A est la suivante :

— Si $A = \emptyset$, sa première représentation est la liste vide `[]` ;

— Si $A \neq \emptyset$, notons m son plus grand élément. La première représentation de A est la liste $[c_0, c_1, \ldots, c_m]$ où $c_i = 1$ si $i \in A$ et $c_i = 0$ si $i \notin A$.

Par exemple, la première représentation de $\{2, 4, 5\}$ est la liste `[0,0,1,0,1,1]`.

1. Écrire une fonction Python `cardinal(R)` qui étant donné la première représentation `R` d'un ensemble A retourne le nombre d'éléments de A.
2. Écrire une fonction Python `intersection(R,S)` qui étant donné la première représentation `R` d'un ensemble A et la première représentation `S` d'un ensemble B retourne la première représentation de $A \cap B$.
3. Faire de même pour l'union.

On considère maintenant une autre représentation des sous-ensembles finis de $\mathbb{N}$. La *seconde représentation* de A est la liste des éléments de A ordonnés par ordre croissant. Par exemple, si $A = \{2, 4, 5\}$, la seconde représentation de A est la liste `[2,4,5]`.

3. Écrire une fonction `conv1vers2(R1)` qui prend en entrée la première représentation d'un ensemble fini A et retourne la seconde représentation de A.
4. Écrire une fonction `conv2vers1(R2)` qui effectue la conversion dans l'autre sens.
5. À partir des fonctions précédentes, comment peut-on calculer la seconde représentation de l'union et de l'intersection de deux ensembles donnés par leur seconde représentation ? Écrire les fonction correspondante. Les fonctions obtenues ainsi sont-elles efficaces ?

Codes de Gray

Soit $A_n = \{0,1\}^n$ l'ensemble des 2^n mots (ou uplets) composés de 0 et de 1 de longueur n. Un *code de Gray* de dimension n est une façon d'écrire tous les mots de A_n sous la forme d'une suite de telle sorte que deux mots consécutifs de cette suite n'ont qu'une seule lettre de différence. Par exemple, pour $n = 3$, la suite d'éléments ci-dessous forme un code de Gray :

$$000, 001, 011, 010, 110, 111, 101, 100.$$

Dans la suite, on codera les mots par des listes (exemple, `[1,0,1,1]` pour 1011). Un code de Gray sera donc vu comme une liste de listes.

1. Écrire une fonction `poids(x)` prenant en entrée une liste x et qui renvoie le poids de x c'est-à-dire le nombre de 1 dans x. Par exemple, `poids([1,1,0,0,0,1,0])` renverra 3.

On définit ci-dessous un code de Gray G_n. Son premier élément est $[0, 0, \ldots, 0]$ (liste de n zéros) et étant donné une liste x de taille n, l'élément y venant immédiatement après x dans la liste G_n est défini comme ceci :

— Si x est de poids pair alors y est égal à x sauf pour son dernier élément $y[n-1]$ qui est remplacé par $1 - x[n-1]$;

— Si x est de poids impair et différent de $[1, 0, \ldots, 0]$: soit i l'indice défini par $x[i] = 1$ et $x[j] = 0$ pour tout $j > i$. L'élément y est alors égal à x sauf pour l'élément $y[i-1]$ qui est remplacé par $1 - x[i-1]$;

— L'élément $x = [1, 0, ..., 0]$ est le dernier élément du code et on pose par convention $y = x$.

On admet que G_n ainsi défini est bien un code de Gray.

2. Écrire une fonction `successeurGray(n,x)` qui prend en entrée un élément x et un entier n et renvoie l'élément y successeur de x dans le code de Gray G_n.
3. Écrire une fonction `gray(n)` qui renvoie le code de Gray G_n en partant de l'élément $x = [0, ..., 0]$ et en appliquant successivement la fonction `successeurGray(n,x)` jusqu'à la production de tous les mots du code.

Suites de Syracuse

Soit $f : \mathbb{N} \to \mathbb{N}$ définie par

$$f(k) = \begin{cases} \frac{k}{2} \text{ si } k \text{ est pair} \\ 3k+1 \text{ sinon} \end{cases}$$

Pour un entier $p > 0$, on définit la suite $(s_n(p))_{n\in\mathbb{N}}$ par $s_0(p) = p$ et pour tout n, $s_{n+1}(x) = f(s_n(x))$. La conjecture dite de Syracuse énonce que toutes les suites $(s_n(p))_n$ atteignent 1.

1. Écrire une fonction `succ(k)` prenant en entrée un entier k et retournant $f(k)$.
2. Écrire une fonction `syracuse(p,n)` prenant en entrée deux entiers positifs k et n et renvoyant la liste des n premiers termes de la suite $(s_n(p))_n$.

On suppose maintenant que la conjecture de Syracuse est vraie.

3. Écrire une fonction `temps(p)` prenant en entrée un entier $p > 0$ et renvoyant le plus petit entier n tel que $s_n(p) = 1$. Que peut-il se passer à l'exécution de cette fonction si la conjecture de Syracuse n'est pas vraie ?
4. Écrire une fonction `altitude(p)` retournant $\max_{n\in\mathbb{N}}(s_n(p))$.